

# Rețele Locale

Note de curs și lucrări de laborator

Tiberiu T. Cociaș



Universitatea  
Transilvania  
din Brașov

2026

# Cuprins

---

---

|          |                                                             |          |
|----------|-------------------------------------------------------------|----------|
| <b>1</b> | <b>Transmisiuni analogice și digitale</b>                   | <b>3</b> |
| 1.1      | Despre ce este vorba, de fapt . . . . .                     | 3        |
| 1.1.1    | Ce veți putea face după acest curs . . . . .                | 4        |
| 1.2      | Vocabularul minim: bit, octet, semnal . . . . .             | 4        |
| 1.2.1    | Capcana unităților . . . . .                                | 5        |
| 1.3      | Cât de mare e o rețea: LAN, MAN, WAN . . . . .              | 6        |
| 1.4      | Echipamentele și nivelul la care lucrează . . . . .         | 6        |
| 1.4.1    | Interfața de rețea . . . . .                                | 8        |
| 1.5      | Protocolul: de ce trebuie să vorbim aceeași limbă . . . . . | 8        |
| 1.6      | De ce comunicăm pe niveluri . . . . .                       | 9        |
| 1.6.1    | Modelul OSI: șapte niveluri . . . . .                       | 10       |
| 1.6.2    | Stiva TCP/IP: patru niveluri, dar reale . . . . .           | 10       |
| 1.6.3    | Încapsularea: cum coboară un mesaj prin stivă . . . . .     | 10       |
| 1.7      | Nivelul fizic și cele patru sarcini ale lui . . . . .       | 11       |
| 1.7.1    | Patru combinații de date și semnal . . . . .                | 12       |
| 1.8      | Codări de linie: cum arată un bit pe fir . . . . .          | 12       |
| 1.8.1    | Ce trebuie să observați, pe rând . . . . .                  | 13       |
| 1.8.2    | Codarea în bloc: 4B/5B . . . . .                            | 13       |
| 1.8.3    | Cum se combină în tehnologiile reale . . . . .              | 15       |
| 1.9      | Modulația: date digitale peste semnal analogic . . . . .    | 15       |
| 1.9.1    | Rata de bit nu este rata de simbol . . . . .                | 16       |
| 1.10     | Greșeli frecvente . . . . .                                 | 17       |
| 1.11     | Rezumat și glosar . . . . .                                 | 18       |

|          |                                                            |           |
|----------|------------------------------------------------------------|-----------|
| 1.12     | Ce urmează și unde citiți mai departe . . . . .            | 19        |
| <b>2</b> | <b>Medii de transmisie</b>                                 | <b>23</b> |
| 2.1      | Din cursul trecut . . . . .                                | 23        |
| 2.1.1    | Ce veți putea face după acest curs . . . . .               | 23        |
| 2.2      | Trei familii de medii . . . . .                            | 24        |
| 2.3      | Cablul coaxial . . . . .                                   | 24        |
| 2.4      | Cablul torsadat: de ce se răsucesc firele . . . . .        | 25        |
| 2.4.1    | Cu ecran sau fără . . . . .                                | 26        |
| 2.4.2    | Categoriile . . . . .                                      | 26        |
| 2.5      | Cele trei mufări clasice . . . . .                         | 27        |
| 2.6      | Fibra optică . . . . .                                     | 28        |
| 2.6.1    | Ferestrele de lungime de undă . . . . .                    | 28        |
| 2.7      | Mediul fără fir . . . . .                                  | 30        |
| 2.8      | Multiplexarea: mai multe conversații pe un mediu . . . . . | 31        |
| 2.9      | Performanța unei legături . . . . .                        | 32        |
| 2.9.1    | Cele patru componente ale latenței . . . . .               | 32        |
| 2.10     | Ce strică semnalul pe drum . . . . .                       | 33        |
| 2.11     | Greșeli frecvente . . . . .                                | 34        |
| 2.12     | Rezumat și glosar . . . . .                                | 35        |
| 2.13     | Ce urmează și unde citiți mai departe . . . . .            | 35        |
| <b>3</b> | <b>Nivelul legătură de date</b>                            | <b>39</b> |
| 3.1      | Din cursurile trecute . . . . .                            | 39        |
| 3.1.1    | Ce veți putea face după acest curs . . . . .               | 39        |
| 3.2      | Ce îi lipsește nivelului fizic . . . . .                   | 40        |
| 3.3      | Cele două subniveluri: LLC și MAC . . . . .                | 40        |
| 3.4      | Încapsularea la nivelul 2 . . . . .                        | 41        |
| 3.5      | Cadrul Ethernet, câmp cu câmp . . . . .                    | 41        |
| 3.6      | Adresa MAC . . . . .                                       | 42        |
| 3.6.1    | Cele trei tipuri de adrese . . . . .                       | 43        |
| 3.7      | Timpii caracteristici ai unei rețele Ethernet . . . . .    | 44        |
| 3.8      | Mediul partajat și CSMA/CD . . . . .                       | 45        |
| 3.9      | Domenii de coliziune și de broadcast . . . . .             | 46        |
| 3.10     | Cum funcționează un switch . . . . .                       | 47        |
| 3.10.1   | Îmbătrânirea intrărilor . . . . .                          | 48        |
| 3.11     | Metode de comutare . . . . .                               | 48        |
| 3.12     | Familia Ethernet, pe scurt . . . . .                       | 49        |
| 3.13     | Greșeli frecvente . . . . .                                | 50        |
| 3.14     | Rezumat și glosar . . . . .                                | 50        |
| 3.15     | Ce urmează și unde citiți mai departe . . . . .            | 50        |

|          |                                                               |           |
|----------|---------------------------------------------------------------|-----------|
| <b>4</b> | <b>Optimizarea rețelelor locale</b>                           | <b>53</b> |
| 4.1      | Din cursul trecut . . . . .                                   | 53        |
| 4.1.1    | Ce veți putea face după acest curs . . . . .                  | 53        |
| 4.2      | Ce nu merge într-un LAN plat . . . . .                        | 54        |
| 4.2.1    | De ce nu pur și simplu un ruter? . . . . .                    | 55        |
| 4.3      | Ce este un VLAN . . . . .                                     | 55        |
| 4.4      | Porturi de acces și legături trunk . . . . .                  | 55        |
| 4.4.1    | Problema . . . . .                                            | 56        |
| 4.4.2    | Soluția: legătura trunk . . . . .                             | 56        |
| 4.4.3    | Eticheta 802.1Q . . . . .                                     | 56        |
| 4.5      | VLAN-ul nativ . . . . .                                       | 57        |
| 4.6      | Rutarea între VLAN-uri . . . . .                              | 57        |
| 4.6.1    | Soluția clasică: o interfață per VLAN . . . . .               | 58        |
| 4.6.2    | Soluția router-on-a-stick . . . . .                           | 58        |
| 4.7      | Redundanța și problema buclelor . . . . .                     | 59        |
| 4.8      | Spanning Tree: rolurile și costurile . . . . .                | 59        |
| 4.8.1    | Cele patru roluri . . . . .                                   | 60        |
| 4.8.2    | Bridge ID: cine devine rădăcină . . . . .                     | 60        |
| 4.8.3    | Costurile legăturilor . . . . .                               | 60        |
| 4.9      | Cei patru pași ai algoritmului . . . . .                      | 61        |
| 4.10     | Stările porturilor și timpul de convergență . . . . .         | 63        |
| 4.11     | Variantele moderne de STP . . . . .                           | 63        |
| 4.12     | EtherChannel: mai multe cabluri, o singură legătură . . . . . | 64        |
| 4.12.1   | Protocoalele de negociere . . . . .                           | 65        |
| 4.13     | Greșeli frecvente . . . . .                                   | 65        |
| 4.14     | Rezumat și glosar . . . . .                                   | 66        |
| 4.15     | Ce urmează și unde citiți mai departe . . . . .               | 66        |
| <b>5</b> | <b>Protocolul IP</b>                                          | <b>69</b> |
| 5.1      | Din cursurile trecute . . . . .                               | 69        |
| 5.1.1    | Ce veți putea face după acest curs . . . . .                  | 69        |
| 5.2      | De ce mai avem nevoie de un set de adrese . . . . .           | 70        |
| 5.3      | Cele trei funcții ale nivelului rețea . . . . .               | 70        |
| 5.4      | Antetul IPv4 . . . . .                                        | 71        |
| 5.5      | Adresa, masca și cele două operații . . . . .                 | 72        |
| 5.5.1    | Notăția CIDR . . . . .                                        | 72        |
| 5.6      | Clase, adrese private, adrese rezervate . . . . .             | 73        |
| 5.6.1    | Adrese private . . . . .                                      | 74        |
| 5.6.2    | Adrese cu semnificație specială . . . . .                     | 74        |
| 5.7      | Subnetarea . . . . .                                          | 75        |
| 5.7.1    | Varianta simplă: subrețele egale . . . . .                    | 75        |



|          |                                                          |           |
|----------|----------------------------------------------------------|-----------|
| 5.8      | VLSM: subnetarea optimă . . . . .                        | 76        |
| 5.9      | ARP: de la adresa IP la adresa MAC . . . . .             | 78        |
| 5.9.1    | Ce se întâmplă când destinația e în altă rețea . . . . . | 78        |
| 5.10     | DHCP: adrese fără configurare manuală . . . . .          | 79        |
| 5.10.1   | DHCP relay . . . . .                                     | 79        |
| 5.11     | Greșeli frecvente . . . . .                              | 80        |
| 5.12     | Rezumat și glosar . . . . .                              | 81        |
| 5.13     | Ce urmează și unde citiți mai departe . . . . .          | 81        |
| <b>6</b> | <b>Rutare</b>                                            | <b>85</b> |
| 6.1      | Din cursul trecut . . . . .                              | 85        |
| 6.1.1    | Ce veți putea face după acest curs . . . . .             | 85        |
| 6.2      | Ce este, de fapt, un ruter . . . . .                     | 86        |
| 6.3      | Ce este o rută și ce e metrica . . . . .                 | 87        |
| 6.4      | De unde vin rutele . . . . .                             | 87        |
| 6.5      | Distanța administrativă . . . . .                        | 88        |
| 6.5.1    | Regula completă de selecție . . . . .                    | 88        |
| 6.6      | Tabela de rutare și procesul de decizie . . . . .        | 90        |
| 6.7      | Ruta implicită . . . . .                                 | 91        |
| 6.8      | Rute statice . . . . .                                   | 92        |
| 6.8.1    | Ruta flotantă . . . . .                                  | 93        |
| 6.9      | Traseul complet al unui pachet . . . . .                 | 94        |
| 6.10     | Sumarizarea rutelor și interfața nulă . . . . .          | 94        |
| 6.10.1   | Cum se calculează . . . . .                              | 94        |
| 6.10.2   | Efectul secundar și interfața nulă . . . . .             | 96        |
| 6.11     | Greșeli frecvente . . . . .                              | 96        |
| 6.12     | Rezumat și glosar . . . . .                              | 96        |
| 6.13     | Ce urmează și unde citiți mai departe . . . . .          | 97        |
| <b>7</b> | <b>Liste de acces</b>                                    | <b>99</b> |
| 7.1      | Din cursul trecut . . . . .                              | 99        |
| 7.1.1    | Ce veți putea face după acest curs . . . . .             | 100       |
| 7.2      | Ce este o listă de acces . . . . .                       | 100       |
| 7.2.1    | Filtrarea nu este singura utilizare . . . . .            | 100       |
| 7.2.2    | Ce poate „vedea” o listă dintr-un pachet . . . . .       | 101       |
| 7.3      | Ruter cu ACL sau firewall dedicat? . . . . .             | 102       |
| 7.3.1    | Unde se pun serviciile expuse public . . . . .           | 102       |
| 7.4      | Cum se citește o listă de reguli . . . . .               | 102       |
| 7.4.1    | Ordinea de parcurgere, ca exercițiu . . . . .            | 103       |
| 7.5      | Unde se aplică o listă . . . . .                         | 103       |
| 7.6      | Masca wildcard . . . . .                                 | 105       |

|          |                                                             |            |
|----------|-------------------------------------------------------------|------------|
| 7.6.1    | Cuvintele cheie care scurtează scrierea . . . . .           | 106        |
| 7.6.2    | Calculator: de la prefix la wildcard și înapoi . . . . .    | 106        |
| 7.7      | Listele standard . . . . .                                  | 106        |
| 7.7.1    | De ce lista standard se plasează lângă destinație . . . . . | 107        |
| 7.8      | Listele extinse . . . . .                                   | 107        |
| 7.8.1    | Protocoale și porturi uzuale . . . . .                      | 109        |
| 7.8.2    | Evaluator interactiv . . . . .                              | 109        |
| 7.8.3    | Unde se plasează lista extinsă . . . . .                    | 110        |
| 7.9      | Listele cu nume și editarea lor . . . . .                   | 111        |
| 7.10     | Liste care țin cont de context . . . . .                    | 112        |
| 7.10.1   | Problema traficului de întoarcere . . . . .                 | 112        |
| 7.10.2   | Soluția minimală: <b>established</b> . . . . .              | 112        |
| 7.10.3   | Listele reflexive . . . . .                                 | 113        |
| 7.10.4   | Liste condiționate temporal . . . . .                       | 113        |
| 7.11     | Verificare, jurnalizare, comentarii . . . . .               | 114        |
| 7.12     | Protejarea accesului la echipament . . . . .                | 114        |
| 7.13     | Greșeli frecvente . . . . .                                 | 115        |
| 7.14     | Rezumat și glosar . . . . .                                 | 116        |
| 7.15     | Ce urmează . . . . .                                        | 116        |
| <b>8</b> | <b>Securizarea rețelei</b>                                  | <b>119</b> |
| 8.1      | Din cursul trecut . . . . .                                 | 119        |
| 8.1.1    | Ce veți putea face după acest curs . . . . .                | 120        |
| 8.2      | Nivelul transport: de ce există porturi . . . . .           | 120        |
| 8.2.1    | Intervalele de porturi . . . . .                            | 121        |
| 8.3      | Antetul TCP, câmp cu câmp . . . . .                         | 122        |
| 8.4      | Deschiderea și închiderea unei conexiuni . . . . .          | 123        |
| 8.5      | Cele trei familii de atacuri . . . . .                      | 123        |
| 8.5.1    | Cum funcționează un SYN flood . . . . .                     | 124        |
| 8.6      | iptables: tabele și lanțuri . . . . .                       | 125        |
| 8.7      | Anatomia unei reguli iptables . . . . .                     | 126        |
| 8.8      | Politica implicită și un firewall complet . . . . .         | 129        |
| 8.9      | Arhitectura perimetrului . . . . .                          | 131        |
| 8.9.1    | IDS și IPS . . . . .                                        | 131        |
| 8.10     | SSH și cele trei concepte de securitate . . . . .           | 132        |
| 8.10.1   | Confidențialitatea: problema cheii comune . . . . .         | 132        |
| 8.10.2   | Autentificarea: parolă sau cheie . . . . .                  | 134        |
| 8.10.3   | Integritatea: codul MAC . . . . .                           | 135        |
| 8.11     | Greșeli frecvente . . . . .                                 | 135        |
| 8.12     | Rezumat și glosar . . . . .                                 | 135        |
| 8.13     | Ce urmează . . . . .                                        | 136        |

|           |                                                         |            |
|-----------|---------------------------------------------------------|------------|
| <b>9</b>  | <b>NAT și tunelare</b>                                  | <b>139</b> |
| 9.1       | Din cursurile trecute . . . . .                         | 139        |
| 9.1.1     | Ce veți putea face după acest curs . . . . .            | 140        |
| 9.2       | Problema: de ce s-au terminat adresele . . . . .        | 140        |
| 9.3       | Ce face, concret, translatarea . . . . .                | 141        |
| 9.4       | Cele trei tipuri de translatare . . . . .               | 141        |
| 9.4.1     | NAT static - unu la unu . . . . .                       | 141        |
| 9.4.2     | NAT dinamic - n la m . . . . .                          | 142        |
| 9.4.3     | PAT - n la unu . . . . .                                | 142        |
| 9.5       | Cei patru termeni care încurcă pe toată lumea . . . . . | 143        |
| 9.6       | NAT pe echipamente Cisco . . . . .                      | 143        |
| 9.7       | NAT în Linux . . . . .                                  | 145        |
| 9.7.1     | Port forwarding . . . . .                               | 146        |
| 9.8       | Ce ne costă NAT . . . . .                               | 147        |
| 9.9       | Conceptul de tunelare . . . . .                         | 148        |
| 9.10      | Tunelul GRE . . . . .                                   | 149        |
| 9.11      | Tunelul SSH . . . . .                                   | 150        |
| 9.12      | IPv6, pe scurt . . . . .                                | 150        |
| 9.12.1    | Scrierea adreselor . . . . .                            | 151        |
| 9.13      | Tunelul 6to4 . . . . .                                  | 152        |
| 9.14      | Greșeli frecvente . . . . .                             | 153        |
| 9.15      | Rezumat și glosar . . . . .                             | 153        |
| 9.16      | Ce urmează . . . . .                                    | 154        |
| <b>10</b> | <b>Rutare dinamică</b>                                  | <b>157</b> |
| 10.1      | Din cursul de rutare . . . . .                          | 157        |
| 10.1.1    | Ce veți putea face după acest curs . . . . .            | 158        |
| 10.2      | De ce nu ne ajung rutele statice . . . . .              | 158        |
| 10.3      | Familiile de protocoale . . . . .                       | 158        |
| 10.4      | Distance-vector și problema buclelor . . . . .          | 160        |
| 10.4.1    | Numărătoarea spre infinit . . . . .                     | 160        |
| 10.5      | Cum lucrează un protocol link-state . . . . .           | 160        |
| 10.5.1    | Adiacențele și protocolul Hello . . . . .               | 161        |
| 10.5.2    | LSP: unitatea de informație . . . . .                   | 162        |
| 10.5.3    | Cei patru pași . . . . .                                | 162        |
| 10.6      | Algoritmul lui Dijkstra . . . . .                       | 163        |
| 10.7      | OSPF: caracteristici de bază . . . . .                  | 163        |
| 10.8      | Mesajele și stările adiacenței . . . . .                | 165        |
| 10.9      | DR și BDR pe rețele multiacces . . . . .                | 167        |
| 10.9.1    | Criteriile de alegere . . . . .                         | 167        |
| 10.10     | OSPF pe mai multe arii . . . . .                        | 168        |

|           |                                                   |            |
|-----------|---------------------------------------------------|------------|
| 10.10.1   | Tipurile de LSA . . . . .                         | 169        |
| 10.10.2   | Codurile din tabela de rutare . . . . .           | 169        |
| 10.11     | Configurare și verificare . . . . .               | 169        |
| 10.12     | Greșeli frecvente . . . . .                       | 171        |
| 10.13     | Rezumat și glosar . . . . .                       | 171        |
| 10.14     | Ce urmează . . . . .                              | 171        |
| <b>11</b> | <b>Rețele wireless</b>                            | <b>175</b> |
| 11.1      | Din cursul de nivel 2 . . . . .                   | 175        |
| 11.1.1    | Ce veți putea face după acest curs . . . . .      | 176        |
| 11.2      | Fizica din spatele deciziilor . . . . .           | 176        |
| 11.3      | Benzile libere și cine le mai folosește . . . . . | 177        |
| 11.4      | Evoluția standardelor 802.11 . . . . .            | 177        |
| 11.5      | Spreading și acces multiplu . . . . .             | 178        |
| 11.6      | Planificarea canalelor . . . . .                  | 180        |
| 11.7      | Organizarea unei rețele wireless . . . . .        | 182        |
| 11.8      | Accesul la mediu . . . . .                        | 182        |
| 11.9      | Cadrul 802.11 . . . . .                           | 183        |
| 11.10     | Două probleme care nu există pe cablu . . . . .   | 183        |
| 11.10.1   | Stația ascunsă . . . . .                          | 183        |
| 11.10.2   | Stația expusă . . . . .                           | 183        |
| 11.11     | Securitatea rețelelor wireless . . . . .          | 184        |
| 11.11.1   | Mecanismele care nu sunt securitate . . . . .     | 184        |
| 11.12     | Greșeli frecvente . . . . .                       | 186        |
| 11.13     | Rezumat și glosar . . . . .                       | 187        |
| 11.14     | Ce urmează . . . . .                              | 187        |
| <b>12</b> | <b>Securitatea LAN și atacuri de rețea</b>        | <b>191</b> |
| 12.1      | Din cursurile de comutare . . . . .               | 191        |
| 12.1.1    | Ce veți putea face după acest curs . . . . .      | 192        |
| 12.2      | Trei familii de atacuri . . . . .                 | 192        |
| 12.3      | Recunoașterea . . . . .                           | 193        |
| 12.4      | Atacuri de blocare a serviciului . . . . .        | 194        |
| 12.4.1    | Smurf attack . . . . .                            | 194        |
| 12.4.2    | TCP SYN flood . . . . .                           | 194        |
| 12.5      | CAM overflow . . . . .                            | 194        |
| 12.6      | Man-in-the-middle prin ARP poisoning . . . . .    | 196        |
| 12.7      | VLAN hopping . . . . .                            | 197        |
| 12.7.1    | Switch spoofing . . . . .                         | 197        |
| 12.7.2    | Double tagging . . . . .                          | 198        |
| 12.8      | Atacuri împotriva STP . . . . .                   | 198        |

|           |                                                                     |            |
|-----------|---------------------------------------------------------------------|------------|
| 12.9      | Parole, hash-uri și salt . . . . .                                  | 200        |
| 12.9.1    | Salt-ul . . . . .                                                   | 201        |
| 12.10     | Factorul uman și codul ostil . . . . .                              | 202        |
| 12.10.1   | Ingineria socială . . . . .                                         | 202        |
| 12.10.2   | Cod executabil ostil . . . . .                                      | 202        |
| 12.11     | Lista scurtă de securizare a unui switch . . . . .                  | 203        |
| 12.12     | Greșeli frecvente . . . . .                                         | 204        |
| 12.13     | Rezumat și glosar . . . . .                                         | 205        |
| 12.14     | Încheiere . . . . .                                                 | 205        |
| <b>13</b> | <b>Laborator 1: Primii pași: topologie, adresare și primul ping</b> | <b>209</b> |
| 13.1      | Dacă nu ați mai făcut asta niciodată . . . . .                      | 209        |
| 13.1.1    | Două moduri de a lucra . . . . .                                    | 209        |
| 13.2      | Echipamente necesare . . . . .                                      | 210        |
| 13.2.1    | Ce cablu, între ce echipamente . . . . .                            | 211        |
| 13.3      | Cinci cuvinte, o dată pentru totdeauna . . . . .                    | 211        |
| 13.4      | Prima rețea: două calculatoare . . . . .                            | 212        |
| 13.5      | Topologia completă . . . . .                                        | 212        |
| 13.5.1    | Planul de adresare . . . . .                                        | 212        |
| 13.6      | Linia de comandă, în trei minute . . . . .                          | 213        |
| 13.6.1    | Cum se citește promptul . . . . .                                   | 213        |
| 13.7      | Pași de lucru . . . . .                                             | 214        |
| 13.8      | Ce poartă un pachet, segment cu segment . . . . .                   | 216        |
| 13.8.1    | În bancul de lucru din pagină . . . . .                             | 216        |
| 13.8.2    | În Packet Tracer . . . . .                                          | 216        |
| 13.9      | Trei defecte de diagnosticat . . . . .                              | 217        |
| 13.10     | Sarcini de lucru . . . . .                                          | 218        |
| 13.11     | Provocare . . . . .                                                 | 219        |
| 13.12     | Ce se predă . . . . .                                               | 219        |
| <b>14</b> | <b>Laborator 2: Comutarea în LAN: tabela MAC, ARP, domenii</b>      | <b>221</b> |
| 14.1      | Ce demontăm astăzi . . . . .                                        | 221        |
| 14.2      | Echipamente necesare . . . . .                                      | 222        |
| 14.3      | Topologia de construit . . . . .                                    | 223        |
| 14.4      | Bancul de lucru . . . . .                                           | 223        |
| 14.5      | Tabela MAC, observată cum se umple . . . . .                        | 223        |
| 14.6      | Switch contra hub . . . . .                                         | 224        |
| 14.7      | ARP, cu cache-ul golit . . . . .                                    | 225        |
| 14.8      | Administrarea switch-ului . . . . .                                 | 225        |
| 14.8.1    | Accesul de la distanță . . . . .                                    | 226        |
| 14.9      | Port security . . . . .                                             | 227        |

|           |                                                                         |            |
|-----------|-------------------------------------------------------------------------|------------|
| 14.9.1    | Provocați violarea . . . . .                                            | 227        |
| 14.10     | Sarcini de lucru . . . . .                                              | 228        |
| 14.11     | Provocare . . . . .                                                     | 229        |
| 14.12     | Ce se predă . . . . .                                                   | 229        |
| <b>15</b> | <b>Laborator 3: VLAN-uri, trunk-uri și rutare inter-VLAN</b>            | <b>233</b> |
| 15.1      | Problema pe care o rezolvă VLAN-urile . . . . .                         | 233        |
| 15.2      | Echipamente necesare . . . . .                                          | 234        |
| 15.3      | Topologia și planul . . . . .                                           | 234        |
| 15.4      | Bancul de lucru . . . . .                                               | 236        |
| 15.5      | Pasul 1: crearea VLAN-urilor . . . . .                                  | 236        |
| 15.6      | Pasul 2: legăturile trunk . . . . .                                     | 237        |
| 15.7      | Pasul 3: router-on-a-stick . . . . .                                    | 238        |
| 15.8      | Varianta modernă: switch de nivel 3 . . . . .                           | 239        |
| 15.9      | Spanning Tree, observat . . . . .                                       | 240        |
| 15.10     | Patru defecte de diagnosticat . . . . .                                 | 241        |
| 15.11     | Sarcini de lucru . . . . .                                              | 241        |
| 15.12     | Provocare . . . . .                                                     | 242        |
| 15.13     | Ce se predă . . . . .                                                   | 242        |
| <b>16</b> | <b>Laborator 4: Proiectarea adreselor: VLSM și DHCP</b>                 | <b>245</b> |
| 16.1      | Ce înseamnă „a proiecta adresele” . . . . .                             | 245        |
| 16.2      | Echipamente necesare . . . . .                                          | 246        |
| 16.3      | Tema de proiectare . . . . .                                            | 247        |
| 16.4      | Cum se calculează, pas cu pas . . . . .                                 | 247        |
| 16.4.1    | Un exemplu complet, pentru primele două . . . . .                       | 248        |
| 16.4.2    | Verificarea fiecărei subrețele . . . . .                                | 249        |
| 16.5      | Topologia de implementat . . . . .                                      | 249        |
| 16.5.1    | Tabelul de adresare - se completează înainte de orice comandă . . . . . | 249        |
| 16.6      | Bancul de lucru . . . . .                                               | 250        |
| 16.7      | Implementarea planului . . . . .                                        | 250        |
| 16.8      | DHCP pe ruter . . . . .                                                 | 251        |
| 16.9      | DHCP relay . . . . .                                                    | 252        |
| 16.10     | Patru probleme de diagnosticat . . . . .                                | 252        |
| 16.11     | Sarcini de lucru . . . . .                                              | 252        |
| 16.12     | Provocare . . . . .                                                     | 253        |
| 16.13     | Ce se predă . . . . .                                                   | 254        |
| <b>17</b> | <b>Laborator 5: Rutare: statică, rute flotante și OSPF</b>              | <b>257</b> |
| 17.1      | Ce știe un ruter și de unde . . . . .                                   | 257        |
| 17.2      | Echipamente necesare . . . . .                                          | 258        |
| 17.3      | Topologia . . . . .                                                     | 259        |

|           |                                                              |            |
|-----------|--------------------------------------------------------------|------------|
| 17.4      | Bancul de lucru . . . . .                                    | 259        |
| 17.5      | Partea I: rutare statică . . . . .                           | 259        |
| 17.6      | Ruta implicită și ruta flotantă . . . . .                    | 261        |
| 17.7      | Sumarizare și Null0 . . . . .                                | 261        |
| 17.8      | Partea a II-a: OSPF . . . . .                                | 262        |
| 17.9      | Costul, care decide drumul . . . . .                         | 263        |
| 17.9.1    | Forțați o cale suboptimă . . . . .                           | 264        |
| 17.10     | Experimentul care justifică tot laboratorul . . . . .        | 264        |
| 17.11     | Sarcini de lucru . . . . .                                   | 265        |
| 17.12     | Provocare . . . . .                                          | 265        |
| 17.13     | Ce se predă . . . . .                                        | 266        |
| <b>18</b> | <b>Laborator 6: Liste de acces și NAT</b>                    | <b>269</b> |
| 18.1      | Cum se citește o listă de acces . . . . .                    | 269        |
| 18.2      | Echipamente necesare . . . . .                               | 270        |
| 18.3      | Topologia și politica . . . . .                              | 271        |
| 18.4      | Bancul de lucru . . . . .                                    | 271        |
| 18.5      | Pasul 0: rețeaua funcțională, fără filtre . . . . .          | 271        |
| 18.6      | Traducerea politicii în reguli . . . . .                     | 273        |
| 18.6.1    | P1 - Oaspeții . . . . .                                      | 273        |
| 18.6.2    | P2 - Biroul . . . . .                                        | 274        |
| 18.6.3    | P3 - DMZ-ul nu inițiază . . . . .                            | 274        |
| 18.6.4    | P4 - Accesul de administrare . . . . .                       | 275        |
| 18.6.5    | Verificați-vă înțelegerea . . . . .                          | 275        |
| 18.7      | Contorul, cel mai bun instrument de depanare . . . . .       | 275        |
| 18.8      | NAT: ieșirea în Internet . . . . .                           | 276        |
| 18.9      | Sarcini de lucru . . . . .                                   | 278        |
| 18.10     | Provocare . . . . .                                          | 278        |
| 18.11     | Ce se predă . . . . .                                        | 278        |
| <b>19</b> | <b>Laborator 7: Securitatea rețelei de acces și wireless</b> | <b>281</b> |
| 19.1      | Regula de aur a acestei lucrări . . . . .                    | 281        |
| 19.2      | Echipamente necesare . . . . .                               | 282        |
| 19.3      | Topologia . . . . .                                          | 283        |
| 19.4      | Bancul de lucru . . . . .                                    | 283        |
| 19.5      | Pasul 0: rețeaua care merge . . . . .                        | 283        |
| 19.6      | Atacul 1: switch-ul care devine hub . . . . .                | 285        |
| 19.7      | Atacul 2: serverul DHCP fals . . . . .                       | 286        |
| 19.8      | Atacul 3: ARP poisoning . . . . .                            | 286        |
| 19.9      | Închiderea celorlalte uși . . . . .                          | 287        |
| 19.9.1    | VLAN hopping . . . . .                                       | 287        |

---

|                                       |                                        |            |
|---------------------------------------|----------------------------------------|------------|
| 19.9.2                                | Atacuri asupra Spanning Tree . . . . . | 288        |
| 19.9.3                                | Administrarea switch-ului . . . . .    | 288        |
| 19.10                                 | Rețeaua wireless . . . . .             | 289        |
| 19.11                                 | Lista de verificare . . . . .          | 290        |
| 19.12                                 | Sarcini de lucru . . . . .             | 290        |
| 19.13                                 | Provocare . . . . .                    | 291        |
| 19.14                                 | Ce se predă . . . . .                  | 291        |
| <b>Listă de notații</b>               |                                        | <b>295</b> |
| <b>Anexă. Răspunsuri la întrebări</b> |                                        | <b>303</b> |





# 1. Transmisiuni analogice și digitale

---

---

Despre ce este vorba, de fapt  
Vocabularul minim: bit, octet, semnal  
Cât de mare e o rețea: LAN, MAN, WAN  
Echipamentele și nivelul la care lucrează  
Protocolul: de ce trebuie să vorbim aceeași limbă  
De ce comunicăm pe niveluri  
Nivelul fizic și cele patru sarcini ale lui  
Codări de linie: cum arată un bit pe fir  
Modulația: date digitale peste semnal analogic  
Greșeli frecvente  
Rezumat și glosar  
Ce urmează și unde citiți mai departe  
Întrebări de verificare

---

---

Acesta este primul curs și pornește de la zero: nu presupune că știți ce e un bit, ce e un protocol sau de ce un cablu de rețea are opt fire. Până la final veți înțelege ce se întâmplă, fizic, între momentul în care apăsați Enter și momentul în care mesajul ajunge la celălalt capăt - și veți avea vocabularul cu care se poartă toate discuțiile din restul semestrului.

## 1.1 Despre ce este vorba, de fapt

Aveți două calculatoare într-o cameră și vreți ca unul să îi trimită celuilalt un fișier. Pare simplu. Nu este. Ca să funcționeze, cineva a trebuit să răspundă, pe rând, la vreo cincisprezece întrebări:

- Ce fel de fir le leagă și cât de lung poate fi?
- Cum arată, în tensiune electrică, cifra 1? Dar cifra 0?
- De unde știe receptorul unde se termină un bit și începe următorul?
- Cum știe că mesajul a început? Cum știe că s-a terminat?
- Dacă amândoi vorbesc simultan, ce se întâmplă?
- Dacă un bit se strică pe drum, cum ne dăm seama?
- Dacă sunt zece calculatoare, nu două, cum spui cui îi este destinat mesajul?
- Dacă celălalt calculator e în altă clădire? În altă țară?

Fiecare dintre aceste întrebări are un răspuns, iar răspunsurile se numesc **protocoale**. Cursul acesta răspunde la primele cinci - cele fizice. Restul semestrului urcă, întrebare cu

întrebare, până la ultima.

**Dacă e prima dată când auziți asta.** O rețea de calculatoare este interconectarea mai multor sisteme de calcul astfel încât ele să poată schimba informație. Definiția e scurtă, dar ascunde o diferență importantă. În **interiorul** unui calculator, componentele comunică prin trasee de cupru pe placa de bază, lungi de câțiva centimetri, controlate de chipset. Nimeni nu se atinge de ele, nimeni altcineva nu le ascultă, iar semnalul ajunge întotdeauna intact. **Între** calculatoare nu mai avem niciuna dintre aceste garanții: distanța poate fi de metri sau de mii de kilometri, mediul poate fi zgomotos, iar celălalt capăt poate să aparțină altcuiva. Tot ce urmează în acest curs există ca să compenseze aceste trei lipsuri.

### 1.1.1 Ce veți putea face după acest curs

**Ce veți putea face după acest capitol.**

- Să explicați ce este un bit, un octet și de ce 100 Mbps nu înseamnă 100 MB pe secundă
- Să clasificați o rețea după acoperire: LAN, MAN, WAN
- Să spuneți, pentru orice echipament de rețea, la ce nivel lucrează și ce decizie ia
- Să explicați de ce comunicația e organizată pe niveluri și să treceți între modelele OSI și TCP/IP
- Să citiți și să desenați forma de undă pentru NRZ-L, NRZ-I, Manchester, MLT-3 și PAM-5
- Să calculați debitul binar pornind de la rata de simbol și schema de modulație

### 1.2 Vocabularul minim: bit, octet, semnal

Nu se poate discuta despre rețele fără patru cuvinte. Le luăm pe rând, fără să presupunem nimic.

**Bitul.** Un **bit** este cea mai mică unitate de informație posibilă: un răspuns la o întrebare cu două variante. Da sau nu. Aprins sau stins. 1 sau 0. Nu există nimic mai mic, pentru că o întrebare cu o singură variantă nu transmite nicio informație. Un bit singur e aproape inutil. Puterea vine din combinare: cu 2 biți exprimați 4 lucruri diferite, cu 3 biți 8, cu 8 biți 256. Regula este că **n biți exprimă  $2^n$  valori** - și de aici vine practic toată aritmetica din acest curs.

**Octetul și numerele.** Un **octet** (în engleză *byte*) este un grup de 8 biți. Poate reprezenta  $2^8 = 256$  de valori diferite, adică numerele de la 0 la 255. Nu e o alegere întâmplătoare: 255 e exact numărul maxim dintr-un octet al unei adrese IP, iar asta explică de ce nu există adresa 192.168.300.1. Același octet poate fi scris în trei feluri, care înseamnă exact același lucru: **zecimal** - cum numărăm noi: 141 **binar** - cum „gândește” echipamentul: 10001101 **hexazecimal** - o prescurtare comodă a binarului: 0x8D

Conversia binar  $\leftrightarrow$  zecimal nu este magie și nu se învață pe de rost. Fiecare poziție dintr-un octet are o **greutate** - de la dreapta la stânga: 1, 2, 4, 8, 16, 32, 64, 128. Valoarea zecimală este pur și simplu suma greutăților pozițiilor care conțin 1. Atât.

Jucați-vă cu el până când conversia devine reflex. Veți avea nevoie de ea la fiecare curs de aici înainte, iar la subnetare (cursul 5) va fi singura unealtă care contează. Apăsați

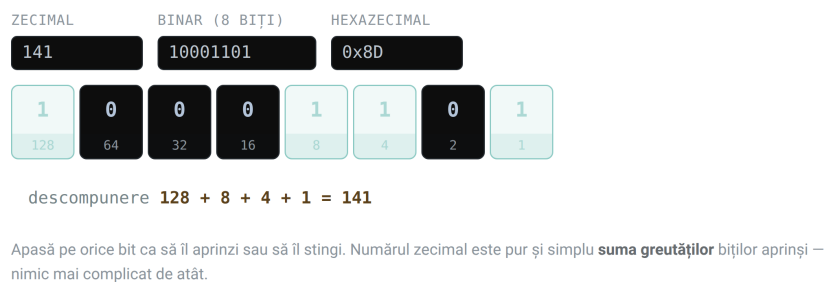


Figura 1.1 Convertor și antrenor de numerație.

„exersează” și încercați să ghiciți reprezentarea binară înainte de a se afișa.

**Hexazecimalul, în trei propoziții.** Baza 16 folosește cifrele 0–9 plus literele A–F pentru valorile 10–15. Motivul pentru care există este pur practic: **exact 4 biți încap într-o cifră hexazecimală**, deci un octet se scrie mereu cu două cifre hexa, iar conversia se face pe grupe de câte patru, fără niciun calcul.  $1000\ 1101 \rightarrow 8\ \text{și}\ D \rightarrow 0x8D$ . Din acest motiv adresele MAC și cele IPv6 se scriu hexazecimal: sunt mult mai scurte decât în binar și mult mai ușor de convertit decât zecimalul.

**Semnalul.** Un bit este o idee. Ca să ajungă de la un calculator la altul, ideea trebuie transformată în ceva fizic: o tensiune pe un fir, o pulsație de lumină într-o fibră, o undă radio în aer. Acest „ceva fizic” se numește **semnal**. Tot restul cursului se ocupă de o singură întrebare: **cum transformăm un șir de biți într-un semnal pe care celălalt capăt îl poate citi corect** - și înapoi.

### 1.2.1 Capcana unităților

Aici se produce cea mai frecventă neînțelegere dintre un administrator de rețea și un utilizator nemulțumit. Vitezele de rețea se măsoară în **biți**, iar dimensiunile fișierelor în **octeți**. Între ele e un factor de 8 - plus o a doua diferență, subtilă.

Tabela 1.1 Capcana unităților.

| Notăție    | Se citește               | Valoare                 | Unde apare            |
|------------|--------------------------|-------------------------|-----------------------|
| b          | bit                      | 1 bit                   | viteze de transmisie  |
| B          | octet (byte)             | 8 biți                  | dimensiuni de fișiere |
| kbps, Mbps | kilo/megabiți pe secundă | $10^3, 10^6$ biți/s     | lățime de bandă       |
| KB, MB     | kilo/megaocteți          | $2^{10}, 2^{20}$ octeți | capacitate de stocare |

#### Exemplu

Aveți o conexiune de 100 Mbps. În cât timp, în cel mai bun caz, descărcați un fișier de 500 MB?

**Rezolvare.**  $500\ \text{MB} = 500 \times 2^{20}\ \text{octeți} = 524\ 288\ 000\ \text{octeți} = 4\ 194\ 304\ 000\ \text{biți}$ .  
 $100\ \text{Mbps} = 10^8\ \text{biți pe secundă}$ .  $4\ 194\ 304\ 000 / 100\ 000\ 000 \approx 42\ \text{de secunde}$  - și asta

doar în cazul ideal, fără anteturi de protocol, fără retransmisii, fără nimeni altcineva pe legătură. În realitate, socotiți 15–20 % în plus. Observați: un utilizator care se aștepta la „500 MB pe 100 Mbps înseamnă 5 secunde” a greșit de opt ori - exact factorul dintre bit și octet.

### 1.3 Cât de mare e o rețea: LAN, MAN, WAN

Cea mai veche clasificare a rețelelor este după distanța dintre noduri. Nu e o simplă convenție de vocabular: fiecare clasă și-a dezvoltat protocoale proprii, potrivite ordinului ei de mărime, pentru că problemele sunt cu adevărat diferite la 10 metri față de 10 000 de kilometri.

Tabela 1.2 Cât de mare e o rețea: LAN, MAN, WAN.

| Distanța între noduri | Ce acoperă                                    | Tip de rețea | Protocoale dominante                                                   |
|-----------------------|-----------------------------------------------|--------------|------------------------------------------------------------------------|
| 1 mm – 1 cm           | pe siliciu, între procesoare                  | micro-rețele | magistrale interne                                                     |
| ~1 m                  | spațiul personal                              | PAN          | Bluetooth, Zigbee                                                      |
| 10 m                  | o cameră                                      | <b>LAN</b>   |                                                                        |
| Local Area Network    | Ethernet (IEEE 802.3),<br>Wi-Fi (IEEE 802.11) |              |                                                                        |
| 100 m                 | o clădire                                     | MAN          | rar întâlnite azi ca entitate distinctă<br>MPLS, PPP, Frame Relay, ATM |
| 1 km                  | un campus                                     |              |                                                                        |
| 10 km                 | un oraș                                       |              |                                                                        |
| 100 – 1000 km         | o țară, un continent                          | WAN          | stiva TCP/IP                                                           |
| ~10 000 km            | planeta                                       | Internet     |                                                                        |

Disciplina se numește **Rețele Locale**, deci accentul cade pe rândurile din mijloc - dar veți observa că ele nu pot fi înțelese fără ce se întâmplă la graniță.

**Unde se termină un LAN.** Granița dintre rețeaua locală și restul lumii este **ruterul**, numit și *gateway* - cuvânt care în engleză înseamnă exact „poartă”. Tot ce se află de partea internă a ruterului aparține rețelei locale; tot ce trece dincolo de el intră în altă rețea.

Această frontieră organizează întregul semestru: **cursurile 2–4** se ocupă de ce se întâmplă în interiorul ei, **cursurile 5–10** de ce se întâmplă la ea și dincolo de ea, iar **cursurile 8, 11 și 12** de cine încearcă să treacă prin ea fără să aibă voie.

### 1.4 Echipamentele și nivelul la care lucrează

Într-un magazin, echipamentele de rețea par să difere prin număr de porturi și preț. În realitate, diferența esențială este alta: **cât de adânc se uită fiecare în ceea ce trece prin el.**

1. Un **repetor** nu se uită deloc. Primește un semnal slăbit, îl amplifică și îl trimite mai

departe. Nu știe și nu îl interesează ce transportă.

2. Un **switch** se uită la adresa de nivel 2, cea scrisă în placa de rețea, și decide pe ce port să trimită cadrul.
3. Un **ruter** se uită la adresa de nivel 3, adresa IP, și decide spre ce rețea să trimită pachetul.
4. Un **firewall** se uită și mai adânc - până la conținutul propriu-zis - și decide dacă are voie să treacă.

Cu cât se uită mai adânc, cu atât ia decizii mai bune - și cu atât consumă mai mult timp. Acesta este compromisul care revine în tot restul cursului.

| ECHIPAMENT           | CE FACE ȘI LA CE NIVEL                                 |
|----------------------|--------------------------------------------------------|
| Placă de rețea (NIC) | nivelul 2 - comută cadre după adresa MAC               |
| Repetor / hub        | nivelul 3 - dirijează pachete după adresa IP           |
| Switch               | traduce între medii diferite: cupru ↔ fibră            |
| Ruter                | nivelurile 3-7 - decide ce are voie să treacă          |
| Firewall             | nivelul 1 - regenerează semnalul, fără să îl înțeleagă |
| Media convertor      | punctul de contact al unui calculator cu rețeaua       |

Apasă un element din stânga, apoi perechea lui din dreapta.

Figura 1.2 Potrivii fiecare echipament cu rolul lui.

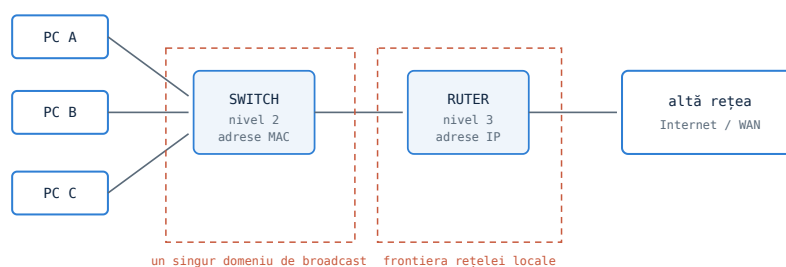


Figura 1.3 Switch-ul extinde rețeaua locală; ruterul o încheie. Toate stațiile din stânga ruterului se aud între ele fără intermediar; ca să vorbească cu ceva din dreapta, trebuie să ceară voie ruterului.

**Ce înseamnă „domeniu”.** Nu vă alarmați dacă ultimele două coloane nu vă spun încă nimic - sunt explicate pe larg în cursul 3. Deocamdată, două propoziții: Un **domeniu de coliziune** este grupul de dispozitive care se pot deranja reciproc dacă vorbesc în același timp. Un **domeniu de broadcast** este grupul care aude un anunț general dat de oricare dintre ele. Ambele sunt cu atât mai bune cu cât sunt mai mici.

Tabela 1.3 Echipamentele și nivelul la care lucrează.

| Echipament      | Nivel OSI            | Decizie luată pe baza       | Efect asupra domeniilor                 |
|-----------------|----------------------|-----------------------------|-----------------------------------------|
| Repetor, hub    | 1 - fizic            | nimic; regenerează semnalul | extinde și coliziunea, și broadcast-ul  |
| Switch (bridge) | 2 - legătură de date | adresa MAC destinație       | separă coliziunea, extinde broadcast-ul |
| Ruter           | 3 - rețea            | adresa IP destinație        | separă și coliziunea, și broadcast-ul   |
| Firewall / IPS  | 3-7                  | anteturi și conținut        | filtrează, nu doar dirijează            |

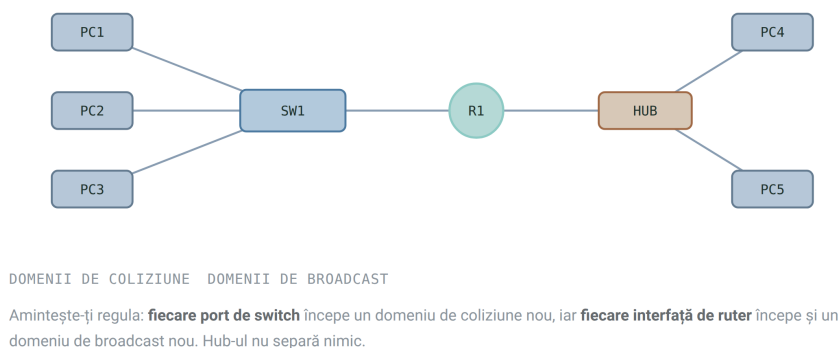


Figura 1.4 Exercițiu: câte domenii vedeți în topologia asta?.

### 1.4.1 Interfața de rețea

Termenul **interfață** desemnează un punct de comunicație cu o rețea: o placă de rețea a unui calculator, un port al unui switch, un port al unui ruter. Un calculator cu o singură placă are o singură interfață; unul cu două plăci are două. Un ruter are mai multe și tocmai de aceea poate lega rețele diferite - practic, aceasta este întreaga lui rațiune de a exista.

În sistemele de operare, interfața este și o abstracție software. Pe Linux, plăcile Ethernet apar ca `eth0`, `eth1`, iar o interfață virtuală numită **loopback** (`lo`, adresa `127.0.0.1`) permite unui calculator să își adreseze propria stivă de protocoale ca și cum ar fi în rețea. Este primul lucru pe care îl testați când „nu merge nimic”: dacă nici loopback-ul nu răspunde, problema nu e în rețea, e în calculator.

## 1.5 Protocolul: de ce trebuie să vorbim aceeași limbă

### Observație

**Protocol.** Un set de reguli care guvernează modul în care două entități schimbă informație. Ambele capete ale unei transmisii trebuie să folosească **același** protocol; altfel semnalul ajunge, dar mesajul nu.

### Analogie

Doi oameni se întâlnesc. Unul spune „zao shang hao”. Celălalt aude perfect fiecare sunet - semnalul a ajuns intact - dar nu înțelege nimic, pentru că nu împart aceeași convenție despre ce înseamnă sunetele. Aceasta e diferența dintre

**a auzi**

și

**a înțelege**

, iar în rețele e diferența dintre nivelul fizic și tot ce vine deasupra lui.

Un protocol de rețea stabilește, tipic, patru lucruri:

1. **Sintaxa** - cum arată mesajul: ce câmpuri are, în ce ordine, pe câți biți fiecare.
2. **Semantica** - ce înseamnă fiecare câmp și ce trebuie făcut cu el.
3. **Sincronizarea** - cine vorbește când, cât așteaptă, ce face dacă nu primește răspuns.
4. **Tratarea erorilor** - ce se întâmplă când ceva se strică.

Veți vedea aceste patru componente la fiecare protocol studiat în semestru. Când un protocol nou vi se pare complicat, întrebați-vă care dintre cele patru vă scapă - de obicei e a treia.

## 1.6 De ce comunicăm pe niveluri

Să presupunem că scriem singuri un program de e-mail, fără să existe nimic înainte. Ar trebui să ne ocupăm de tensiunea de pe fir, de retransmiterea pachetelor pierdute, de ordinea lor, de formatul mesajului și de multe altele - toate în același cod. Ar fi imposibil de întreținut. Și, mai rău, ar trebui rescris de la zero de fiecare dată când apare un tip nou de cablu.

Soluția, veche de peste patruzeci de ani, este **stratificarea**: împărțim problema în niveluri, iar fiecare nivel oferă un serviciu nivelului de deasupra și folosește serviciile nivelului de dedesubt, fără să știe cum sunt ele implementate.

### Analogie

Trimiteți un colet. Îl împachetați și scrieți adresa - acesta e nivelul vostru. Îl duceți la poștă, care îl pune într-un sac cu alte colete pentru același oraș - alt nivel. Sacul urcă într-un camion - alt nivel. Șoferul nu știe și nu îl interesează ce e în colet; voi nu știți și nu vă interesează pe ce autostradă merge camionul. Fiecare nivel are propriul lui „ambalaj” și propria lui adresă, iar la destinație totul se despachetează în ordine inversă.

**Ce câștigăm.** Un nivel poate fi înlocuit fără să afecteze restul: trecerea de la cupru la fibră nu schimbă nimic la aplicație. Problemele se localizează: „nu merge” devine „nu merge la nivelul 3”. Producători diferiți pot implementa niveluri diferite. Se poate învăța pe rând, ceea ce face exact acest curs.

### Ce pierdem

- Fiecare nivel adaugă antet - deci trafic care nu e informație utilă



- Informația utilă unui nivel nu îi este accesibilă altuia; NAT, din cursul 9, este considerat o încălcare tocmai din acest motiv
- Trecerea prin niveluri costă timp de procesare

### 1.6.1 Modelul OSI: șapte niveluri

Apăsați pe fiecare nivel din stânga. Nu încercați să memorați tabelul - reveniți la el ori de câte ori aveți nevoie. Ce trebuie să rămână este ideea: **fiecare nivel are propriul tip de adresă și propria unitate de date.**

7. Aplicație → Aplicație

6. Prezentare → Aplicație

5. Sesiune → Aplicație

4. Transport → Transport

3. Rețea → Internet

2. Legătură de date → Acces la mediu

1. Fizic → Acces la mediu

#### 7. Aplicație

Oferă servicii direct programului folosit de om: transfer de fișiere, poștă electronică, navigare, control de la distanță. Aici trăiesc protocoalele pe care le recunoașteți după nume.

|                 |                            |
|-----------------|----------------------------|
| UNITATE DE DATE | date                       |
| ADRESA FOLOSITĂ | -                          |
| ECHIPAMENTE     | servere, aplicații, proxy  |
| EXEMPLE         | HTTP, DNS, SSH, SMTP, IMAP |
| ÎN TCP/IP       | Aplicație                  |

**DE REȚINUT**

Când browserul afișează o pagină, aici se termină drumul.

Apasă pe orice nivel ca să vezi ce rezolvă, cu ce adresă lucrează și ce echipamente trăiesc acolo.

Figura 1.5 Explorați stiva OSI, nivel cu nivel.

### 1.6.2 Stiva TCP/IP: patru niveluri, dar reale

Modelul OSI este un model de **referință**: îl folosim ca vocabular comun. Stiva TCP/IP este ce rulează efectiv în Internet. Ele nu se contrazic - TCP/IP comasează nivelurile pe care OSI le separă, pentru că în practică nimeni nu implementează separat sesiunea și prezentarea.

### 1.6.3 Încapsularea: cum coboară un mesaj prin stivă

Aceasta este cea mai importantă figură din tot cursul. Urmăriți-o pas cu pas.

**De reținut, chiar dacă uitați restul.** Nivelul 2 lucrează cu **adrese MAC** și **cadre**. Nivelul 3 cu **adrese IP** și **pachete**. Nivelul 4 cu **porturi** și **segmente**.

Când depanați o rețea, prima întrebare utilă este întotdeauna: **la ce nivel se rupe?** Restul semestrului vă dă uneltele cu care răspundeți la ea.

Tabela 1.4 Stiva TCP/IP: patru niveluri, dar reale.

| OSI                                         | TCP/IP         | Ce rezolvă                                                              | Exemple               |
|---------------------------------------------|----------------|-------------------------------------------------------------------------|-----------------------|
| 7. Aplicație<br>6. Prezentare<br>5. Sesiune | Aplicație      | servicii pentru utilizator, reprezentarea datelor, controlul dialogului | HTTP, DNS, SSH, SMTP  |
| 4. Transport                                | Transport      | controlul fluxului, siguranța transmisiei, multiplexarea pe porturi     | TCP, UDP              |
| 3. Rețea                                    | Internet       | adresare globală și alegerea căii spre destinație                       | IPv4, IPv6, ICMP      |
| 2. Legătură de date<br>1. Fizic             | Acces la mediu | accesul la mediul partajat, încadrarea, transmisia binară               | Ethernet, 802.11, PPP |

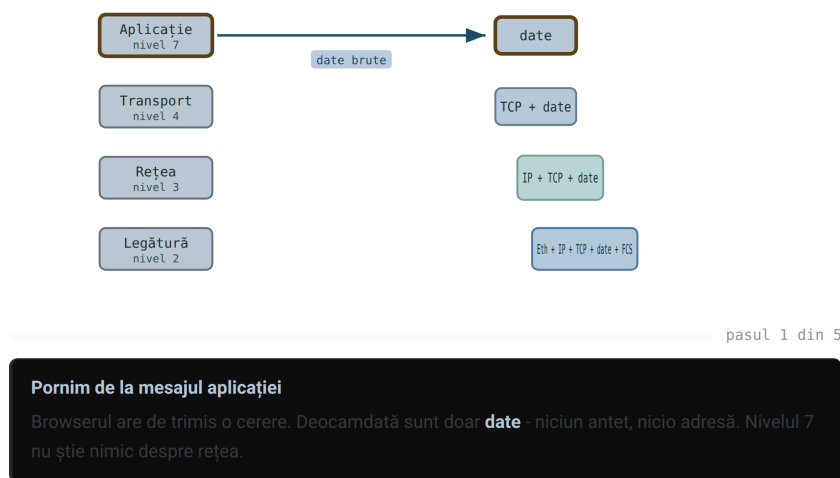


Figura 1.6 Încapsularea, pas cu pas.

## 1.7 Nivelul fizic și cele patru sarcini ale lui

Nivelul fizic are o misiune enunțabilă într-o propoziție: **să transforme un bit într-un semnal și înapoi**. În practică, misiunea aceasta se desface în patru sarcini distincte.

1. **Transformarea bit → semnal.** Alegerea nivelurilor de tensiune, a lungimii de undă sau a frecvenței care reprezintă valorile logice.
2. **Sincronizarea biților.** Receptorul trebuie să știe unde se termină un bit și unde începe următorul. Pare un detaliu; este, de fapt, motivul pentru care există jumătate din codările pe care le vom vedea imediat.

3. **Controlul vitezei.** Negocierea ratei de transmisie pe care ambele capete o pot susține - motivul pentru care o placă gigabit și un switch de 100 Mbps ajung, totuși, la o înțelegere.
4. **Multiplexarea.** Punerea mai multor fluxuri logice pe același mediu fizic. Subiect tratat pe larg în cursul următor.

**De ce sincronizarea e o problemă reală.** Imaginați-vă că vă dictez o serie de cifre, dar fără nicio pauză între ele și fără să vă spun cât durează una. Dacă vă dictez zece de „zero” la rând, cu voce constantă, câte ați notat? Nu aveți cum să știți - poate erau opt, poate douăsprezece. Exact aceasta e problema receptorului: dacă tensiunea rămâne constantă mult timp, el nu are cum să numere biții. Soluția e fie un ceas separat (scump: încă un fir), fie o codare care **garantează schimbări dese ale semnalului**, din care ceasul se poate reconstitui. Aproape toate codările moderne aleg a doua variantă.

### 1.7.1 Patru combinații de date și semnal

Cuvintele „digital” și „analogic” se aplică **separat** datelor și semnalului. Rezultă patru situații posibile - toate cu utilizare reală.

Tabela 1.5 Patru combinații de date și semnal.

| Datele sunt | Semnalul este | Operația numește          | Exemplu din viața reală               |
|-------------|---------------|---------------------------|---------------------------------------|
| digitale    | digital       | <b>codare de linie</b>    | Ethernet pe cablu torsadat            |
| digitale    | analogic      | <b>modulație</b>          | Wi-Fi, DSL, modem pe linie telefonică |
| analogice   | digital       | eșantionare și cuantizare | vocea digitalizată, VoIP              |
| analogice   | analogic      | modulație analogică       | radio AM și FM, telefonie clasică     |

**De unde vine cuvântul „modem”.** Un modem este un **MO**dulator/**DE**Modulator: transformă datele digitale ale calculatorului în semnal analogic potrivit mediului și face operația inversă la recepție. Cuvântul a supraviețuit tehnologiei care l-a produs - cutia de la furnizorul de Internet tot modem se numește, deși înăuntru nu mai seamănă deloc cu strămoșul ei de pe linia telefonică.

## 1.8 Codări de linie: cum arată un bit pe fir

O **codare de linie** este regula prin care un șir de biți devine o formă de undă. Este cea mai concretă parte a acestui curs: aici se vede, la propriu, cum arată informația pe cablu.

O codare bună rezolvă simultan trei probleme:

1. **Distincția.** Receptorul trebuie să poată deosebi fără ambiguitate un 1 de un 0, chiar și cu puțin zgomot pe fir.

2. **Sincronizarea.** Receptorul trebuie să își poată recupera ceasul din chiar semnalul primit.
3. **Absența componentei continue.** Media semnalului trebuie să fie zero, pentru că transformatoarele de izolare din plăcile de rețea nu transmit componenta continuă. Un semnal care stă mult timp „sus” pur și simplu nu trece prin ele.

Codările se clasifică după numărul de niveluri de tensiune folosite: **unipolare** (un nivel plus absența semnalului), **polare** (două niveluri, pozitiv și negativ) și **bipolare** sau multi-nivel (trei sau mai multe).

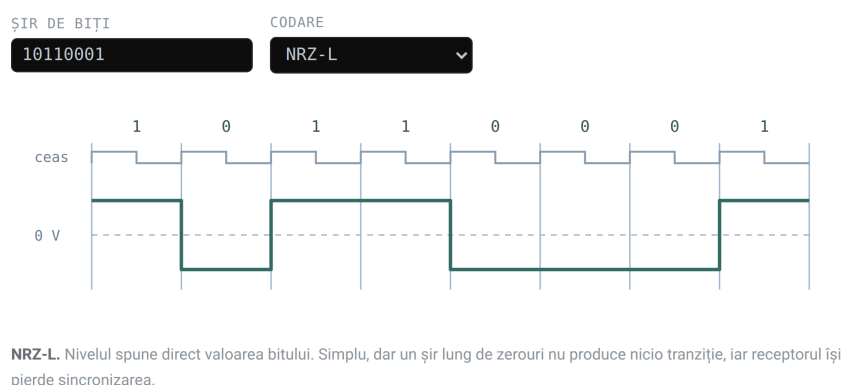


Figura 1.7 Generator de forme de undă - încercați fiecare codare.

### 1.8.1 Ce trebuie să observați, pe rând

Nu treceți mai departe fără să faceți efectiv experimentele de mai jos. Sunt cinci minute și înlocuiesc o oră de citit.

#### Exemplu

Cu NRZ, un șir alternat 101010... la 100 Mbps produce o schimbare de nivel la fiecare bit, adică o frecvență fundamentală de 50 MHz. Cablul Cat 5 este certificat până la 100 MHz, dar la 50 MHz emite deja destulă radiație cât să deranjeze perechile vecine și echipamentele din jur.

MLT-3 avansează în ciclul de patru stări doar la biții de 1, deci îi trebuie **patru** biți ca să revină la punctul de plecare. Frecvența fundamentală scade la 25 MHz. Aceasta este, esențial, ideea care a permis ca 100 Mbps să circule pe cablul telefonic ieftin deja tras prin clădiri - și de aici a plecat tot restul.

### 1.8.2 Codarea în bloc: 4B/5B

Problema șirurilor lungi de zerouri se poate rezolva elegant nu la nivelul semnalului, ci **înaintea** lui: rescriem datele astfel încât să nu mai conțină secvențe problematice.

Codarea **4B/5B** înlocuiește fiecare grup de 4 biți de date cu un grup de 5 biți ales dintr-un tabel construit special: niciun cuvânt de cod nu are mai mult de trei zerouri consecutive. Receptorul face traducerea inversă și recuperează datele originale.

Tabela 1.6 Ce trebuie să observați, pe rând.

| Codare                                   | Regula                                                                            | Experiment de făcut             | Ce veți observa                                                                       |
|------------------------------------------|-----------------------------------------------------------------------------------|---------------------------------|---------------------------------------------------------------------------------------|
| <b>NRZ-L</b><br>Non-Return to Zero Level | nivelul <i>este</i> valoarea bitului                                              | scrieți 00000000                | o linie perfect dreaptă - receptorul nu mai are ce număra                             |
| <b>NRZ-I</b><br>NRZ Inverted             | 1 = schimbă nivelul, 0 = păstrează-l                                              | scrieți 11111111, apoi 00000000 | unii produc tranziții la fiecare bit, zerourile tot niciuna                           |
| <b>Manchester</b>                        | tranziție obligatorie la mijlocul fiecărui bit                                    | orice șir                       | semnalul se schimbă de două ori mai des decât rata de bit                             |
| <b>Manchester diferențial</b>            | tranziția de la mijloc rămâne; informația e în prezența unei tranziții la început | comparați cu Manchester simplu  | rezistă la inversarea firelor                                                         |
| <b>MLT-3</b>                             | trei niveluri parcurse ciclic, avansează doar la 1                                | scrieți 11111111                | ciclul $0 \rightarrow +1 \rightarrow 0 \rightarrow -1$ ; frecvența scade de patru ori |
| <b>PAM-5</b>                             | doi biți per simbol, pe niveluri de amplitudine                                   | orice șir par                   | jumătate din numărul de tranziții pentru aceiași biți                                 |

Tabela 1.7 Codarea în bloc: 4B/5B.

| date (4b) | cod (5b) | date (4b) | cod (5b) | simbol de control   | cod (5b) |
|-----------|----------|-----------|----------|---------------------|----------|
| 0000      | 11110    | 1000      | 10010    | Q - linie liniștită | 00000    |
| 0001      | 01001    | 1001      | 10011    | I - inactiv         | 11111    |
| 0010      | 10100    | 1010      | 10110    | J - început flux    | 11000    |
| 0011      | 10101    | 1011      | 10111    | K - început flux    | 10001    |
| 0100      | 01010    | 1100      | 11010    | T - sfârșit flux    | 01101    |
| 0101      | 01011    | 1101      | 11011    | R - resetare        | 00111    |
| 0110      | 01110    | 1110      | 11100    | S - set             | 11001    |
| 0111      | 01111    | 1111      | 11101    | H - eroare          | 00100    |

**Observație**

**Costul redundanței, și partea bună a lui.** 4B/5B introduce 25 % supraîncărcare: ca să transmiteți 100 Mbps de date utile, linia trebuie să ducă **125 Mbaud**. Pare risipă. Dar din cele 32 de cuvinte posibile de 5 biți, doar 16 sunt folosite pentru date. Celelalte 16 nu se irosesc - devin **simboluri de control** care marchează începutul cadrului, sfârșitul lui sau starea de repaus a liniei. Cu alte cuvinte, redundanța cumpărată pentru sincronizare rezolvă gratuit și problema delimitării mesajelor.

**1.8.3 Cum se combină în tehnologiile reale**

Niciun standard nu folosește o singură tehnică. Se combină în lanț, fiecare rezolvând altă problemă:

Tabela 1.8 Cum se combină în tehnologiile reale.

| Standard         | Mediu          | Lanțul de codare                    | De ce așa                                             |
|------------------|----------------|-------------------------------------|-------------------------------------------------------|
| 100BASE-TX       | cupru, Cat 5   | date → 4B/5B → MLT-3                | 4B/5B dă sincronizarea, MLT-3 coboară frecvența       |
| 100BASE-FX       | fibră multimod | date → 4B/5B → NRZ-I                | pe fibră frecvența nu e o problemă, deci NRZ-I ajunge |
| 1000BASE-SX / LX | fibră          | date → 8B/10B → NRZ                 | 8B/10B echilibrează și componenta continuă            |
| 1000BASE-T       | cupru, Cat 5e  | date → PAM-5, pe 4 perechi simultan | 250 Mbps pe fiecare pereche, în ambele sensuri        |

**1.9 Modulația: date digitale peste semnal analogic**

Dacă mediul nu suportă tranziții bruște de tensiune - o linie telefonică, aerul, un canal radio - biții nu pot fi transmiși direct. Trebuie transportați modificând un semnal continuu, numit **purtătoare**.

**Ce e o purtătoare.** Imaginați-vă un fluierat constant, la aceeași înălțime și aceeași intensitate. În sine nu transmite nimic - dar dacă îl faceți mai tare și mai încet după o convenție dinainte stabilită, ați transmis informație **peste** fluierat. Fluieratul este purtătoarea; modificarea lui este modulația.

O undă sinusoidală are exact trei parametri pe care îi putem modifica - și, prin urmare, există exact trei modulații fundamentale.

Tabela 1.9 Modulația: date digitale peste semnal analogic.

| Parametrul modificat             | Denumire                            | Cum arată                                         | Unde apare                       |
|----------------------------------|-------------------------------------|---------------------------------------------------|----------------------------------|
| amplitudinea (înălțimea unde)    | <b>ASK</b> - Amplitude Shift Keying | 1 = undă puternică,<br>0 = undă slabă sau absentă | fibră optică, RFID               |
| frecvența (cât de des oscilează) | <b>FSK</b> - Frequency Shift Keying | 1 și 0 sunt două frecvențe distincte              | modemuri vechi, Bluetooth clasic |
| faza (decalajul unde)            | <b>PSK</b> - Phase Shift Keying     | 1 și 0 diferă prin poziția în ciclu               | Wi-Fi, satelit, DVB              |

### 1.9.1 Rata de bit nu este rata de simbol

#### Observație

**Bit rate și baud rate.** **Rata de bit** (bps) este numărul de biți transmiși într-o secundă. **Rata de simbol** (baud) este numărul de schimbări de stare ale semnalului într-o secundă. Întotdeauna  $\text{baud} \leq \text{bps}$ , iar raportul dintre ele este exact numărul de biți codificați într-un simbol.

Distincția asta este cheia întregii evoluții a comunicațiilor. Un canal are o limită fizică pentru rata de simbol, impusă de lățimea lui de bandă - nu o puteți depăși oricât ați plăti. Singura cale de a transmite mai mulți biți pe secundă este să **codificați mai mulți biți în fiecare simbol**.

Aici intervin schemele combinate. **QAM** (Quadrature Amplitude Modulation) variază simultan amplitudinea și faza, obținând mai multe puncte distincte în plan - fiecare punct fiind un simbol.

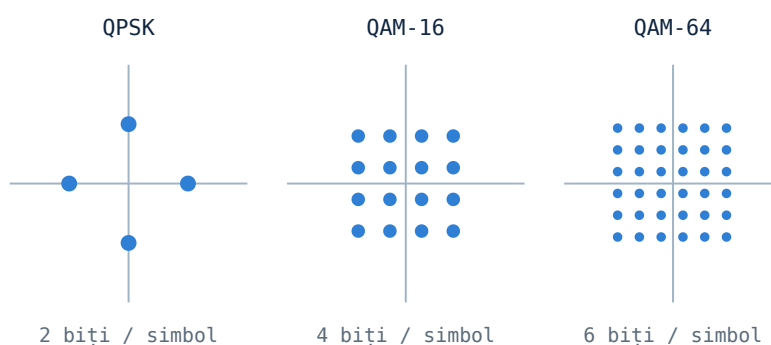


Figura 1.8 Diagrame de constelații. Fiecare punct este o combinație distinctă de amplitudine și fază, deci un simbol. Cu cât punctele sunt mai multe și mai apropiate, cu atât transmitem mai mulți biți per simbol - și cu atât mai puțin zgomot este nevoie ca receptorul să confunde două puncte vecine.

**Compromisul care se repetă peste tot.** Mai multe puncte de constelație = mai mulți biți per simbol = debit mai mare. Dar punctele devin mai apropiate, deci e nevoie de

un semnal mai curat pentru a le distinge.

De aceea, când vă îndepărtați de routerul Wi-Fi, viteza scade: echipamentul **coboară automat** pe o modulație cu mai puține puncte, mai robustă. Nu s-a stricat nimic - se negociază, în timp real, între debit și fiabilitate.

| RATA DE SIMBOL | UNITATE | MODULAȚIE | FLUXURI MIMO |
|----------------|---------|-----------|--------------|
| 2400           | baud    | QAM-16    | 1            |

puncte de constelație **16**  
 biți per simbol  $\log_2(16) = 4$   
 calcul  $2.400 \text{ simb/s} \times 4 \text{ b/simb}$   
 debit brut **9.600 kbps**

Un canal are o limită fizică pentru numărul de simboluri pe secundă. Singura cale de a transmite mai mulți biți este să pui mai multă informație în fiecare simbol — asta face modulația.

Figura 1.9 Calculator: de la rata de simbol la debit.

### Exemplu

Se consideră o linie cu o capacitate de 2400 baud. Câți biți de date pot fi trimiși pe secundă dacă se folosește QAM-16 pentru modulare?

**Rezolvare.** QAM-16 are 16 puncte de constelație. Numărul de biți codificați într-un simbol este  $\log_2 16 = 4$ .  $4 \text{ biți/simbol} \times 2400 \text{ simboluri/s} = 9600 \text{ bps}$  Aceeași linie, cu QAM-64, ar duce  $6 \times 2400 = 14\,400 \text{ bps}$  - fără nicio modificare fizică, doar schimbând schema de modulație. Verificați ambele în calculatorul de mai sus.

### Exemplu

Un canal Wi-Fi transmite 1000 de simboluri pe microsecundă folosind QAM-256, pe un singur flux spațial. Ce debit brut rezultă? Dar dacă se folosesc patru fluxuri MIMO simultan?

**Rezolvare.**  $\log_2 256 = 8 \text{ biți/simbol}$ .  $1000 \text{ simboluri}/\mu\text{s} = 10^9 \text{ simboluri/s}$ . Rezultă **8 Gbps** brut pe flux; cu patru fluxuri, 32 Gbps. Debitul **util** este semnificativ mai mic - codarea de corecție a erorilor, anteturile și intervalele de gardă consumă o parte importantă. Este observația care revine la fiecare tehnologie din acest curs: cifra de pe cutie nu este cifra pe care o vedeți.

## 1.10 Greșeli frecvente

### Observație

#### Greșeli frecvente.

- „100 Mbps înseamnă 100 MB pe secundă” Factor de 8 între bit și octet, plus diferența dintre puterile lui 10 și ale lui 2. O legătură de 100 Mbps transferă, în cel mai bun caz, aproximativ 11,9 MB pe secundă. Verificați întotdeauna dacă litera este mică (bit) sau mare (octet).
- „Un switch e un hub mai bun” Nu sunt din aceeași familie. Hub-ul repetă



orbește semnalul pe toate porturile; switch-ul citește adresa și trimite doar unde trebuie. Diferența e de nivel, nu de calitate. Rețineți nivelul: hub = 1, switch = 2, ruter = 3.

- „**OSI și TCP/IP sunt două lucruri concurente**” Nu sunt. OSI e un model de referință, folosit ca vocabular; TCP/IP e ce rulează efectiv. Se suprapun, nu se contrazic. Când cineva spune „nivelul 3”, vorbește în OSI, chiar dacă rețeaua rulează TCP/IP.
- „**Modulația și codarea de linie sunt același lucru**” Codarea de linie produce un semnal *digital* (trepte de tensiune). Modulația modifică un semnal *analogic* continuu. Întrebați-vă ce fel de semnal iese: trepte → codare; undă → modulație.
- „**Cu cât modulația e mai densă, cu atât e mai bine**” Doar dacă semnalul e suficient de curat. QAM-1024 pe o legătură zgomotoasă produce mai multe erori și retransmisii decât QPSK - deci, în final, un debit mai mic. Debitul real depinde de raportul semnal-zgomot, nu doar de schema aleasă.

### 1.11 Rezumat și glosar

Cursul acesta a construit vocabularul și a coborât până la fir. Trei idei rezistă:

1. Comunicația e organizată pe **niveluri**, fiecare cu adresa și unitatea lui de date. Când ceva nu merge, prima întrebare este la ce nivel.
2. Nivelul fizic transformă biți în semnal, iar problema lui grea nu e distincția între 1 și 0, ci **sincronizarea**.
3. Debitul se obține fie transmitând mai multe simboluri pe secundă - limitat fizic - fie punând mai mulți biți în fiecare simbol. A doua cale a produs toată evoluția din ultimii treizeci de ani.

1 Transportul adaugă antetul TCP, cu porturile

2 Aplicația produce datele

3 Nivelul fizic transformă cadrul în semnal pe fir

4 Rețeaua adaugă antetul IP, cu adresele

5 Legătura de date adaugă antetul Ethernet și FCS-ul

Mută rândurile cu săgeți până ajung în ordinea corectă.

Figura 1.10 Puneți în ordine pașii încapsulării.

## 1.12 Ce urmează și unde citiți mai departe

Cursul următor rămâne la nivelul fizic, dar schimbă întrebarea: nu *cum* arată semnalul, ci *prin ce* îl trimitem. Cupru, fibră sau aer - ce poate duce fiecare, cât de departe, ce se strică pe drum și cum punem mai multe conversații pe un singur fir.

La **laboratorul 1** construiți prima topologie în Packet Tracer și urmăriți un pachet real de la un capăt la altul, cu toate anteturile pe care le-ați văzut aici.

- IEEE 802.3 - standardul Ethernet, capitolele despre nivelul fizic (PCS, PMA)
- Andrew S. Tanenbaum, *Computer Networks*, capitolul 2 - nivelul fizic
- James Kurose, Keith Ross, *Computer Networking: A Top-Down Approach*, capitolul 1
- RFC 1122 - cerințele stivei TCP/IP pentru stații
- Pentru cine vrea partea matematică: teorema lui Nyquist și formula lui Shannon pentru capacitatea unui canal cu zgomot

### Întrebări de verificare

1. Care este unitatea de date de la nivelul 3?
  - a) Cadru
  - b) Pachet
  - c) Segment
  - d) Bit
2. De ce este util ca stiva sa fie impartita pe niveluri?
  - a) Pentru ca merge mai repede
  - b) Pentru ca un nivel poate fi inlocuit fara sa afecteze restul
  - c) Pentru ca reduce dimensiunea pachetelor
  - d) Pentru ca elimina erorile
3. De ce codarea Manchester nu isi pierde niciodata sincronizarea?
  - a) Pentru ca foloseste trei niveluri de tensiune
  - b) Pentru ca fiecare bit contine obligatoriu o tranzitie la mijloc
  - c) Pentru ca transmite ceasul pe un fir separat
  - d) Pentru ca inlocuieste grupurile de 4 biti cu grupuri de 5
4. Ce rezolva codarea 4B/5B in lantul 100BASE-TX?
  - a) Cripoteaza datele pe fir
  - b) Elimina secventele lungi de zerouri, care ar face receptorul sa piarda ceasul
  - c) Reduce numarul de fire de la patru la doua

- d) Corectează erorile de transmisie
5. Câte valori distincte pot fi reprezentate pe 6 biti?
- a) 12
  - b) 36
  - c) 64
  - d) 128
6. Care echipament separă domeniile de coliziune, dar extinde domeniile de broadcast?
- a) Hub-ul
  - b) Switch-ul
  - c) Routerul
  - d) Repetitorul
7. O linie de 4800 baud folosește QPSK. Ce debit binar obține?
- a) 4800 bps
  - b) 9600 bps
  - c) 19200 bps
  - d) 2400 bps
8. În stiva TCP/IP, cui nivel OSI îi corespunde nivelul Internet?
- a) Nivelului 2, legătură de date
  - b) Nivelului 3, rețea
  - c) Nivelului 4, transport
  - d) Nivelurilor 5-7
9. Ce adresă folosește nivelul transport ca să livreze datele programului potrivit?
- a) Adresa MAC
  - b) Adresa IP
  - c) Numărul de port
  - d) Numele de domeniu
10. Ce înseamnă că un semnal nu are componentă continuă?
- a) Că nu se transmite niciodată
  - b) Că media lui în timp este zero, deci poate trece prin transformatoarele de izolare

- c) Ca frecvența este constantă
  - d) Ca folosește doar două niveluri
11. Un octet are valoarea binară 11000000. Cât este în zecimal?
- a) 96
  - b) 128
  - c) 192
  - d) 224
12. De ce se folosește hexazecimalul pentru adresele MAC și IPv6?
- a) Pentru că ocupă mai puțină memorie
  - b) Pentru că o cifră hexa reprezintă exact 4 biți, deci conversia este imediată
  - c) Pentru că este mai sigur
  - d) Pentru că zecimalul nu suportă valori mari



## 2. Medii de transmisie

---

---

Din cursul trecut  
Trei familii de medii  
Cablul coaxial  
Cablul torsadat: de ce se răsucesc firele  
Cele trei mufări clasice  
Fibra optică  
Mediul fără fir  
Multiplexarea: mai multe conversații pe un mediu  
Performanța unei legături  
Ce strică semnalul pe drum  
Greșeli frecvente  
Rezumat și glosar  
Ce urmează și unde citiți mai departe  
Întrebări de verificare

---

---

Protocolul poate fi elegant și topologia perfect proiectată, dar în cele din urmă un fir, o fibră sau o bucată de aer duce biții. Cursul acesta pune față în față cele trei familii de medii, explică de la zero de ce se răsucesc firele într-un cablu de rețea, ce se strică pe drum și cum reușim să punem zeci de conversații pe același mediu fizic.

### 2.1 Din cursul trecut

**Ce trebuie să aveți în minte.** Un **bit** devine un **semnal** - tensiune, lumină sau undă radio - prin **codare de linie** (semnal digital) sau prin **modulație** (semnal analogic). Comunicația e organizată pe **niveluri**. Cursul acesta rămâne la **nivelul 1, fizic**. Problema grea a nivelului fizic nu e distincția dintre 1 și 0, ci **sincronizarea**: receptorul trebuie să știe unde se termină un bit. **Rata de simbol** (baud) este limitată fizic de mediu. Cursul acesta arată de *ce* este limitată.

#### 2.1.1 Ce veți putea face după acest curs

**Ce veți putea face după acest capitol.**

- Să alegeți mediul potrivit pentru o distanță și o viteză date
- Să explicați de ce se răsucesc firele într-un cablu UTP și ce înseamnă categoria lui
- Să realizați corect mufarea straight-through, crossover și rollover

- Să explicați diferența dintre fibra monomod și cea multimod
- Să comparați FDM, WDM și TDM
- Să calculați latența unei legături, cu toate componentele ei
- Să recunoașteți atenuarea, diafonia și zgomotul și să știți ce le oprește

## 2.2 Trei familii de medii

Ce înseamnă „mediu de transmisie”. Mediul este pur și simplu lucrul fizic prin care călătorește semnalul. Un fir de cupru, un miez de sticlă, aerul dintr-o încăpere. Nimic mai mult. Împărțirea fundamentală este între medii **ghidate** și **neghidate**. Într-un mediu ghidat, semnalul e constrâns să urmeze un traseu fizic - merge unde îl duce firul. Într-un mediu neghidat, se propagă liber prin spațiu, iar **oricine se află în raza lui îl poate recepționa**. Această singură diferență explică aproape toate deosebirile care urmează, inclusiv cele de securitate: ca să asculți un cablu trebuie să intri în clădire; ca să asculți Wi-Fi îți ajunge să fii în parcare.

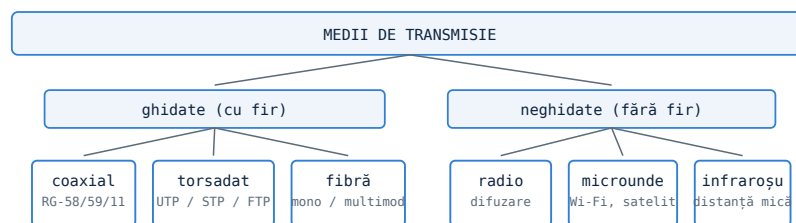


Figura 2.1 Taxonomia mediilor. Rețelele locale moderne folosesc practic doar trei dintre frunzele acestui arbore: cablul torsadat pentru acces, fibra pentru magistrală și microundele pentru mobilitate.

## 2.3 Cablul coaxial

Un conductor central, un strat izolator (dielectric), o tresă metalică de ecranare și o manta exterioară - toate concentrice, de unde și numele.

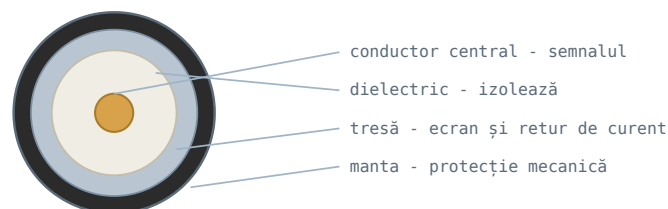


Figura 2.2 Structura unui cablu coaxial. Ecranul are dublu rol: face parte din circuit și, în același timp, izolează semnalul de mediul exterior.

Ethernet a început pe coaxial, dar l-a abandonat - și merită înțeles de ce, pentru că motivul explică forma tuturor rețelelor moderne.

Tabela 2.1 Cablul coaxial.

| Tip   | Impedanță   | Utilizare istorică                            |
|-------|-------------|-----------------------------------------------|
| RG-59 | 75 $\Omega$ | televiziune prin cablu, supraveghere video    |
| RG-58 | 50 $\Omega$ | Thin Ethernet (10BASE2)                       |
| RG-11 | 50 $\Omega$ | Thick Ethernet (10BASE5),<br>magistrale lungi |

**De ce a murit coaxialul în LAN-uri.** Topologia era de tip **magistrală**: un singur cablu, toate calculatoarele agățate de el. Consecințele erau două, ambele grave. Primo, toată lumea împărțea același mediu, deci exista un singur **domeniu de coliziune** pentru toată rețeaua. Secundo, o singură întrerupere a cablului - un conector desfăcut, un scaun tras peste el - pica **tot** segmentul, iar găsirea locului era un coșmar.

Cablul torsadat a adus topologia **stea**: fiecare stație are cablul ei până la un echipament central. O defecțiune afectează o singură stație și se localizează imediat.

## 2.4 Cablul torsadat: de ce se răsucesc firele

Cablul de rețea pe care îl vedeți în orice birou are opt fire, grupate în patru perechi, iar fiecare pereche este răsucită cu un pas ușor diferit de celelalte. Nu e ornament și nu e o consecință a procesului de fabricație. Este soluția la o problemă concretă.

**Transmisia diferențială, explicată fără formule.** Un semnal se poate transmite în două feluri. **Nediferențial**: un fir duce semnalul, iar receptorul îl compară cu masa. **Diferențial**: două fire duc același semnal, dar în oglindă unul față de celălalt, iar receptorul citește **diferența** dintre ele. Diferența pare o complicație inutilă - până apare o perturbație. Un motor pornit alături, un cablu de forță din apropiere, un telefon mobil: toate induc tensiune în fire. Dacă cele două fire ale unei perechi sunt **aproape identic expuse**, perturbația induce în ele aproape aceeași tensiune. Iar când receptorul face diferența, tensiunea comună se **scade singură** și dispare. Ca să fie identic expuse, firele trebuie să schimbe permanent poziția relativă unul față de altul. Adică să fie **răsucite**. Asta e tot.

### Analogie

Doi martori povestesc același eveniment, dar amândoi au fost mituiți cu aceeași sumă și în același fel. Dacă îi ascultați separat, minciuna trece. Dacă vă interesează doar

### diferența

dintre relatările lor, mita comună dispare din calcul și rămâne doar ce diferă cu adevărat.

Pașii diferiți de răsucire între perechi rezolvă o a doua problemă: **diafonia** - semnalul dintr-o pereche care se strecoară în perechea vecină. Dacă toate perechile ar avea același pas, ar sta „în fază” pe toată lungimea cablului și s-ar cupla puternic. Cu pași diferiți, cuplajul se anulează în medie.



Tabela 2.2 Cu ecran sau fără.

| Prescurtare       | Ce înseamnă                                 | Când se folosește                                     |
|-------------------|---------------------------------------------|-------------------------------------------------------|
| <b>UTP</b>        | Unshielded Twisted Pair - fără niciun ecran | majoritatea covârșitoare a instalațiilor de birou     |
| <b>FTP / ScTP</b> | o folie de ecranare peste toate perechile   | medii cu perturbații moderate                         |
| <b>STP</b>        | ecran individual pentru fiecare pereche     | hale industriale, spitale, lângă echipamente de forță |

### 2.4.1 Cu ecran sau fără

#### Observație

**Ecranarea greșit executată e mai rea decât lipsa ei.** Un ecran ajută **doar** dacă este legat corect la masă, la ambele capete, printr-un conector și un panou care îi asigură continuitatea. Un ecran nelegat, sau legat la un singur capăt, se transformă într-o antenă care culege perturbații și le injectează exact acolo unde nu trebuie.

Regula practică: dacă instalația nu are împământare de calitate, UTP-ul bun bate STP-ul prost.

### 2.4.2 Categoriile

„Categorie” unui cablu nu se referă la viteză, ci la **banda de frecvențe** pe care o poate transporta cu parametri garantați. Viteza rezultă din bandă plus schema de codare - de aceea Cat 5e duce 1 Gbps deși are aceeași bandă ca Cat 5.

Tabela 2.3 Categoriile.

| Categorie     | Bandă garantată | Viteză susținută              | Standard tipic                    | Observație                                     |
|---------------|-----------------|-------------------------------|-----------------------------------|------------------------------------------------|
| Cat 3         | 16 MHz          | 10 Mbps                       | 10BASE-T                          | cablarea telefonică veche                      |
| Cat 5         | 100 MHz         | 100 Mbps                      | 100BASE-TX                        | practic scoasă din uz                          |
| Cat 5e        | 100 MHz         | 1 Gbps                        | 1000BASE-T                        | diafonie mult mai bună; minimul acceptabil azi |
| Cat 6         | 250 MHz         | 1 Gbps (10 Gbps până la 55 m) | 1000BASE-T / 10GBASE-T            | alegerea uzuală pentru instalații noi          |
| Cat 6a        | 500 MHz         | 10 Gbps la 100 m              | 10GBASE-T                         | mai gros și mai greu de tras                   |
| Cat 7 / Cat 8 | 600 MHz / 2 GHz | 10 – 40 Gbps                  | legături scurte în centre de date | conectori speciali, distanțe mici              |

**Limita de 100 de metri nu depinde de categorie.** Indiferent dacă folosiți Cat 5e sau Cat 8, un segment de cablu torsadat nu are voie să depășească **100 de metri**: 90 m de cablu fix plus  $2 \times 5$  m de patch-corduri la capete. Limita vine din bugetul de atenuare și din timpul de propagare, nu din calitatea cablului.

Dacă vă trebuie mai mult, aveți exact trei opțiuni: un switch intermediar, un media convertor spre fibră, sau fibră de la bun început.

| CERINȚĂ                                                 | CE ALEGEȚI                                     |
|---------------------------------------------------------|------------------------------------------------|
| Legătură de 40 m între birou și camera de rețea, 1 Gbps | fibră monomod, la 1310 sau 1550 nm             |
| Legătură de 300 m între două clădiri                    | fibră optică - cuprul nu are voie peste 100 m  |
| Traseu de 60 m printr-o hală, lângă motoare mari        | UTP cu alimentare prin PoE                     |
| Magistrală de 20 km între două puncte ale orașului      | Cat 6a, sau Cat 6 dacă rămâneți sub 55 m       |
| Legătură 10 Gbps de 40 m într-un centru de date         | cablu ecranat, cu împământare la ambele capete |
| Punct de acces montat în tavan, fără priză de curent    | UTP Cat 5e sau Cat 6                           |

Apasă un element din stânga, apoi perechea lui din dreapta.

Figura 2.3 Alegeți mediul potrivit fiecărei situații.

**PoE - alimentare prin cablul de rețea.** **Power over Ethernet** transportă și curent, nu doar date, pe același cablu torsadat. Un punct de acces Wi-Fi, o cameră IP sau un telefon IP montat în tavan primesc astfel și rețea, și alimentare, dintr-un singur cablu - fără priză de curent lângă el. Standardele oferă, în ordine, aproximativ 15 W (802.3af), 30 W (802.3at) și până la 90 W (802.3bt). Este unul dintre motivele pentru care cuprul nu a dispărut, deși fibra e superioară în orice altă privință: **fibra nu poate duce curent**.

## 2.5 Cele trei mufări clasice

Ordinea firelor în mufa RJ-45 este stabilită de standardele TIA/EIA-568A și 568B. Diferența dintre ele este doar interschimbarea perechii verzi cu cea portocalie - dar combinația celor două capete decide ce fel de cablu obțineți.

**De ce există regula asta.** Un calculator **transmite** pe pinii 1-2 și **receptează** pe 3-6. Un switch face exact invers: receptează pe 1-2 și transmite pe 3-6 - pentru că a fost proiectat să se lege la calculatoare. Prin urmare, între ele, firele merg drept și fiecare emisie nimereste o recepție. Dar între două calculatoare, ambele ar transmite pe 1-2 și ar asculta pe 3-6: două guri față în față și două urechi față în față. Cablul crossover inversează perechile și repară situația. Aceeași problemă apare între două switch-uri, din același motiv.

**Dacă în practică nu mai contează.** Aproape toate echipamentele fabricate în ultimii douăzeci de ani implementează **Auto-MDI/MDI-X**: detectează singure dacă e nevoie de inversare și o fac electronic. Puteți lega două switch-uri cu cablu direct și va merge.

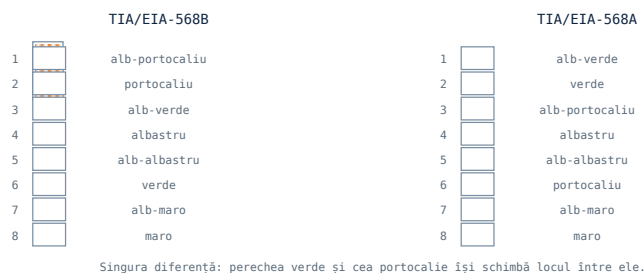


Figura 2.4 Ordinea firelor în cele două standarde. Pinii 4, 5, 7 și 8 sunt identici; se schimbă doar 1, 2, 3 și 6 - exact perechile folosite de Ethernet la 10 și 100 Mbps.

Tabela 2.4 Cele trei mufări clasice.

| Tip de cablu                        | Capăt 1 | Capăt 2                        | Când se folosește                                                                   |
|-------------------------------------|---------|--------------------------------|-------------------------------------------------------------------------------------|
| <b>Straight-through</b><br>(direct) | 568B    | 568B                           | între echipamente de <b>niveluri diferite</b> :<br>PC ↔ switch, switch ↔ ruter      |
| <b>Crossover</b><br>(inversor)      | 568A    | 568B                           | între echipamente de <b>același nivel</b> : PC ↔ PC, switch ↔ switch, ruter ↔ ruter |
| <b>Rollover</b><br>(de consolă)     | 568B    | ordine complet inversată (8→1) | de la calculator la portul de consolă al unui echipament                            |

Regula rămâne totuși de știut, din trei motive: Packet Tracer o respectă strict, la echipamente vechi chiar contează, iar la o legătură care „nu se aprinde” tipul cablului rămâne una dintre primele suspiciuni.

## 2.6 Fibra optică

**Cum stă lumina într-un fir de sticlă.** Fibra are un **miez** de sticlă foarte pură, înconjurat de o **manta** tot din sticlă, dar cu indice de refracție puțin mai mic. Când lumina din miez ajunge la graniță sub un unghi suficient de mic față de perete, nu iese - se reflectă complet înapoi în miez. Fenomenul se numește **reflexie totală internă** și e același care face ca suprafața unei piscine, privită de dedesubt, să pară o oglindă. Raza de lumină avansează astfel în zig-zag prin miez, kilometru după kilometru, fără să scape afară.

### 2.6.1 Ferestrele de lungime de undă

Atenuarea sticlei nu este constantă cu lungimea de undă. Există trei „ferestre” în care este minimă și în care, prin urmare, lucrează toate echipamentele optice.

**Ce e un decibel, pe scurt.** dB nu este o unitate absolută, ci un **raport**, exprimat

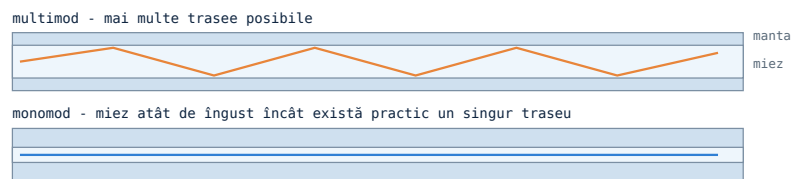


Figura 2.5 Într-o fibră multimod, razele care merg în zig-zag parcurg mai mult decât cele axiale, deci ajung mai târziu: un impuls scurt se lățește pe drum. Fenomenul se numește dispersie modală și limitează distanța. Fibra monomod nu are problema, pentru că nu are decât un singur traseu.

Tabela 2.5 Fibra optică.

| Tip                      | Miez / manta ( $\mu\text{m}$ ) | Sursă de lumină | Distanță tipică          | Cost              |
|--------------------------|--------------------------------|-----------------|--------------------------|-------------------|
| Multimod, in-dice gradat | 50/125<br>62,5/125             | LED sau VCSEL   | sute de metri            | echipament ieftin |
| Monomod                  | ~9/125                         | laser           | zeci - sute de kilometri | echipament scump  |

Tabela 2.6 Ferestrele de lungime de undă.

| Lungime de undă | Atenuare tipică | Utilizare                                                     |
|-----------------|-----------------|---------------------------------------------------------------|
| 850 nm          | ~3,5 dB/km      | multimod, legături scurte, componente ieftine                 |
| 1310 nm         | ~1,25 dB/km     | monomod, distanțe medii                                       |
| 1550 nm         | ~0,5 dB/km      | monomod, distanțe mari; compatibil cu amplificatoarele optice |

logaritmic.  $-3$  dB înseamnă jumătate din putere.  $-10$  dB înseamnă a zecea parte.  $-20$  dB înseamnă a suta parte. Avantajul scării logaritmice: pierderile se **adună** în loc să se înmulțească. Un traseu cu 2 dB de fibră, 1 dB de conectori și 0,5 dB de îmbinări are pierdere totală 3,5 dB - o simplă adunare.

|                                 |                                 |                                            |
|---------------------------------|---------------------------------|--------------------------------------------|
| PUTERE EMITĂTOR (DBM)           | LUNGIME FIBRĂ (KM)              | ATENUARE FIBRĂ (DB/KM)                     |
| <input type="text" value="-8"/> | <input type="text" value="12"/> | <input type="text" value="1550 nm - 0,5"/> |
| NUMĂR CONECTORI (BUC)           | NUMĂR ÎMBINĂRI (BUC)            | SENSIBILITATE RECEPTOR (DBM)               |
| <input type="text" value="4"/>  | <input type="text" value="2"/>  | <input type="text" value="-25"/>           |

pierdere în fibră **6.00 dB**  
 pierdere în conectori (0,5 dB/buc) **2.00 dB**  
 pierdere în îmbinări (0,1 dB/buc) **0.20 dB**  
 putere la receptor **-16.20 dBm**  
 marjă rămasă **8.80 dB** - sub 3 dB legătura este riscantă, sub 0 nu funcționează

Așa se proiectează, în realitate, o legătură optică: se pornește de la puterea emițătorului și se scade tot ce se pierde pe drum. Măriți lungimea până când marja devine negativă.

Figura 2.6 Bugetul optic al unei legături.

**Ce face fibra mai bine.** Distanțe imposibile pentru cupru: zeci de kilometri Nu conduce curent: nicio problemă de împământare, nicio diferență de potențial între clădiri Nu emite și nu culege perturbații electromagnetice Nu poate fi ascultată fără a fi atinsă fizic Bandă enormă, practic nesaturată

**Ce o face incomodă**

- Nu poate transporta alimentare - deci nici PoE
- Conectorizarea cere scule și pregătire; nu se face „din mers”
- Raza de curbură limitată: îndoită prea strâns, lumina scapă
- Un conector murdar poate costa mai mult decât un kilometru de fibră

**Regula de aur a fibrei.** Capacele de protecție nu se scot decât în momentul conectării, iar suprafața se curăță de fiecare dată. Un fir de praf pe un miez de  $9\ \mu\text{m}$  blochează o parte semnificativă din lumină - și produce o defecțiune intermitentă, adică exact genul pe care îl căutați trei zile.

## 2.7 Mediul fără fir

Undele electromagnetice acoperă un spectru enorm, dar rețelele de calculatoare folosesc din el o felie îngustă: microundele, aproximativ între 1 și 10 GHz. Alegerea nu e întâmplătoare și rezultă dintr-un compromis pe care merită să îl aveți în minte ori de câte ori cineva propune „o bandă mai bună”.

Tabela 2.7 Mediul fără fir.

| Proprietate                 | Frecvențe mici             | Frecvențe mari      |
|-----------------------------|----------------------------|---------------------|
| Distanța de propagare       | mare                       | mică                |
| Penetrarea obstacolelor     | bună                       | slabă               |
| Interferențe                | multe - banda e aglomerată | puține              |
| Lățime de bandă disponibilă | mică                       | mare                |
| Costul echipamentelor       | mic                        | mare                |
| Absorbția în atmosferă      | mică                       | crește cu frecvența |

**Traducerea practică.** Frecvențele joase merg departe și trec prin ziduri, dar banda disponibilă acolo e mică și deja ocupată de alții. Frecvențele înalte oferă bandă generoasă, dar sunt oprite de primul perete.

De aceea Wi-Fi la 2,4 GHz acoperă mai mult decât la 5 GHz, iar 5 GHz mai mult decât 6 GHz - exact în ordinea în care **crește** debitul disponibil. Nu există bandă „mai bună”; există bandă potrivită scopului.

### Observație

**Benzile ISM.** Industrial, Scientific and Medical - porțiuni de spectru care pot fi folosite **fără licență și fără taxă**, cu respectarea unor limite de putere. În România, alocările sunt reglementate de ANCOM prin Tabelul Național de Atribuire a Benzilor de Frecvențe. Faptul că banda e liberă explică simultan succesul enorm al Wi-Fi și aglomerația din ea:

Tabela 2.8 Mediul fără fir.

| Tehnologie        | Bandă           | Aplicație tipică                                   |
|-------------------|-----------------|----------------------------------------------------|
| Radio AM          | 540 – 1600 kHz  | difuzare pe unde medii, sute de kilometri          |
| Radio FM          | 88 – 108 MHz    | difuzare comercială                                |
| LoRaWAN           | 868 MHz (UE)    | senzori IoT: kilometri, consum minuscul, debit mic |
| Zigbee, Bluetooth | 2,4 GHz         | automatizări, periferice, distanțe scurte          |
| Wi-Fi             | 2,4 / 5 / 6 GHz | rețele locale fără fir                             |

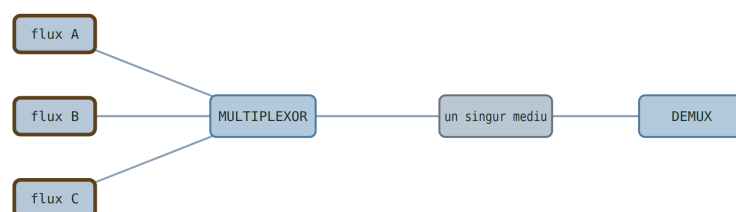
în aceeași bandă de 2,4 GHz stau Wi-Fi, Bluetooth, Zigbee, telecomenzi, stații meteo și cuptorul cu microunde.

## 2.8 Multiplexarea: mai multe conversații pe un mediu

Un mediu costă bani. Fibra trebuie trasă prin canalizație, spectrul trebuie licențiat, canalul trebuie întreținut. Prin urmare, aproape întotdeauna vrem ca pe el să circule mai multe comunicații **simultan**.

### Observație

**Multiplexare și demultiplexare.** **Multiplexarea** este gruparea mai multor fluxuri de date într-un singur semnal transmis peste un mediu partajat. **Demultiplexarea** este operația inversă, la celălalt capăt. Întrebarea centrală e ce anume se împarte: **frecvența, timpul sau lungimea de undă**.



pasul 1 din 4

#### Problema

Trei fluxuri independente trebuie să ajungă la celălalt capăt, dar avem un singur mediu fizic. Fără o schemă de partajare, ele s-ar suprapune și niciunul nu ar ajunge întreg.

Figura 2.7 Trei feluri de a împărți un mediu.

Tabela 2.9 Multiplexarea: mai multe conversații pe un mediu.

| Metodă                                        | Resursa împărțită           | Tip                | Exemplu                                                |
|-----------------------------------------------|-----------------------------|--------------------|--------------------------------------------------------|
| <b>FDM</b> - Frequency Division Multiplexing  | banda de frecvențe          | analogică          | canalele de radio și TV, DSL                           |
| <b>WDM</b> - Wavelength Division Multiplexing | lungimile de undă din fibră | analogică (optică) | magistrale optice; DWDM duce zeci de canale pe o fibră |
| <b>TDM</b> - Time Division Multiplexing       | timpul, în intervale fixe   | digitală           | E1/T1, magistrala unui switch                          |

## 2.9 Performanța unei legături

Trei mărimi descriu o legătură și sunt permanent confundate între ele - inclusiv în reclamele furnizorilor de Internet.

### Observație

**Lățime de bandă, debit util, latență. Lățimea de bandă** este capacitatea teoretică maximă a mediului. Cifra de pe cutie.

**Debitul util** (throughput) este cantitatea de date efectiv transferată în unitatea de timp - mereu mai mică, din cauza anteturilor, coliziunilor și retransmisiilor.

**Latența** este timpul necesar unui bit ca să ajungă dintr-un punct în altul.

### Analogie

O autostradă cu opt benzi are

#### bandă

mare: încap multe mașini simultan. Dar dacă limita de viteză e 30 km/h,

#### latența

rămâne mare - o mașină individuală tot ajunge târziu. Iar dacă jumătate din benzi sunt blocate de lucrări,

#### debitul util

e mult sub capacitatea teoretică.

Bandă mare și latență mare coexistă perfect: exact așa arată o legătură prin satelit.

### 2.9.1 Cele patru componente ale latenței

1. **Timpul de transmisie** - cât durează să „împingi” cadrul pe fir: dimensiune împărțită la debit. Depinde de *cât de mare* e cadrul.
2. **Timpul de propagare** - cât îi trebuie semnalului să parcurgă mediul: aproximativ 5  $\mu$ s pe kilometru, atât în cupru cât și în fibră ( $\approx 2/3$  din viteza luminii în vid). Depinde doar de *distanță*.
3. **Latența echipamentelor** - timpul de procesare din fiecare switch și ruter de pe

traseu. Crește cu nivelul la care lucrează echipamentul.

4. **Timpul de așteptare în cozi** - apare când legătura de ieșire e ocupată. Singura componentă care variază puternic în timp, deci singura care produce **jitter** - și cea care strică videoconferințele.

| DIMENSIUNE CADRU (OCTEȚI)   | DEBITUL LEGĂTURII (MBPS)    | LUNGIMEA LEGĂTURII (M) |
|-----------------------------|-----------------------------|------------------------|
| 1518                        | 100                         | 100                    |
| ECHIPAMENTE PE TRASEU (BUC) | LATENȚĂ PER ECHIPAMENT (MS) |                        |
| 2                           | 20                          |                        |

timp de transmisie **121.44  $\mu$ s**  
 timp de propagare **0.56  $\mu$ s**  
 latența echipamentelor **40.00  $\mu$ s**  
 total, fără cozi **162.00  $\mu$ s**

Încercați două scenarii extreme. **LAN:** 100 m, 2 echipamente - vedeți cine domină. **Transatlantic:** puneți 6000 km și 20 de echipamente și priviți cum propagarea înghite totul.

Figura 2.8 Calculator de latență.

### Exemplu

Un cadru Ethernet de 1518 octeți este transmis pe o legătură de 100 Mbps, lungă de 100 m de UTP Cat 5e. Cât durează până când ultimul bit ajunge la destinație, ignorând echipamentele?

**Rezolvare. Transmisie:**  $1518 \times 8 \text{ biți} / 10^8 \text{ bps} = 12\,144 / 10^8 \approx 121,4 \mu\text{s}$  **Propagare:**  $100 \text{ m} \times 5,56 \text{ ns/m} \approx 0,56 \mu\text{s}$  **Total  $\approx 122 \mu\text{s}$ .** Observați raportul: pe distanțe scurte, transmisia domină complet propagarea - de 200 de ori. Pe o legătură transatlantică situația se inversează: propagarea de  $\sim 30 \text{ ms}$  face ca timpul de transmisie al unui cadru să devină neglijabil. Aceasta este exact rațiunea pentru care protocoalele de transport folosesc **ferestre glisante**: pe o legătură lungă, a aștepta confirmarea fiecărui cadru înainte de a-l trimite pe următorul ar irosi 99 % din capacitate.

**Atenție la unități - din nou.** Mbps se măsoară în puteri ale lui 10 ( $10^6$  biți pe secundă), iar MB în puteri ale lui 2 ( $2^{20}$  octeți). O legătură de 100 Mbps transferă, în cel mai bun caz, aproximativ **11,9 MB** pe secundă - nu 100, nu 12,5.

## 2.10 Ce strică semnalul pe drum

Patru fenomene degradează un semnal, indiferent de mediu. Fiecare are o soluție cunoscută, iar împreună explică practic toate limitele numerice din acest curs.

**Diferența dintre repetor și amplificator.** Un **amplificator** mărește totul: și semnalul, și zgomotul acumulat. După câteva amplificatoare în cascadă, zgomotul devine mai mare decât semnalul. Un **repetor** face altceva: citește biții, îi decide (1 sau 0) și îi regenerează curat. Zgomotul nu se acumulează, pentru că la fiecare pas semnalul e reconstruit de la zero. De aceea lanțurile lungi folosesc repetoare, nu amplificatoare.



Tabela 2.10 Ce strică semnalul pe drum.

| Fenomen                     | Ce se întâmplă                                              | Se manifestă ca                                             | Soluție                                                                 |
|-----------------------------|-------------------------------------------------------------|-------------------------------------------------------------|-------------------------------------------------------------------------|
| <b>Atenuare</b>             | amplitudinea scade cu distanța, până sub pragul de detecție | legătura nu se stabilește deloc peste o anumită lungime     | repetor sau amplificator; respectarea lungimii maxime                   |
| <b>Diafonie</b> (crosstalk) | semnalul dintr-o pereche induce tensiune în perechea vecină | erori care cresc cu traficul                                | torsadare cu pași diferiți, ecranare individuală                        |
| <b>Zgomot</b>               | perturbații externe se suprapun peste semnal                | erori aleatoare, adesea corelate cu un utilaj din apropiere | ecranare, transmisie diferențială, trasee departe de cablurile de forță |
| <b>Dispersie</b>            | impulsul se lățește și se suprapune peste următorul         | limită de distanță pe fibra multimod                        | fibră monomod, regenerare optică                                        |

Două echipamente rezolvă situațiile de graniță:

- **Repetorul** reface semnalul degradat și prelungește segmentul. Există variante electrice, optice și wireless. Atenție: un repetor extinde și domeniul de coliziune, deci nu e gratuit.
- **Media convertorul** traduce între medii diferite - electric ↔ optic, electric ↔ wireless - și permite unei magistrale de fibră să se termine într-un switch cu porturi de cupru. Este soluția standard când trebuie legate două clădiri.

## 2.11 Greșeli frecvente

### Observație

#### Greșeli frecvente.

- **„Cat 6 e mai rapid decât Cat 5e”** Ambele duc 1 Gbps la 100 m. Cat 6 are bandă mai mare și diafonie mai mică, deci mai multă rezervă și posibilitatea de 10 Gbps pe distanțe scurte - dar pe o legătură gigabit obișnuită nu veți vedea nicio diferență. Categoria dă rezervă și viitor, nu viteză imediată.
- **„Am pus cablu ecranat, deci sunt protejat”** Un ecran nelegat corect la masă la ambele capete se comportă ca o antenă și înrăutățește situația. Ecranarea are sens doar împreună cu o instalație de împământare corectă.
- **„Legătura are 150 m, dar merge, deci e în regulă”** Poate merge azi, la trafic mic, și poate cădea peste șase luni când crește traficul sau se schimbă temperatura. Limita de 100 m nu e o sugestie. Peste 100 m: switch intermediar sau fibră.
- **„Bandă mare înseamnă latență mică”** Sunt independente. O legătură prin satelit are bandă mare și latență de sute de milisecunde. Pentru jocuri și telefonie contează latența; pentru descărcări contează banda.

- „Fibra e întotdeauna mai bună, deci o folosim peste tot” Fibra nu duce alimentare, cere conectorizare pregătită și echipamente scumpe la capete. Pentru 40 m până la un birou, UTP-ul e alegerea corectă. Fibra pentru magistrale și distanțe; cuprul pentru acces.

## 2.12 Rezumat și glosar

1. Mediile se împart în **ghidate** și **neghidate**, iar din această singură deosebire decurg diferențele de distanță, cost și securitate.
2. Torsadarea și transmisia diferențială sunt aceeași idee privită din două unghiuri: fac ca perturbația **comună** să se anuleze singură.
3. Fibra câștigă la distanță și imunitate; cuprul câștigă la cost, la simplitate și pentru că poate duce alimentare.
4. Multiplexarea împarte **frecvența** (FDM), **timpul** (TDM) sau **culoarea** (WDM).
5. Latența are patru componente, iar care dintre ele domină depinde complet de scara rețelei.

## 2.13 Ce urmează și unde citiți mai departe

Cu mediul lămurit, cursul următor urcă un nivel: cum se organizează biții în cadre, cum se identifică stațiile între ele, ce se întâmplă când doi vorbesc în același timp și de ce switch-ul a înlocuit hub-ul.

- TIA/EIA-568 - cablarea structurată a clădirilor comerciale
- ITU-T G.652 și G.657 - caracteristicile fibrei monomod
- IEEE 802.3af / at / bt - cele trei generații de PoE
- ANCOM - Tabelul Național de Atribuire a Benzilor de Frecvențe Radio
- Andrew S. Tanenbaum, *Computer Networks*, capitolul 2 - medii de transmisie și multiplexare

## Întrebări de verificare

1. De ce sunt firele unei perechi UTP rasucite între ele?
  - a) Ca să încapă mai multe fire în cablu
  - b) Pentru că perturbația induce aceeași tensiune în ambele fire, iar receptorul citește diferența dintre ele
  - c) Ca să crească impedanța cablului
  - d) Pentru a permite alimentarea PoE
2. Ce cablu se folosește între două switch-uri, la echipamente fără Auto-MDI/MDI-X?
  - a) Straight-through

- b) Crossover
  - c) Rollover
  - d) Coaxial
3. De ce fibra monomod duce mult mai departe decat cea multimod?
- a) Pentru ca are miezul mai gros
  - b) Pentru ca miezul ingust permite practic un singur traseu al luminii, deci nu apare dispersie modala
  - c) Pentru ca foloseste LED-uri in loc de laser
  - d) Pentru ca are atenuare mai mare
4. O legatura de 100 m intre doua switch-uri intr-o cladire: care componenta a latentei domina?
- a) Propagarea
  - b) Latenta echipamentelor
  - c) Timpul de transmisie si latenta echipamentelor
  - d) Jitterul
5. Ce imparte WDM intre fluxuri?
- a) Intervale de timp
  - b) Lungimi de unda diferite pe aceeasi fibra
  - c) Perechi diferite din cablul torsadat
  - d) Adrese MAC diferite
6. Un cablu UTP de 130 m intre doua switch-uri. Ce faceti?
- a) Nimic, daca legatura se aprinde
  - b) Puneti un switch intermediar sau treceti pe fibra
  - c) Folositi cablu Cat 8
  - d) Reduceti viteza la 10 Mbps
7. Ce inseamna o atenuare de 3 dB?
- a) Semnalul a scazut cu 3%
  - b) Semnalul a scazut la jumatate din putere
  - c) Semnalul a crescut de 3 ori
  - d) Nu inseamna nimic fara o referinta

8. De ce nu se poate alimenta un punct de acces prin fibra optica?

- a) Pentru ca fibra e prea subtire
- b) Pentru ca fibra transporta lumina, nu curent electric
- c) Pentru ca standardul nu permite
- d) Pentru ca ar arde miezul



## 3. Nivelul legătură de date

---

---

Din cursurile trecute  
Ce îi lipsește nivelului fizic  
Cele două subniveluri: LLC și MAC  
Încapsularea la nivelul 2  
Cadrul Ethernet, câmp cu câmp  
Adresa MAC  
Timpii caracteristici ai unei rețele Ethernet  
Mediul partajat și CSMA/CD  
Domenii de coliziune și de broadcast  
Cum funcționează un switch  
Metode de comutare  
Familia Ethernet, pe scurt  
Greșeli frecvente  
Rezumat și glosar  
Ce urmează și unde citiți mai departe  
Întrebări de verificare

---

---

Nivelul fizic mută biți, dar nu știe cui. Nivelul legătură de date rezolvă exact această lipsă: împachetează biții în cadre, dă fiecărei interfețe un nume unic pe lume, stabilește cine are voie să vorbească pe un mediu partajat și - prin switch - construiește singur harta rețelei locale, fără ca nimeni să îi spună nimic.

### 3.1 Din cursurile trecute

**Ce trebuie să aveți în minte.** Nivelul 1, **fizic** transformă biți în semnal: tensiuni, lumină sau unde radio. Mediul poate fi **ghidat** (cupru, fibră) sau **neghidat** (radio). Un mediu partajat înseamnă că mai mulți vorbesc pe același suport. Fiecare nivel adaugă propriul **antet** - proces numit **încapsulare**. Cursul acesta urcă la **nivelul 2**: adrese MAC, cadre, comutare.

#### 3.1.1 Ce veți putea face după acest curs

**Ce veți putea face după acest capitol.**

- Să explicați rolul nivelului 2 și separarea lui în subnivelurile LLC și MAC
- Să citiți un cadru Ethernet câmp cu câmp și să spuneți la ce folosește fiecare

- Să recunoașteți o adresă MAC unicast, multicast sau broadcast
- Să explicați CSMA/CD și să spuneți de ce a dispărut din rețelele moderne
- Să numărați corect domeniile de coliziune și de broadcast dintr-o topologie
- Să descrieți cele trei acțiuni ale unui switch: învățare, comutare, inundare
- Să comparați metodele de comutare store-and-forward și cut-through

### 3.2 Ce îi lipsește nivelului fizic

Nivelul fizic gestionează un flux de biți. Atât. Are trei lipsuri importante, iar nivelul 2 există exact ca să le compenseze:

Tabela 3.1 Ce îi lipsește nivelului fizic.

| Nivelul fizic nu poate...                        | Nivelul legătură de date...               |
|--------------------------------------------------|-------------------------------------------|
| să identifice stațiile de pe mediu               | le identifică prin <b>adresare</b>        |
| să spună unde începe și unde se termină un mesaj | organizează fluxul în <b>cadre</b>        |
| să comunice direct cu software-ul                | oferă un <b>serviciu clar</b> nivelului 3 |

**De ce e nevoie de „cadre”.** Imaginați-vă că primiți un șir nesfârșit de litere, fără spații și fără semne de punctuație. Ca să înțelegeți ceva, trebuie să știți unde se termină un cuvânt și unde începe altul. Un **cadru** (în engleză *frame*) face exact asta pentru biți: îi grupează într-un pachet cu început clar, sfârșit clar, adresa destinatarului și o sumă de control care spune dacă a ajuns întreg.

### 3.3 Cele două subniveluri: LLC și MAC

Nivelul 2 stă exact la granița dintre hardware și software, așa că a fost împărțit în două subniveluri, fiecare privind în altă direcție.

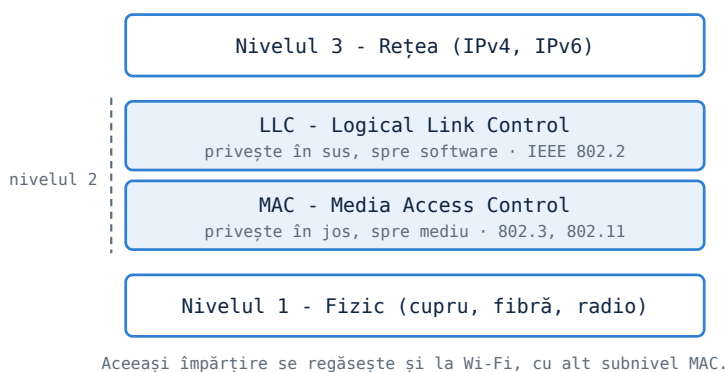


Figura 3.1 Nivelul 2, văzut din interior. LLC este independent de mediu; MAC depinde complet de el.

**De ce împărțirea asta contează.** Datorită subnivelului LLC, același pachet IP circulă neschimbat peste Ethernet, peste Wi-Fi sau peste PPP. Schimbarea mediului cere doar înlocuirea subnivelului MAC - restul stivei nu află nimic.

Tabela 3.2 Cele două subniveluri: LLC și MAC.

| Subnivel   | Cu cine vorbește           | Ce face                                                                                       | Standard            |
|------------|----------------------------|-----------------------------------------------------------------------------------------------|---------------------|
| <b>LLC</b> | cu software-ul (nivelul 3) | multiplexează protocoalele de nivel superior, oferă control de flux; nu îl interesează mediul | IEEE 802.2          |
| <b>MAC</b> | cu mediul (nivelul 1)      | construiește cadrul, decide cine transmite și când; depinde complet de tehnologie             | 802.3, 802.11, FDDI |

Este exact separarea care permite unui laptop să treacă de pe cablu pe Wi-Fi fără ca vreo aplicație deschisă să observe.

### 3.4 Încapsularea la nivelul 2

Ca datele să ajungă la destinatarul potrivit e nevoie de informație suplimentară, iar aceasta este adăugată de nivelul 2 și organizată în cadre. Majoritatea protocoalelor de nivel 2 - nu doar Ethernet - folosesc același set de câmpuri:

Tabela 3.3 Încapsularea la nivelul 2.

| Câmp                            | La ce folosește                                                                                        |
|---------------------------------|--------------------------------------------------------------------------------------------------------|
| <b>Start cadru</b>              | o secvență de biți care anunță începutul; fără ea receptorul nu ar ști de unde să înceapă să numere    |
| <b>Adrese<br/>Tip / lungime</b> | adresele MAC ale sursei și destinației<br>ce protocol de nivel 3 este încapsulat, sau lungimea datelor |
| <b>Date</b>                     | mesajul propriu-zis, adică tot ce a construit nivelul 3                                                |
| <b>CRC / FCS</b>                | o sumă de control care permite detectarea erorilor de transmisie                                       |

Alte protocoale de nivel 2, pe care le veți întâlni în WAN-uri: **PPP**, **Frame Relay**, **ATM**, iar istoric **Token Ring**. Toate rezolvă aceleași probleme, cu alte compromisuri.

### 3.5 Cadrul Ethernet, câmp cu câmp

Structura cadrului este aproape identică pentru toate implementările Ethernet, de la 10 Mbps la 100 Gbps. Aceasta e o realizare remarcabilă de inginerie: viteza a crescut de zece mii de ori, iar formatul a rămas.

**Ce e un CRC, pe scurt.** **Cyclic Redundancy Check** este un fel de „amprentă” a mesajului: emițătorul o calculează printr-o operație matematică peste toți biții și o atașază



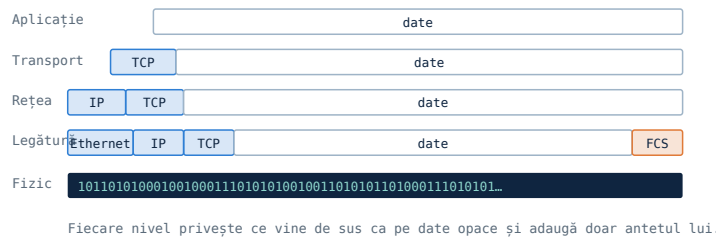


Figura 3.2 Încapsularea, văzută de sus în jos. Observați că nivelul 2 este singurul care adaugă ceva și la sfârșitul mesajului: suma de control.

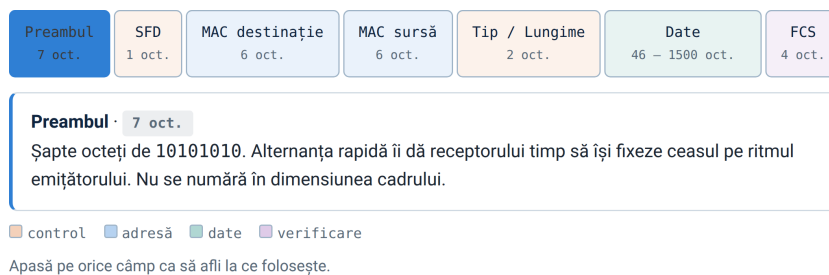


Figura 3.3 Apăsați pe fiecare câmp al cadrului Ethernet.

la sfârșit. Receptorul recalculează aceeași amprentă și compară. Dacă valorile diferă, ceva s-a stricat pe drum. CRC-ul nu spune *ce* s-a stricat și nu poate repara - doar ridică mâna. E o alegere deliberată: detectarea e ieftină, corectarea ar fi scumpă, iar la rata de eroare a unui cablu modern e mai eficient să retransmiți întregul cadru.

**MTU: cifra care revine tot semestrul.** MTU (Maximum Transmission Unit) este dimensiunea maximă a încărcăturii utile - 1500 de octeți la Ethernet clasic. Un pachet IP mai mare trebuie fragmentat sau respins.

Veți întâlni MTU-ul din nou la tunelare (cursul 9), unde antetul suplimentar micșorează spațiul disponibil, și la OSPF (cursul 10), unde un MTU nepotrivit între două rutere blochează adiacența într-o stare pe care altfel nu ați putea-o explica.

### 3.6 Adresa MAC

Ethernet identifică fiecare interfață printr-o adresă de 48 de biți, scrisă în hexazecimal. Este scrisă în memoria ROM a plăcii de rețea, din fabrică.

**De ce sunt unice pe toată planeta.** IEEE vinde blocuri de 24 de biți (OUI - Organizationally Unique Identifier) producătorilor de echipamente. Fiecare producător primește un prefix propriu și îl completează singur cu ce dorește, atât timp cât nu repetă. Rezultatul:  $2^{48} \approx 281$  de mii de miliarde de adrese posibile, împărțite administrativ, fără nicio autoritate care să verifice fiecare placă în parte. Sistemul funcționează pentru că fiecare producător are interesul să nu producă două plăci identice. Ca observație practică: din primii trei octeți ai unei adrese MAC puteți afla producătorul. Există baze de date publice pentru asta, iar la depanare este surprinzător de util.

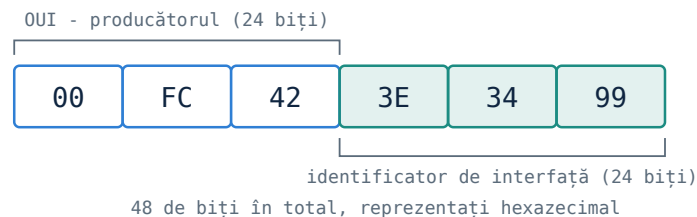


Figura 3.4 Structura unei adrese MAC. Primii 24 de biți formează OUI -ul, atribuit de IEEE producătorului; ultimii 24 sunt aleși de producător, de regulă în ordine de fabricație.

Adresa MAC are două proprietăți esențiale, ambele cu consecințe directe asupra a ceea ce urmează în curs:

1. Este **plată** - nu conține nicio ierarhie. Din adresa 00:FC:42:3E:34:99 nu se poate deduce absolut nimic despre locul unde se află acea placă.
2. Are **domeniu local** - este vizibilă și utilizabilă doar în interiorul domeniului de broadcast. Dincolo de primul ruter, nimeni nu o mai vede.

**Consecința.** Un ruter care ar trebui să dirijeze trafic după adrese MAC ar avea nevoie de o intrare separată pentru **fiecare interfață din lume** - fără nicio posibilitate de grupare, pentru că adresele nu sunt organizate geografic. Tabela nu ar încăpea nicăieri și nu ar putea fi întreținută de nimeni.

Exact aceste două proprietăți fac necesar nivelul 3, subiectul cursului 5.

### 3.6.1 Cele trei tipuri de adrese

Tabela 3.4 Cele trei tipuri de adrese.

| Tip              | Cum se recunoaște                                                | Exemplu           | Cine o acceptă                |
|------------------|------------------------------------------------------------------|-------------------|-------------------------------|
| <b>Unicast</b>   | primul octet <b>par</b> (bitul cel mai puțin semnificativ = 0)   | 00:10:A7:22:FE:63 | o singură interfață           |
| <b>Multicast</b> | primul octet <b>impar</b> (bitul cel mai puțin semnificativ = 1) | 01:00:5E:00:A1:11 | stațiile abonate ace-lui grup |
| <b>Broadcast</b> | toți cei 48 de biți pe 1                                         | FF:FF:FF:FF:FF:FF | toate stațiile din do-meniu   |

#### Exemplu

Care dintre următoarele adrese sunt multicast? 02:00:00:00:00:01, 01:80:C2:00:00:00, 33:33:00:00:00:01, FF:FF:FF:FF:FF:FF

**Rezolvare.** 02 = 00000010 - ultimul bit este 0, deci **unicast**. (Bitul următor, pe 1, înseamnă „administrată local”, adică setată de software, nu din fabrică.) 01 = 00000001 -

ultimul bit este 1, deci **multicast**. Această adresă anume, 01:80:C2:00:00:00, este cea pe care circulă mesajele Spanning Tree din cursul 4. 33 = 00110011 - ultimul bit este 1, deci **multicast**. Prefixul 33:33 este rezervat pentru multicast IPv6. FF:FF:FF:FF:FF:FF este **broadcast** - tehnic, un caz particular de multicast, în care grupul este „toată lumea”.

### 3.7 Timpii caracteristici ai unei rețele Ethernet

Trei mărimi de timp explică aproape toate valorile „ciudate” din standardul Ethernet.

Tabela 3.5 Timpii caracteristici ai unei rețele Ethernet.

| Mărime                | Definiție                                                                    | 10 Mbps                                | 100 Mbps                       | 1 Gbps |
|-----------------------|------------------------------------------------------------------------------|----------------------------------------|--------------------------------|--------|
| <b>Bit time</b>       | durata unui singur bit pe fir                                                | 100 ns                                 | 10 ns                          | 1 ns   |
| <b>Slot time</b>      | timpul necesar semnalului să parcurgă cel mai lung segment și să se întoarcă | 512 bit times =<br><b>64 de octeți</b> | 4096 bit times =<br>512 octeți |        |
| <b>Interframe gap</b> | pauza minimă obligatorie între două cadre succesive                          | 96 bit times                           |                                |        |

**De unde vine dimensiunea minimă de 64 de octeți.** Slot time-ul explică o valoare care altfel pare complet arbitrară.

Într-un mediu partajat, o stație trebuie să **încă transmită** în momentul în care o eventuală coliziune se întoarce la ea. Dacă ar termina mai devreme, ar pleca liniștită mai departe, convinsă că totul a mers bine - și nimeni nu ar retransmite cadrul pierdut.

Cadrul minim este exact atât cât să dureze un slot time. Cadrele mai scurte, rezultate din coliziuni, se numesc **runt frames** și sunt aruncate.

Interframe gap-ul are alt rol: dă stațiilor lente timp să proceseze cadrul curent și să se pregătească pentru următorul. La 1 Gbps înseamnă 96 de nanosecunde - o eternitate pentru electronică, o nimica toată pentru trafic.



Figura 3.5 Calculator de timpi Ethernet.

### 3.8 Mediul partajat și CSMA/CD

Ethernet a fost proiectat pentru un mediu multiacces: un singur cablu coaxial la care se conectau toate stațiile. Într-un astfel de mediu, două stații pot începe să transmită exact în același timp, iar semnalele lor se suprapun. Rezultatul se numește **coliziune**, iar cadrele implicate devin inutilizabile.

#### Analogie

O ședință fără moderator, la telefon. Regula nescrisă este: ascuți, iar dacă nu vorbește nimeni, începi. Dacă doi încep simultan, se aude un talmeș-balmeș, amândoi se opresc, fiecare așteaptă un timp

#### diferit

- altfel ar reporni tot simultan - și încearcă din nou. Aceasta e, cuvânt cu cuvânt, descrierea CSMA/CD.

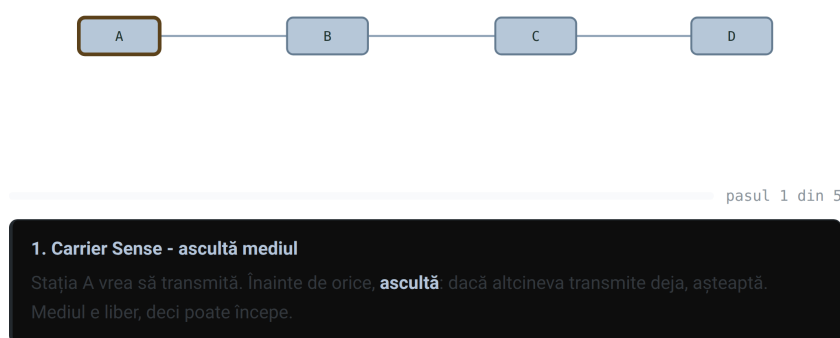


Figura 3.6 CSMA/CD, pas cu pas.

#### Observație

**CSMA/CD este astăzi istorie - dar nu inutilă.** Într-o rețea comutată full-duplex, fiecare port al switch-ului formează, împreună cu stația conectată, un domeniu de coliziune propriu, în care există exact doi participanți și **fire separate pentru fiecare sens**. Coliziunile nu mai pot apărea fizic. 10 Gigabit Ethernet nici măcar nu mai definește modul half-duplex.

Mecanismul rămâne totuși esențial de înțeles, din trei motive: explică dimensiunea minimă a cadrului, explică distanțele maxime ale segmentelor și - mai ales - este ideea pe care Wi-Fi a trebuit să o reinventeze, sub forma CSMA/CA din cursul 11, pentru că în aer coliziunile **nu pot fi detectate**.

### 3.9 Domenii de coliziune și de broadcast

#### Observație

**Cele două domenii.** **Domeniul de coliziune** este grupul de segmente în care două transmisii simultane se pot ciocni. Cu cât e mai mic, cu atât mai bine - ideal, doi participanți.

**Domeniul de broadcast** este grupul de stații care primesc un cadru de broadcast emis de oricare dintre ele. Cu cât e mai mare, cu atât mai mult trafic inutil procesează fiecare stație.

Tabela 3.6 Domenii de coliziune și de broadcast.

| Echipament      | Domeniul de coliziune                      | Domeniul de broadcast                           |
|-----------------|--------------------------------------------|-------------------------------------------------|
| Hub / repetor   | îl extinde                                 | îl extinde                                      |
| Switch / bridge | <b>îl delimitează</b> - câte unul per port | îl extinde                                      |
| Ruter           | îl delimitează                             | <b>îl delimitează</b> - câte unul per interfață |

**Cum se numără, practic. Domenii de coliziune:** porniți de la fiecare cablu. Un cablu care duce la un port de switch sau de ruter își are propriul domeniu. Un hub, în schimb, unește într-un singur domeniu **toate** cablurile conectate la el. **Domenii de broadcast:** același raționament, dar de data asta și switch-urile unesc - doar ruterele separă. Practic: numărați câte „insule” rămân dacă tăiați rețeaua la fiecare interfață de ruter.

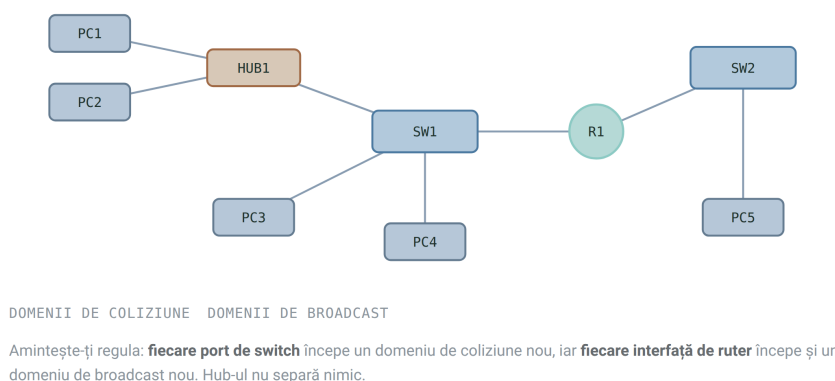


Figura 3.7 Exercițiu: numărați domeniile din această topologie.

#### Exemplu

Un ruter are două interfețe. Pe prima este conectat un switch cu 4 stații; pe a doua, un hub cu 3 stații. Câte domenii de coliziune și câte domenii de broadcast există?

**Rezolvare. Domenii de coliziune: 6.** Switch-ul creează câte unul pentru fiecare dintre cele 4 legături spre stații, plus unul pentru legătura cu ruterul - deci 5. Hub-ul,

împreună cu cele 3 stații și cu legătura spre ruter, formează un singur domeniu. Total  $5 + 1 = 6$ . **Domenii de broadcast: 2.** Câte unul pentru fiecare interfață a ruterului. Toate cele 4 stații de pe switch se aud în broadcast între ele; toate cele 3 de pe hub, la fel; dar cele două grupuri nu se aud reciproc. Capcana clasică: legătura switch-ruter este un domeniu de coliziune propriu, ușor de uitat. Iar legătura hub-ruter **nu** este un domeniu separat, ci face parte din domeniul hub-ului.

### 3.10 Cum funcționează un switch

Aici se află partea cea mai elegantă din tot cursul. Un switch nu este configurat de nimeni cu harta rețelei: și-o construiește singur, dintr-o singură observație inteligentă.

**Observația care face totul posibil. Adresa MAC sursă a unui cadru spune pe ce port se află expeditorul.**

Nu e nevoie de nicio configurare, de niciun protocol de descoperire, de nicio autoritate centrală. Informația este deja acolo, în fiecare cadru care trece - trebuie doar citită și reținută.

Asocierile se păstrează în **tabela CAM** (Content Addressable Memory), numită și tabela de adrese MAC. Pentru fiecare cadru primit, switch-ul execută întotdeauna aceiași doi pași, în această ordine:

1. **Învățare.** Citește MAC-ul **sursă**. Dacă nu există în tabelă, adaugă perechea (MAC sursă, port de intrare). Dacă există deja, îi resetează contorul de vârstă.
2. **Comutare.** Citește MAC-ul **destinație** și îl caută în tabelă. Trei rezultate posibile: găsit, pe **alt** port → **unicast**: trimite cadrul doar pe acel port; găsit, pe chiar **portul de intrare** → **drop**: destinația e de aceeași parte, retransmiterea ar fi inutilă; **negăsit**, sau adresă de broadcast/multicast → **flood**: trimite pe toate porturile în afară de cel de intrare.

SURSĂ

A ▼

DESTINAȚIE

C ▼

Tabela CAM a switch-ului

| <small>ADRESĂ MAC</small>    | <small>PORT</small> |
|------------------------------|---------------------|
| <i>tabela CAM este goală</i> |                     |

Ce face switch-ul

niciun cadru trimis încă

Figura 3.8 Simulator: tabela CAM a unui switch cu patru stații.

**Experimentul de făcut, în ordinea asta.** Goliți tabela. Trimiteți un cadru de la **A** la **C**. Observați: switch-ul învață A, dar nu știe unde e C, deci **inundă**. Trimiteți acum de la **C** la **A**. Switch-ul învață și C, iar A îl știe deja, deci **unicast**. Trimiteți din nou de la **A** la **C**. De data asta este direct unicast. Ce ați văzut: rețeaua **se învață singură** după primul schimb, iar traficul inutil dispare de la sine. Niciun administrator nu a intervenit.

### 3.10.1 Îmbătrânirea intrărilor

Fiecare intrare din tabelă are asociată o **vârstă**, resetată la fiecare cadru primit de la acea adresă. Când vârsta depășește un prag - implicit 300 de secunde pe echipamentele Cisco - intrarea este ștearsă.

#### Exemplu

De ce trebuie șterse intrările? Ce s-ar întâmpla dacă tabela ar fi permanentă?

**Rezolvare.** Dacă mutați un calculator de pe portul 3 pe portul 7, switch-ul ar continua să trimită traficul pentru el pe portul 3 - la nesfârșit, deși acolo nu mai e nimeni. În practică, problema se rezolvă de la sine mult mai repede: primul cadru emis de calculator de pe noul port corectează imediat intrarea. Îmbătrânirea acoperă cazul în care stația a plecat și **nu mai vorbește deloc** - altfel tabela s-ar umple, în timp, cu fantome. Există și un motiv de securitate: o tabelă plină face switch-ul să inunde tot traficul, ceea ce este exact scopul atacului *CAM overflow* din cursul 12.

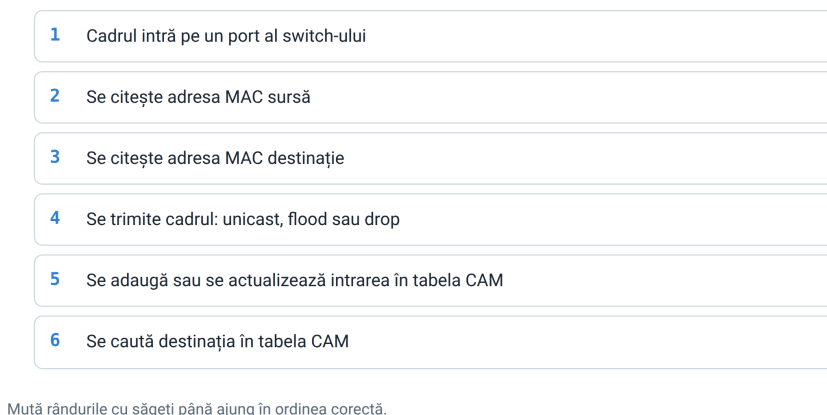


Figura 3.9 Puneți în ordine pașii pe care îi face un switch.

### 3.11 Metode de comutare

Cât așteaptă un switch înainte să ia decizia? Există trei răspunsuri, cu trei compromisuri diferite.

Compromisul e limpede: cu cât switch-ul citește mai mult înainte de a decide, cu atât filtrează mai bine, dar cu atât întârzie mai mult.

**Ce se folosește în practică.** Echipamentele moderne rulează, aproape întotdeauna, **store-and-forward**. Costul în latență a devenit neglijabil la vitezele actuale, iar verificarea FCS-ului este obligatorie: un switch cut-through **propagă** cadrele corupte mai departe în rețea, unde vor fi aruncate oricum - dar după ce au consumat bandă.

Cut-through supraviețuiește în centre de date și în tranzacții financiare, unde microsecundele contează cu adevărat.

Tabela 3.7 Metode de comutare.

| Metodă                            | Ce așteaptă înainte de a comuta                        | Latență                              | Cadre eronate                   |
|-----------------------------------|--------------------------------------------------------|--------------------------------------|---------------------------------|
| <b>Store-and-forward</b>          | cadrul întreg; verifică lungimea și FCS-ul             | mare, crește cu dimensiunea cadrului | oprite                          |
| <b>Fragment-free</b>              | primii 64 de octeți                                    | medie, fixă                          | runt frames oprite, restul trec |
| <b>Fast-forward</b> (cut-through) | doar adresa destinație - primii 6 octeți după preambul | minimă, fixă                         | toate trec                      |

### 3.12 Familia Ethernet, pe scurt

Ethernet a apărut în 1973 la Xerox PARC, realizat de Bob Metcalfe și David Boggs, cu o lățime de bandă de 2,94 Mbps. Numele vine de la „eter” - substanța ipotetică prin care se credea cândva că se propagă lumina. În 1983, IEEE a transformat standardul industrial DIX (DEC–Intel–Xerox) în standardul **802.3**, care rezistă și azi.

Tabela 3.8 Familia Ethernet, pe scurt.

| Generație        | An        | Standard             | Viteză        | Codare                        | CSMA/CD                      |
|------------------|-----------|----------------------|---------------|-------------------------------|------------------------------|
| Fast Ethernet    | 1995      | 100BASE-TX / FX      | 100 Mbps      | 4B/5B<br>+ MLT-3<br>sau NRZ-I | da (half-duplex)             |
| Gigabit Ethernet | 1998–1999 | 1000BASE-T / SX / LX | 1 Gbps        | PAM-5<br>sau 8B/10B           | opțional                     |
| 10 Gigabit       | 2002      | 10GBASE-T / SR / LR  | 10 Gbps       | diverse                       | <b>nu</b> - doar full-duplex |
| 40 / 100 Gigabit | 2010      | 802.3ba              | 40 / 100 Gbps | multiplă, pe mai multe benzi  | nu                           |

**Ce s-a schimbat și ce a rămas.** Viteza a crescut de zece mii de ori. Mediul s-a schimbat de trei ori. CSMA/CD a dispărut complet.

**Formatul cadrului a rămas același.** Un cadru capturat astăzi pe o legătură de 100 Gbps are exact aceleași câmpuri ca unul din 1983. Aceasta e cea mai bună lecție de proiectare pe care o oferă acest curs: o interfață bine gândită supraviețuiește implementărilor ei.



### 3.13 Greșeli frecvente

#### Observație

##### Greșeli frecvente.

- „**Switch-ul învață din adresa MAC destinație**” Nu. Învață exclusiv din adresa **sursă** - singura care spune sigur de unde a venit cadrul. Destinația se folosește doar pentru a decide unde îl trimite. Rețineți ordinea: întâi învață din sursă, apoi caută destinația.
- „**Un cadru de broadcast este trimis și înapoi pe portul de intrare**” Nicio-dată. Flood înseamnă „pe toate porturile **în afară** de cel pe care a venit”. Altfel s-ar crea o buclă instantanee, chiar și fără cabluri redundante.
- „**Dimensiunea minimă de 64 de octeți e o convenție arbitrară**” Vine direct din slot time: stația trebuie să încă transmită când coliziunea se întoarce la ea.  $64 \text{ octeți} = 512 \text{ bit times} = \text{slot time la } 10 \text{ și } 100 \text{ Mbps}$ .
- „**Un switch cu 24 de porturi are 24 de domenii de broadcast**” Are **24 de domenii de coliziune** și **unul singur** de broadcast. Switch-ul separă coliziunea, dar extinde broadcast-ul. Separarea broadcast-ului cere un ruter - sau VLAN-uri, din cursul 4.
- „**FCS-ul corectează erorile**” Doar le detectează. Un cadru cu FCS greșit se aruncă, iar retransmiterea rămâne treaba nivelurilor superioare. Detectare la nivelul 2, recuperare la nivelul 4.

### 3.14 Rezumat și glosar

1. Nivelul 2 adaugă ce lipsea nivelului fizic: **identitate** (adrese MAC), **delimitare** (cadre) și **detectarea erorilor** (FCS).
2. Adresa MAC este **plată** și **locală** - două limitări care fac necesar nivelul 3.
3. CSMA/CD a rezolvat problema mediului partajat; comutarea full-duplex a făcut-o inutilă.
4. Un switch își construiește singur harta rețelei, din adresele MAC sursă ale cadrelor care trec prin el.
5. Switch-ul **separă** domeniile de coliziune și **extinde** pe cele de broadcast - de aici pornește tot cursul următor.

### 3.15 Ce urmează și unde citiți mai departe

Un switch rezolvă coliziunile, dar lasă un singur domeniu de broadcast oricât de mare ar fi rețeaua - și nu tolerează bucle. Cursul următor atacă ambele probleme: **VLAN-uri** pentru segmentare și **Spanning Tree** pentru redundanță fără bucle.

La **laboratorul 2** veți goli o tabelă CAM reală, o veți umple urmărind-o cadru cu cadru și veți compara, pe aceeași topologie, un switch cu un hub.

- IEEE 802.3 - formatul cadrului și mecanismul CSMA/CD
- IEEE 802.1D - operarea unui bridge transparent
- Registrul public IEEE OUI, pentru identificarea producătorului după adresa MAC
- Andrew S. Tanenbaum, *Computer Networks*, capitolul 4 - subnivelul de acces la mediu
- Charles Spurgeon, *Ethernet: The Definitive Guide*

### Întrebări de verificare

1. Un switch primește un cadru a cărui adresa MAC destinație nu se află în tabela CAM. Ce face?
  - a) Il arunca
  - b) Il trimite pe toate porturile în afara de cel de intrare
  - c) Il trimite înapoi pe portul de intrare
  - d) Cere tabela CAM de la switch-ul vecin
2. De ce cadrul Ethernet minim are 64 de octeți?
  - a) Ca să încapă antetul IP
  - b) Pentru ca transmisia trebuie să dureze cel puțin un slot time, altfel stația nu ar detecta coliziunea
  - c) Pentru ca FCS ocupă 4 octeți
  - d) Este o valoare aleasă arbitrar de IEEE
3. Care adresa MAC este multicast?
  - a) FF:FF:FF:FF:FF:FF
  - b) 00:10:A7:22:FE:63
  - c) 01:00:5E:00:A1:11
  - d) 02:00:00:00:00:01
4. Câte domenii de broadcast are un switch cu 24 de porturi, fără VLAN-uri?
  - a) 24
  - b) 12
  - c) 1
  - d) 0
5. Ce conține câmpul Tip/Lungime când are valoarea 0x0800?
  - a) Lungimea cadrului în octeți
  - b) Protocolul de nivel 3 încapsulat, adică IPv4

- c) Numarul VLAN-ului
  - d) Suma de control
6. Din ce campuri invata un switch, ca sa isi construiasca tabela CAM?
- a) Din adresa MAC destinatie
  - b) Din adresa MAC sursa
  - c) Din adresa IP sursa
  - d) Din campul EtherType
7. De ce nu mai apar coliziuni intr-o retea comutata full-duplex?
- a) Pentru ca switch-urile sunt mai rapide
  - b) Pentru ca fiecare port are doi participanti si fire separate pentru fiecare sens
  - c) Pentru ca CSMA/CD a fost imbunatatit
  - d) Pentru ca se foloseste fibra optica
8. Un switch cut-through primeste un cadru corupt. Ce se intampla?
- a) Il arunca
  - b) Il trimite mai departe, pentru ca a decis deja pe baza adresei destinatie
  - c) Il repara folosind FCS
  - d) Il retine pana la verificare

## 4. Optimizarea rețelelor locale

---

---

|                                                     |
|-----------------------------------------------------|
| Din cursul trecut                                   |
| Ce nu merge într-un LAN plat                        |
| Ce este un VLAN                                     |
| Porturi de acces și legături trunk                  |
| VLAN-ul nativ                                       |
| Rutarea între VLAN-uri                              |
| Redundanța și problema buclelor                     |
| Spanning Tree: rolurile și costurile                |
| Cei patru pași ai algoritmului                      |
| Stările porturilor și timpul de convergență         |
| Variantele moderne de STP                           |
| EtherChannel: mai multe cabluri, o singură legătură |
| Greșeli frecvente                                   |
| Rezumat și glosar                                   |
| Ce urmează și unde citiți mai departe               |
| Întrebări de verificare                             |

---

---

Un switch cu 48 de porturi rezolvă coliziunile, dar lasă 48 de calculatoare într-un singur domeniu de broadcast, fără nicio separare între departamente și fără nicio cale de rezervă dacă un cablu cedează. Cursul acesta prezintă cele patru unelte care transformă o grămadă de switch-uri într-o rețea proiectată: VLAN-ul, legătura trunk, Spanning Tree și agregarea legăturilor.

### 4.1 Din cursul trecut

**Ce trebuie să aveți în minte.** Un switch **învață** din adresa MAC sursă, **comută** după adresa MAC destinație și **inundă** când nu știe unde e destinatarul. Switch-ul **separă** domeniile de coliziune (unul per port) dar **extinde** domeniul de broadcast - toate porturile lui formează unul singur. Un cadru de broadcast este inundat pe toate porturile, indiferent ce conține tabela CAM. Nivelul 2 **nu are TTL**. Rețineți asta - este cheia întregii a doua jumătăți a acestui curs.

#### 4.1.1 Ce veți putea face după acest curs

Ce veți putea face după acest capitol.

- Să explicați ce problemă rezolvă un VLAN și de ce nu e suficient un ruter
- Să distingeți un port de acces de un port trunk și să citiți o etichetă 802.1Q
- Să explicați rolul VLAN-ului nativ și riscul lui de securitate
- Să comparați cele două soluții de rutare între VLAN-uri
- Să explicați de ce o buclă la nivelul 2 este fatală, iar una la nivelul 3 nu
- Să rulați, pas cu pas, algoritmul STA pe o topologie dată
- Să configurați o legătură agregată și să știți ce moduri se combină între ele

## 4.2 Ce nu merge într-un LAN plat

Să luăm un exemplu concret. O firmă are trei departamente - contabilitate, producție și o zonă pentru vizitatori - răspândite pe trei etaje, cu câte un switch pe fiecare etaj, toate legate între ele. Fără nicio separare logică, apar simultan cinci probleme.

Tabela 4.1 Ce nu merge într-un LAN plat.

| Problemă              | Cum se manifestă                                                                                                                                                 | Ce am vrea                                             |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------|
| <b>Securitate</b>     | orice stație poate încerca să acceseze orice altă stație; broadcast-urile ajung peste tot, inclusiv la vizitatori, și pot conține informație utilă unui atacator | blocarea accesului direct între departamente           |
| <b>Eficiență</b>      | un singur broadcast este procesat de <b>toate</b> stațiile din firmă; într-o rețea cu multe switch-uri, impactul devine costisitor                               | limitarea domeniilor de broadcast                      |
| <b>Administrare</b>   | politicile se aplică per switch, dar departamentele sunt amestecate pe etaje                                                                                     | politici per departament, indiferent de locația fizică |
| <b>Calitate (QoS)</b> | telefoanele IP și videoconferința concurează cu descărcările de fișiere pentru aceeași bandă                                                                     | trafic separat, cu prioritate proprie                  |
| <b>Cost</b>           | separarea fizică ar cere switch-uri și cablare suplimentare                                                                                                      | o soluție software peste echipamentele existente       |

**De ce broadcast-urile sunt o problemă reală.** Un cadru de broadcast nu este doar „încă un cadru”. Fiecare stație din domeniu îl primește, îl urcă până la sistemul de operare și îl procesează - chiar dacă apoi decide că nu o privește. Este muncă reală, făcută de fiecare calculator, pentru fiecare broadcast. Într-o rețea cu 500 de stații, protocoalele obișnuite (ARP, DHCP, descoperirea de servicii, anunțuri de rețea) produc suficient trafic de broadcast cât să consume, permanent, o parte vizibilă din procesorul fiecărei stații. Iar dacă apare o buclă, așa cum vom vedea mai jos, rețeaua moare într-o secundă.

### 4.2.1 De ce nu pur și simplu un ruter?

Întrebarea e legitimă: știm din cursul trecut că ruterul separă domenii de broadcast. Răspunsul are patru părți:

1. Ruterele sunt **semnificativ mai scumpe** per port decât switch-urile.
2. Procesează nivelul 3, deci introduc **latență mai mare**.
3. Consumă **o interfață pentru fiecare segment**, iar ruterele au puține interfețe.
4. Cel mai important: ruterul separă după **cablare**, dar stațiile aceluiași departament pot fi răspândite fizic prin toată clădirea. Ar trebui să recablați clădirea la fiecare reorganizare.

### 4.3 Ce este un VLAN

#### Observație

**VLAN - Virtual LAN.** Un domeniu de broadcast format **doar din anumite porturi** ale unuia sau mai multor switch-uri. Un VLAN este definit prin porturile care îi aparțin și este identificat printr-un număr - **VLAN ID** - reprezentat pe 12 biți, deci între 1 și 4094.

#### Analogie

O clădire de birouri cu un singur hol mare, unde toată lumea aude tot. VLAN-urile sunt pereți despărțitori montați peste noapte: fizic e aceeași clădire, aceleași cabluri, aceleași switch-uri - dar acum fiecare departament se aude doar pe sine. Iar dacă cineva se mută la alt etaj, îl muți și pe el în „camera” lui, fără să spargi niciun zid.

Intern, fiecare switch asociază fiecărui port un VLAN ID. Pe echipamentele Cisco, toate porturile aparțin inițial VLAN-ului 1. Un port care aparține unui singur VLAN se numește **port de acces**; pentru stația conectată la el, apartenența la VLAN este complet **transparentă** - nu se configurează nimic pe calculator, iar utilizatorul nu are cum să afle.

#### Observație

**Regula care se uită cel mai des.** Un VLAN trebuie **creat** pe switch înainte de a i se atribui porturi. Și, mai important: pentru a comuta trafic aparținând VLAN-ului X, un switch trebuie să aibă VLAN-ul X configurat local - **chiar dacă traficul doar tranzitează prin el**.

Un VLAN care „nu ajunge” este, în nouă cazuri din zece, un VLAN lipsă pe un switch intermediar. Este prima verificare la orice problemă de acest fel.

### 4.4 Porturi de acces și legături trunk

Până acum am lucrat cu un singur switch. Ce se întâmplă când VLAN-urile trebuie să treacă dintr-un switch în altul?

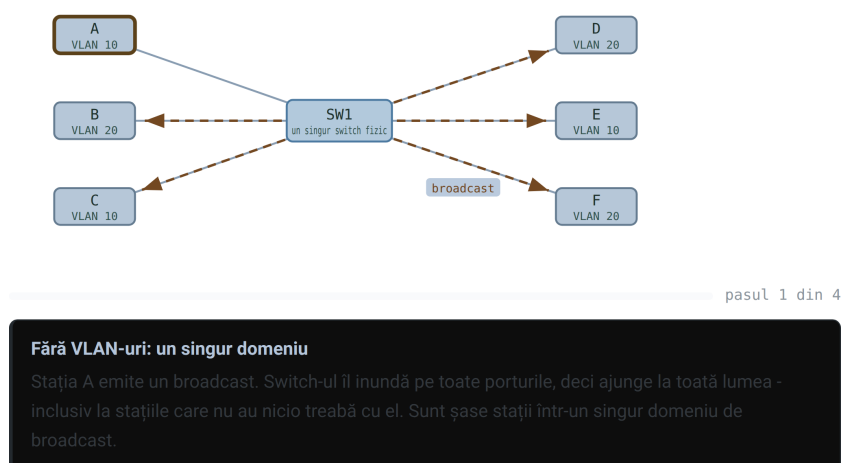


Figura 4.1 Ce face, concret, un VLAN.

#### 4.4.1 Problema

Prima idee este să folosim un cablu pentru fiecare VLAN: trei VLAN-uri, trei cabluri, trei porturi consumate la fiecare capăt. Soluția funcționează perfect - și e complet inutilizabilă. Zece VLAN-uri ar însemna zece porturi ocupate pe fiecare switch, doar pentru interconectare, plus zece cabluri de tras.

#### 4.4.2 Soluția: legătura trunk

##### Observație

**Port de acces vs. port trunk.** Un **port de acces** aparține unui singur VLAN. Cadrele care circulă pe el sunt cadre Ethernet obișnuite, fără nicio marcă. Aici se conectează stațiile.

Un **port trunk** transportă traficul **mai multor VLAN-uri** pe același fir. Fiecare cadru primește o etichetă care spune cărui VLAN îi aparține. Aici se conectează switch-urile între ele.

Ambele capete ale unei legături trunk trebuie configurate ca porturi trunk, iar setul de VLAN-uri permise pe ea este configurabil - implicit, toate.

#### 4.4.3 Eticheta 802.1Q

Rămâne întrebarea concretă: dacă switch-ul 1 trimite pe trunk un cadru din VLAN-ul 10, cum știe switch-ul 2 în ce VLAN să îl plaseze? Răspunsul este că i se inserează 4 octeți suplimentari în cadrul Ethernet, imediat după adresa sursă.

**Ce se întâmplă, de fapt, cu eticheta.** Eticheta este pusă de switch la **intrarea pe trunk** și scoasă la **ieșirea pe un port de acces**. Stațiile nu văd niciodată o etichetă și nu trebuie configurate în niciun fel.

Excepția o constituie serverele de virtualizare și telefoanele IP, care sunt configurate



Figura 4.2 Cadrul etichetat 802.1Q, câmp cu câmp.

deliberat ca porturi trunk, pentru că trebuie să pună singure trafic în mai multe VLAN-uri.

#### 4.5 VLAN-ul nativ

Fiecare legătură trunk are un VLAN special, numit **nativ**, ale cărui cadre circulă **neetichetate** - în format Ethernet obișnuit, ca și cum trunk-ul nici nu ar exista.

Motivul este istoric: compatibilitatea cu echipamente care nu înțeleg 802.1Q și care, altfel, ar arunca toate cadrele de pe trunk. Cele două capete trebuie să aibă configurat **același** VLAN nativ; altfel traficul dintr-un VLAN aterizează, tăcut și fără niciun mesaj de eroare, în alt VLAN.

##### Observație

**VLAN-ul nativ și securitatea.** Faptul că VLAN-ul nativ circulă neetichetat este exact vulnerabilitatea exploatată de atacul de tip **double tagging**, discutat în cursul 12: un atacator aflat în VLAN-ul nativ construiește un cadru cu **două** etichete, primul switch o scoate pe cea exterioară, iar a doua îl duce într-un VLAN în care nu avea voie. Recomandarea practică, valabilă în orice rețea reală: setați ca VLAN nativ un VLAN nefolosit, în care nu se află nicio stație.

##### Exemplu

Topologie: SW1 - SW0 - SW2, legate prin trunk-uri cu VLAN nativ 10. Stația A este pe SW1 în VLAN 10, stația B pe SW2 în VLAN 10, stațiile C și D pe SW0 în VLAN 20. Stația A emite un broadcast. La cine ajunge și în ce format circulă pe legături?

**Rezolvare. Ajunge doar la B.** Broadcast-ul se propagă exclusiv în VLAN-ul emițătorului, iar C și D sunt în VLAN 20. Traseul este A → SW1 → SW0 → SW2 → B. **Pe ambele legături trunk cadrul circulă neetichetat**, în format Ethernet standard - pentru că VLAN-ul 10 este chiar VLAN-ul nativ al trunk-urilor. Dacă VLAN-ul nativ ar fi fost 1, același cadru ar fi circulat etichetat 802.1Q cu VLAN ID 10. Aceasta e singura diferență, dar produce confuzie de fiecare dată la examen.

#### 4.6 Rutarea între VLAN-uri

VLAN-urile izolează - asta e treaba lor. Uneori însă chiar vrem ca departamentul de producție să ajungă la serverul contabilității. Controlat, dar să ajungă.

Fiind domenii de broadcast distincte, două VLAN-uri nu pot comunica fără un dispozitiv



de nivel 3. Există două soluții clasice și una modernă.

#### 4.6.1 Soluția clasică: o interfață per VLAN

Ruterul primește câte o interfață fizică pentru fiecare VLAN, iar fiecare interfață se conectează la un port de acces din VLAN-ul respectiv. Traficul intră pe o interfață și iese pe alta, ruterul făcând rutarea normală între rețelele lui conectate.

#### 4.6.2 Soluția router-on-a-stick

O singură interfață fizică a ruterului se leagă la un port **trunk** al switch-ului și se împarte logic în **subinterfețe**, câte una pentru fiecare VLAN. Fiecare subinterfață primește adresă IP proprie și este informată că traficul vine etichetat 802.1Q cu un anumit VLAN ID.

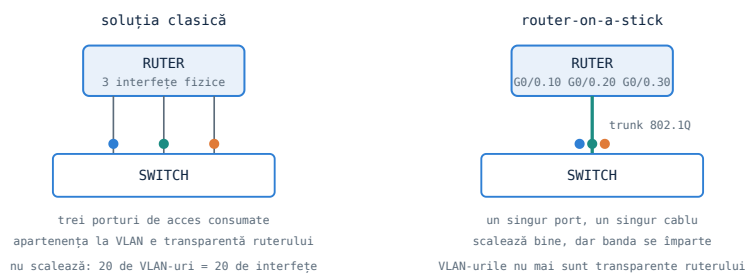


Figura 4.3 Aceeași funcție, două arhitecturi.

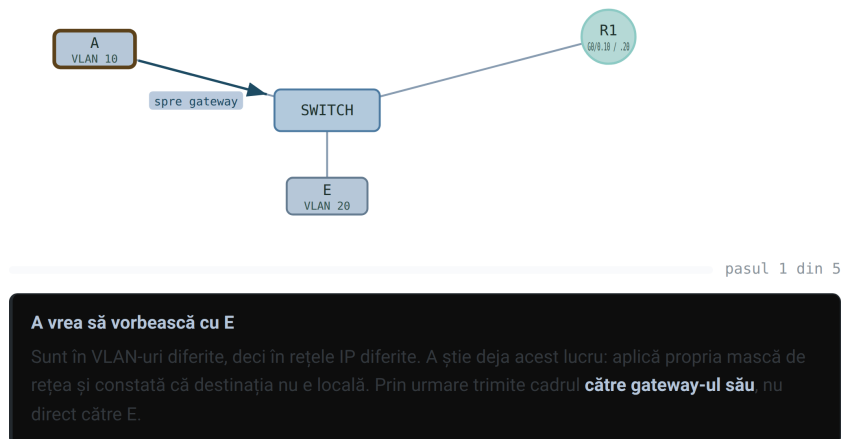


Figura 4.4 Traseul unui pachet prin router-on-a-stick.

**Router-on-a-stick.** Funcționează cu orice ruter, chiar și vechi. Un singur cablu, un singur port. Ruterul poate face simultan NAT, VPN, ACL-uri complexe. Scalează bine la numărul de VLAN-uri.

#### Switch de nivel 3 cu SVI

- Rutare în hardware, pe backplane - fără nicio gâtuire
- Nu consumă niciun cablu și niciun port

- Echipament mai scump
- Funcții de securitate mai limitate decât ale unui ruter

**Ce se folosește azi.** În rețelele moderne de întreprindere, rutarea între VLAN-uri se face aproape întotdeauna pe un **switch de nivel 3**, prin interfețe virtuale numite **SVI** (Switched Virtual Interface). Traficul nu mai părăsește niciodată switch-ul.

Router-on-a-stick rămâne relevant pentru rețele mici, pentru echipamente existente și - mai ales - pentru că este cea mai clară ilustrare a ceea ce face, de fapt, eticheta 802.1Q. Îl veți configura la laboratorul 3.

#### 4.7 Redundanța și problema buclelor

Dacă legătura dintre două switch-uri cade, stațiile din spatele lor nu mai comunică. Soluția evidentă este o legătură de rezervă. Soluția evidentă produce însă o **bucă fizică** - iar o buclă fizică într-o rețea comutată produce trei efecte simultane, toate catastrofale.

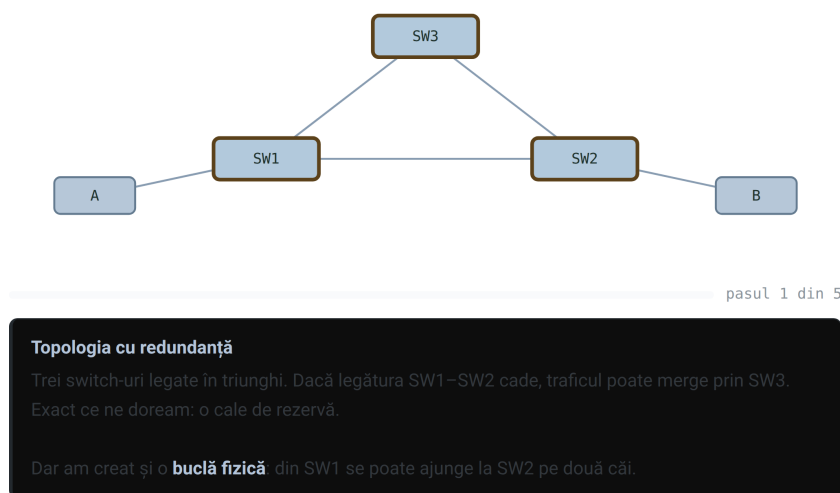


Figura 4.5 Ce se întâmplă, exact, într-o buclă de nivel 2.

**De ce nivelul 3 nu are această problemă.** Un pachet IP prins într-o buclă de rutare are câmpul **TTL** decrementat la fiecare ruter. Când ajunge la zero, pachetul e aruncat și se trimite un mesaj înapoi la sursă. Bucla există, produce trafic inutil și se vede imediat cu **traceroute** - dar rețeaua rămâne funcțională.

Antetul Ethernet nu are niciun echivalent al TTL-ului. Aceasta este singura, dar decisiva, diferență - și motivul pentru care STP a trebuit inventat.

#### 4.8 Spanning Tree: rolurile și costurile

Ideea protocolului **STP** (Spanning Tree Protocol, IEEE 802.1D) este simplă și elegantă: acceptăm bucla fizică, pentru că ne trebuie redundanța, dar **blocăm logic** unul dintre porturile buclei. Dacă legătura principală cade, portul blocat este redeschis.

Algoritmul folosit se numește **STA** (Spanning Tree Algorithm) și este, în esență, calculul unui arbore de acoperire pe graful switch-urilor - o problemă clasică de teoria grafurilor,

aplicată în timp real, de echipamente care nu se cunosc între ele.

#### 4.8.1 Cele patru roluri

Tabela 4.2 Cele patru roluri.

| Rol                         | Cine îl are                                        | Ce face portul                                                          |
|-----------------------------|----------------------------------------------------|-------------------------------------------------------------------------|
| <b>Root bridge</b>          | switch-ul cu cel mai mic Bridge ID din rețea       | toate porturile lui sunt desemnate                                      |
| <b>Root port (RP)</b>       | câte <b>unul singur</b> pe fiecare switch non-root | calea cea mai bună spre rădăcină; trimite și primește date              |
| <b>Designated port (DP)</b> | câte <b>unul singur</b> pe fiecare legătură        | trimite și primește date                                                |
| <b>Blocked port (BLK)</b>   | tot ce rămâne                                      | nu trimite și nu primește date, dar <b>continuă să asculte BPDU-uri</b> |

**De ce portul blocat continuă să asculte.** Dacă ar tăcea complet, nu ar avea cum să afle că legătura principală a căzut. Ascultând în continuare mesajele STP (numite **BPDU**), observă tăcerea de pe calea principală și se poate redeschide. Portul blocat este, cu alte cuvinte, o santinelă: nu lasă nimic să treacă, dar privește permanent și e gata să deschidă poarta.

#### 4.8.2 Bridge ID: cine devine rădăcină

Fiecare switch are un identificator unic pe 64 de biți: **16 biți de prioritate** (implicit 32768) urmați de **48 de biți de adresă MAC**. Comparăția se face lexicografic: întâi prioritatea, iar la egalitate adresa MAC.

##### Observație

**Consecința practică pe care nimeni nu o vrea.** Dacă nu configurați nimic, toate switch-urile au prioritatea 32768, deci decide **adresa MAC cea mai mică** - adică, statistic, echipamentul **cel mai vechi** din rețea.

Rezultatul tipic: rădăcina arborelui ajunge un switch de acces prăfuit dintr-un dulap, iar tot traficul dintre switch-urile moderne de magistrală trece prin el. Rețeaua funcționează, dar mult sub capacitatea ei, iar cauza e practic invizibilă dacă nu știți ce să căutați.

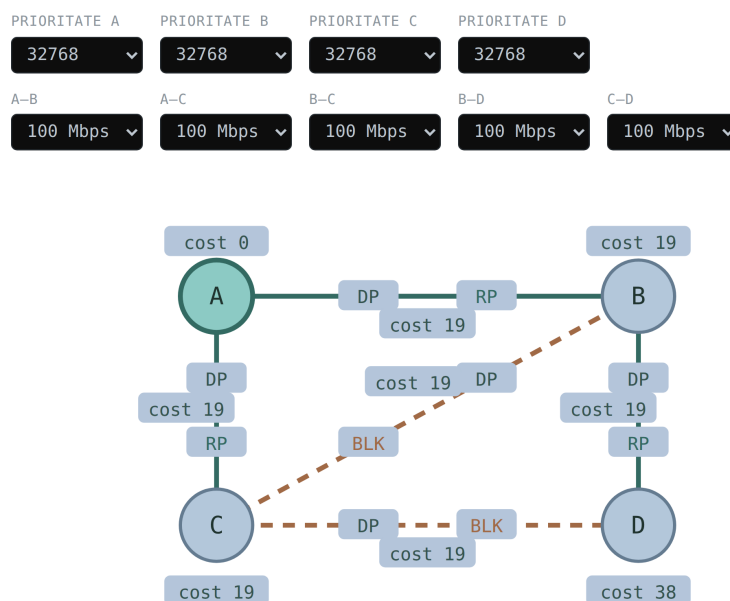
De aceea, într-o rețea reală, rădăcina se stabilește **explicit**.

#### 4.8.3 Costurile legăturilor

Costul unei căi este suma costurilor legăturilor parcurse. Observați că valorile nu sunt liniare: o legătură de 10 Mbps costă de 25 de ori mai mult decât una de 1 Gbps, deși e de 100 de ori mai lentă. Scara a fost aleasă empiric, iar la vitezele moderne a devenit prea comprimată - de aici varianta extinsă.

Tabela 4.3 Costurile legăturilor.

| Lățime de bandă | Cost STP clasic | Cost în varianta extinsă |
|-----------------|-----------------|--------------------------|
| 10 Mbps         | 100             | 2 000 000                |
| 100 Mbps        | 19              | 200 000                  |
| 1 Gbps          | 4               | 20 000                   |
| 10 Gbps         | 2               | 2 000                    |



| SWITCH | BRIDGE ID              | ROL         | COST SPRE RĂDĂCINĂ | PRIN |
|--------|------------------------|-------------|--------------------|------|
| A      | 32768 . 000A.0000.0001 | root bridge | 0                  | —    |
| B      | 32768 . 000B.0000.0002 | non-root    | 19                 | A    |
| C      | 32768 . 000C.0000.0003 | non-root    | 19                 | A    |
| D      | 32768 . 000D.0000.0004 | non-root    | 38                 | B    |

Rădăcina este switch-ul cu cel mai mic Bridge ID (prioritate, apoi adresa MAC). Pe fiecare switch fără rol de rădăcină se alege un singur port rădăcină, iar pe fiecare legătură un singur port desemnat. Ce rămâne se blochează.

Figura 4.6 Calculator STP: patru switch-uri, cinci legături.

**Experimentele de făcut aici.** Lăsați totul implicit. Cine devine rădăcină? De ce? (Uitați-vă la adresele MAC.) Coborâți prioritatea switch-ului **D** la 4096. Arborele se reorganizează complet în jurul lui - aceasta este exact metoda prin care un administrator forțează rădăcina. Reveniți la prioritățile egale și schimbați viteza legăturii **A–C** la 10 Mbps. Priviți cum se mută portul blocat: calea a devenit scumpă, deci STP alege altă rută. Faceți două legături identice ca viteză și cost. Cum se rupe egalitatea? (Prin Bridge ID.)

#### 4.9 Cei patru pași ai algoritmului

Switch-urile schimbă între ele mesaje numite **BPDU** (Bridge Protocol Data Unit), trimise o dată la 2 secunde pe toate porturile, către adresa multicast **01:80:C2:00:00:00**. Un

BPDU conține patru informații: ID-ul rădăcinii cunoscute, costul până la ea, ID-ul propriu și ID-ul portului.

1. **Alegerea rădăcinii.** Fiecare switch pornește crezând că el este rădăcina. Când primește un BPDU cu un Bridge ID mai mic, își schimbă părerea și îl propagă mai departe. După câteva secunde, toată lumea e de acord cine e rădăcina.
2. **Alegerea porturilor rădăcină.** Fiecare switch non-root alege **un singur** port: cel cu costul cumulat minim spre rădăcină. La egalitate de cost decide Bridge ID-ul vecinului, apoi Port ID-ul.
3. **Alegerea porturilor desemnate.** Pe fiecare legătură trebuie să existe exact un port desemnat: cel al switch-ului cu cost mai mic spre rădăcină; la egalitate, cel cu Bridge ID mai mic. Rădăcina are întotdeauna **numai** porturi desemnate.
4. **Blocarea restului.** Orice port care nu e nici rădăcină, nici desemnat trece în starea *blocking*.



Figura 4.7 Puneți în ordine pașii algoritmului STA.

### Exemplu

Cinci switch-uri au următoarele Bridge ID-uri. Care devine rădăcină?

| Switch | Prioritate | Adresă MAC           | Bridge ID                     |
|--------|------------|----------------------|-------------------------------|
| A      | 16384      | 00E0.A3C9.6AB8       | B327680001.97DA.86E8          |
| B      | 32768      | 000C.844D.D163840003 | E496.C80E.E81920060.2F07.EB2B |
| C      | 8192       | 00D0.BC0C.844D       | D163840003.E496.C80E          |
| D      | 16384      | 0003.E496.C80E       | E81920060.2F07.EB2B           |
| E      | 8192       | 0060.2F07.EB2B       | D163840003.E496.C80E          |

**Rezolvare.** Se compară **întâi prioritatea**. Cea mai mică este 8192, deci rămân în cursă doar **C** și **E**. Switch-ul B, deși are cea mai mică adresă MAC dintre toate, este eliminat imediat - prioritatea lui e 32768. Între C și E se compară **adresa MAC**: 00D0... față de 0060... Comparăția se face octet cu octet, de la stânga: primul octet e 00 la ambele; al doilea este D0 = 208 la C și 60 = 96 la E. Deci **E câștigă. Rădăcina este E**. Capcana clasică e să vă uitați întâi la adresa MAC - prioritatea se compară prima, întotdeauna.

Tabela 4.4 Stările porturilor și timpul de convergență.

| Stare      | Transmite date | Învăță adrese MAC | Procesează BPDU     | Durată                        |
|------------|----------------|-------------------|---------------------|-------------------------------|
| Disabled   | nu             | nu                | nu                  | portul e închis administrativ |
| Blocking   | nu             | nu                | doar primește       | până la 20 s (max age)        |
| Listening  | nu             | nu                | trimite și primește | 15 s (forward delay)          |
| Learning   | nu             | <b>da</b>         | trimite și primește | 15 s (forward delay)          |
| Forwarding | <b>da</b>      | da                | trimite și primește | stare finală                  |

#### 4.10 Stările porturilor și timpul de convergență

Un port nu trece direct din „blocat” în „funcțional”. Parcurge o serie de stări, fiecare cu rolul ei.

**De ce există starea „learning”.** Dacă portul ar trece direct în forwarding, tabela CAM ar fi goală și switch-ul ar inunda tot traficul - exact în momentul cel mai delicat, imediat după o reconfigurare a topologiei. Starea *learning* îi dă 15 secunde să își umple tabela ascultând traficul, **fără** să transmită nimic. Când trece în forwarding, știe deja unde e fiecare stație.

##### Observație

**Cincizeci de secunde de tăcere.** Convergența STP clasic poate dura până la **50 de secunde**: 20 pentru max age, plus  $2 \times 15$  pentru tranzițiile prin listening și learning. Pentru un utilizator care doar și-a pornit calculatorul și așteaptă ca portul să se activeze, aproape un minut de tăcere este inacceptabil - și generează un bilet la serviciul de asistență. De aceea au apărut variantele rapide din secțiunea următoare, plus mecanismul **PortFast**, care sare peste tranziții pe porturile unde sigur nu există alt switch.

#### 4.11 Variantele moderne de STP

**Ideea din spatele variantelor per-VLAN.** Cu un singur arbore pentru toate VLAN-urile, o legătură blocată este blocată pentru tot traficul - deci un cablu scump stă complet nefolosit.

Cu un arbore separat pentru fiecare VLAN, puteți face ca VLAN-ul 10 să folosească legătura din stânga și VLAN-ul 20 pe cea din dreapta. Ambele legături transportă trafic, redundanța rămâne, iar banda disponibilă se dublează. Este exact motivul pentru care aceste variante au apărut.

Tabela 4.5 Variantele moderne de STP.

| Variantă           | Standard | Ce aduce                                                                                         |
|--------------------|----------|--------------------------------------------------------------------------------------------------|
| <b>STP</b>         | 802.1D   | originalul; convergență până la 50 s; un singur arbore pentru toate VLAN-urile                   |
| <b>RSTP</b>        | 802.1w   | convergență în câteva secunde; roluri noi de port (alternate, backup)                            |
| <b>PVST+</b>       | Cisco    | câte un arbore <b>pentru fiecare VLAN</b> - permite folosirea simultană a legăturilor redundante |
| <b>Rapid-PVST+</b> | Cisco    | combinația celor două de mai sus; alegerea uzuală într-o rețea Cisco                             |
| <b>MSTP</b>        | 802.1s   | grupează VLAN-urile în câteva instanțe de arbore - scalează mai bine decât un arbore per VLAN    |

#### 4.12 EtherChannel: mai multe cabluri, o singură legătură

STP rezolvă buclele blocând porturi - ceea ce înseamnă că o legătură scumpă stă nefolosită. Agregarea legăturilor rezolvă aceeași problemă altfel: mai multe legături fizice sunt combinate într-o **singură interfață logică**, pe care STP o vede ca pe un singur link.

Nu mai există buclă, deci nu se blochează nimic, iar banda se însumează.

##### Analogie

Patru benzi de autostradă în loc de una. Din punctul de vedere al hărții rutiere, e tot „autostrada A1” - un singur drum. Dar capacitatea e de patru ori mai mare, iar dacă o bandă se închide pentru lucrări, traficul continuă pe celelalte, fără nicio replanificare a traseului.

**Avantaje.** Banda se adună: patru legături de 1 Gbps devin una logică de 4 Gbps Configurarea se face o singură dată, pe interfața logică Redundanță fără timpul de convergență al STP - căderea unui membru nu întrerupe legătura Distribuirea traficului între membri (load balancing)

##### Restricții

- Maximum 8 interfețe fizice într-un grup
- Toți membrii trebuie să aibă aceeași viteză, același duplex, același mod (acces sau trunk), același set de VLAN-uri permise și același VLAN nativ
- Distribuirea se face **per flux**, nu per cadru: o singură sesiune nu depășește banda unui membru

**De ce distribuirea se face per flux.** Dacă switch-ul ar trimite cadrele alternativ pe fiecare legătură, ele ar putea ajunge la destinație **în altă ordine** decât au plecat - legăturile nu au latențe perfect identice. Reordonarea ar strica performanța TCP mult mai mult decât

ar ajuta banda suplimentară. De aceea switch-ul calculează un hash din adresele sursă și destinație și trimite **toate** cadrele unei conversații pe aceeași legătură. Consecința practică: o singură copiere de fișier între două servere nu va depăși niciodată 1 Gbps pe un canal de 4 Gbps. Banda agregată ajută la **multe** conversații simultane, nu la una singură.

#### 4.12.1 Protocoalele de negociere

Tabela 4.6 Protocoalele de negociere.

|                | PAGP                                                             | LACP                           |
|----------------|------------------------------------------------------------------|--------------------------------|
| Origine        | proprietary Cisco                                                | IEEE 802.3ad, ulterior 802.1AX |
| Mod activ      | <b>desirable</b>                                                 | <b>active</b>                  |
| Mod pasiv      | <b>auto</b>                                                      | <b>passive</b>                 |
| Fără negociere | <b>on</b> - recomandat între echipamente de producători diferiți |                                |

Regula de combinare este aceeași la ambele protocoale și merită memorată prin logica ei, nu prin tabel: **cel puțin un capăt trebuie să inițieze**. Două capete pasive nu vor forma niciodată un canal, pentru că amândouă așteaptă. Iar **on** nu se combină cu nimic altceva decât cu **on**, pentru că nu trimite niciun pachet de negociere - celălalt capăt nu are ce să audă.

| COMBINAȚIE DE MODURI | REZULTAT                                            |
|----------------------|-----------------------------------------------------|
| on + on              | niciun canal - unul negociază, celălalt nu răspunde |
| on + active          | canal format - unul inițiază, celălalt răspunde     |
| active + active      | canal format - ambele inițiază                      |
| active + passive     | canal format - niciun protocol de negociere         |
| passive + passive    | niciun canal - amândouă așteaptă                    |

Apasă un element din stânga, apoi perechea lui din dreapta.

Figura 4.8 Ce combinații formează un canal agregat.

#### 4.13 Greșeli frecvente

##### Observație

##### Greșeli frecvente.

- „Am creat VLAN-ul pe switch-urile de capăt, deci e gata" Fiecare switch prin care **tranzitează** traficul trebuie să aibă VLAN-ul configurat local, chiar dacă nu are nicio stație în el. La orice VLAN care „nu ajunge", verificați întâi switch-urile



intermediare.

- **„VLAN-ul nativ nu contează, e doar o setare"** Dacă cele două capete ale unui trunk au VLAN-uri native diferite, traficul dintr-un VLAN aterizează tăcut în altul - fără niciun mesaj de eroare. Setati același VLAN nativ la ambele capete, și alegeți unul nefolosit.
- **„Am pus adresă IP pe interfața fizică a ruterului, la router-on-a-stick"** Adresele stau pe **subinterfețe**. Interfața fizică primește doar **no shutdown**. Dacă o lăsați închisă, toate subinterfețele sunt moarte, oricât de corect ar fi configurate. Fizică: doar activată. Logică: adresă și încapsulare.
- **„STP încetinește rețeaua, hai să îl dezactivăm"** Cea mai costisitoare decizie posibilă. Fără STP, prima buclă accidentală - un cablu pus greșit de cineva grăbit - oprește complet rețeaua. Nu dezactivați STP. Treceți pe Rapid-PVST+ și puneți PortFast pe porturile de acces.
- **„Am legat două switch-uri cu patru cabluri, deci am 4 Gbps"** Fără EtherChannel configurat, STP va bloca trei dintre ele - aveți 1 Gbps și trei cabluri decorative. Agregarea trebuie configurată explicit, pe ambele capete.

#### 4.14 Rezumat și glosar

1. **VLAN-ul** împarte un switch fizic în mai multe domenii de broadcast, definite prin porturi și independente de cablare.
2. **Trunk-ul** transportă mai multe VLAN-uri pe un singur cablu, folosind eticheta 802.1Q. VLAN-ul nativ circulă neetichetat - și de aceea trebuie ales cu grijă.
3. Comunicația între VLAN-uri cere un dispozitiv de **nivel 3**: ruter cu subinterfețe sau switch de nivel 3 cu SVI.
4. O buclă la nivelul 2 este **fatală**, pentru că lipsește TTL-ul. **STP** o rezolvă blocând logic un port.
5. **EtherChannel** transformă mai multe cabluri într-o singură legătură logică - bandă însumată, fără buclă și fără blocare.

#### 4.15 Ce urmează și unde citiți mai departe

Am segmentat rețeaua locală și am făcut-o rezistentă la defecte. Ce nu am rezolvat este comunicația între rețele diferite, la scară globală - iar pentru asta adresele MAC, plate și locale, nu sunt de niciun folos. Cursul următor introduce adresarea ierarhică și nivelul 3.

La **laboratorul 3** configurați exact VLAN-urile, trunk-ul și router-on-a-stick de aici, apoi provocați o buclă și cronometrați cât durează convergența STP.

- IEEE 802.1Q - VLAN-uri și etichetarea cadrelor
- IEEE 802.1D și 802.1w - Spanning Tree și varianta rapidă
- IEEE 802.1s - Multiple Spanning Tree

- IEEE 802.1AX - agregarea legăturilor
- Cisco - ghidurile de configurare pentru VLAN, STP și EtherChannel din documentația Catalyst

### Întrebări de verificare

1. Cate VLAN-uri pot fi identificate cu campul VLAN ID din eticheta 802.1Q?
  - a) 256
  - b) 1024
  - c) 4094 utilizabile din 4096 posibile
  - d) 65536
2. Ce se intampla cu un cadru din VLAN-ul nativ cand traverseaza un trunk?
  - a) Primeste eticheta 802.1Q ca oricare altul
  - b) Circula neetichetat, in format Ethernet standard
  - c) Este aruncat
  - d) Este duplicat pe toate VLAN-urile
3. De ce o bucla la nivelul 2 este mult mai grava decat una la nivelul 3?
  - a) Pentru ca switch-urile sunt mai lente decat ruterele
  - b) Pentru ca antetul de nivel 2 nu are camp TTL, deci nimic nu opreste cadrul
  - c) Pentru ca adresele MAC sunt mai scurte decat cele IP
  - d) Pentru ca nivelul 2 nu foloseste broadcast
4. Cate porturi radacina are switch-ul care este root bridge?
  - a) Unul
  - b) Cate unul pe fiecare legatura
  - c) Niciunul - toate porturile lui sunt desemnate
  - d) Depinde de prioritate
5. Doua switch-uri au ambele porturile LACP in modul passive. Se formeaza EtherChannel?
  - a) Da, LACP negociaza automat
  - b) Nu, cel putin un capat trebuie sa fie active
  - c) Da, dar doar pentru VLAN-ul nativ
  - d) Doar daca vitezele difera

6. Ce se compara primul, la alegerea root bridge-ului?
  - a) Adresa MAC
  - b) Prioritatea
  - c) Numarul de porturi
  - d) Viteza legaturilor
7. La ce foloseste starea learning din STP?
  - a) Pentru a invata topologia STP
  - b) Pentru a-si umple tabela CAM inainte de a incepe sa transmita
  - c) Pentru a negocia viteza portului
  - d) Pentru a alege VLAN-ul nativ
8. O singura copie de fisier intre doua servere, peste un EtherChannel de 4 Gbps. Ce debit maxim obtine?
  - a) 4 Gbps
  - b) 2 Gbps
  - c) 1 Gbps, pentru ca distribuirea se face per flux
  - d) Depinde de VLAN

## 5. Protocolul IP

---

---

Din cursurile trecute  
De ce mai avem nevoie de un set de adrese  
Cele trei funcții ale nivelului rețea  
Antetul IPv4  
Adresa, masca și cele două operații  
Clase, adrese private, adrese rezervate  
Subnetarea  
VLSM: subnetarea optimă  
ARP: de la adresa IP la adresa MAC  
DHCP: adrese fără configurare manuală  
Greșeli frecvente  
Rezumat și glosar  
Ce urmează și unde citiți mai departe  
Întrebări de verificare

---

---

Adresele MAC sunt unice pe planetă și totuși inutilizabile la scară mare, pentru că nu spun nimic despre locul unde se află echipamentul. Nivelul rețea rezolvă problema exact așa cum a rezolvat-o poșta acum două secole: printr-o adresă ierarhică, din care se poate deduce direcția fără a cunoaște destinația finală.

### 5.1 Din cursurile trecute

**Ce trebuie să aveți în minte.** Adresa MAC are 48 de biți, e unică pe planetă, dar e **plată** (fără ierarhie) și **locală** (vizibilă doar în domeniul de broadcast). Un switch comută după adresa MAC; un **ruter** lucrează la nivelul 3 și separă domeniile de broadcast. Un **VLAN** este un domeniu de broadcast. Comunicația între VLAN-uri cere obligatoriu un dispozitiv de nivel 3. Conversia binar  $\leftrightarrow$  zecimal din cursul 1 devine, de azi, unealta de lucru zilnică.

#### 5.1.1 Ce veți putea face după acest curs

**Ce veți putea face după acest capitol.**

- Să explicați de ce adresarea ierarhică face rutarea posibilă la scara Internetului
- Să citiți antetul IPv4 și să spuneți la ce folosește fiecare câmp important
- Să calculați adresa de rețea și adresa de broadcast prin operații pe biți

- Să subnetați un bloc în subrețele egale și, apoi, optim cu VLSM
- Să distingeți adresele publice de cele private și să recunoașteți blocurile rezervate
- Să descrieți un schimb ARP complet și să explicați ce e proxy ARP
- Să urmăriți cele patru mesaje DHCP și să spuneți când e nevoie de un relay

## 5.2 De ce mai avem nevoie de un set de adrese

Adresa MAC identifică unic o interfață în toată lumea. Pare exact ce ne trebuie - și totuși nu este suficientă. Motivul e o singură proprietate: schema este **plată**.

**Ce înseamnă „plată” și de ce e o problemă.** Din adresa 00:FC:42:3E:34:99 nu se poate deduce absolut nimic despre locul unde se află acea placă de rețea. Poate fi în camera de alături sau în Australia. Nu există nicio regulă care să spună „adresele care încep cu X sunt în Europa”. Consecința: un ruter care ar dirija trafic după adrese MAC ar avea nevoie de **o intrare separată pentru fiecare interfață din lume** - miliarde de rânduri, fără nicio posibilitate de grupare. Tabela nu ar încăpea nicăieri, nu ar putea fi actualizată de nimeni și nu ar putea fi căutată suficient de repede.

### Analogie: codul poștal

Un poștaș din Cluj primește un plic pentru „Splaiul Independenței 313, sector 6, București”. Nu are nevoie să știe unde e cea stradă. Îi ajunge prima cifră a codului poștal ca să pună plicul în sacul pentru București. Cine ia sacul mai departe folosește a doua cifră, și așa mai departe.

Fiecare nivel de decizie folosește

### doar prefixul de care are nevoie

. Exact acesta este mecanismul care face rutarea IP posibilă la scara Internetului.



Figura 5.1 Codul poștal este o adresă ierarhică. Aceeași idee, aplicată la 32 de biți, dă adresa IP.

## 5.3 Cele trei funcții ale nivelului rețea

**Ce înseamnă „best-effort”.** IP promite un singur lucru: va **încerca** să livreze pachetul. Nu promite că va reuși, nu promite ordinea și nu vă anunță dacă a eșuat. Un pachet poate fi aruncat de orice ruter aglomerat de pe traseu, fără nicio ceremonie. Pare o slăbiciune. Este, de fapt, decizia de proiectare care a făcut Internetul posibil: rămânând simplu și fără stare, un ruter poate procesa milioane de pachete pe secundă fără să țină minte nimic despre ele. Complexitatea - retransmisii, ordonare, confirmări - a fost mutată la capete, în TCP.

Tabela 5.1 Cele trei funcții ale nivelului rețea.

| Funcție                | Ce înseamnă concret                                                                                                                                    |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| Adresare globală       | fiecare dispozitiv este identificat unic printr-o adresă IP, organizată ierarhic, astfel încât adresele apropiate să poată fi grupate                  |
| Comunicație end-to-end | protocolul este de tip <b>best-effort</b> : nu stabilește conexiuni, nu garantează livrarea, nu garantează ordinea. Garanțiile sunt treaba nivelului 4 |
| Rutare (dirijare)      | echipamente intermediare aleg calea fiecărui pachet pe baza adresei destinație                                                                         |

Tabela 5.2 Cele trei funcții ale nivelului rețea.

| Protocol | Rol                                                                      |
|----------|--------------------------------------------------------------------------|
| IPv4     | protocolul de bază; subiectul acestui curs                               |
| IPv6     | succesorul, cu adrese de 128 de biți; cursul 9                           |
| ICMP     | mesaje de control și eroare: ping, traceroute, „destinație inaccesibilă" |
| IGMP     | gestionarea grupurilor multicast                                         |

## 5.4 Antetul IPv4

IPv4 este definit în RFC 791, publicat în 1981 - și, cu excepții minore, funcționează neschimbat de atunci. Unitatea de date rezultată din încapsularea IP se numește **pachet**.



Figura 5.2 Antetul IPv4, câmp cu câmp.

**Cum funcționează traceroute - un truc elegant.** Trimiteți un pachet cu **TTL = 1**. Primul ruter îl decrementează la 0, îl aruncă și vă trimite un mesaj ICMP *time exceeded* - în care își pune propria adresă. Ați aflat cine e primul salt.

Trimiteți apoi cu **TTL = 2**, apoi 3, apoi 4... Fiecare pachet moare cu un salt mai departe și fiecare ruter se prezintă singur. Așa se construiește lista completă a traseului, folosind un câmp care fusese gândit ca protecție împotriva buclelor.

## 5.5 Adresa, masca și cele două operații

O adresă IPv4 are 32 de biți, scriși de obicei în format zecimal cu punct - patru numere între 0 și 255, câte unul pentru fiecare octet.

141 . 85 . 241 . 139

10001101

.

01010101

.

11110001

.

10001011

Adresa se împarte în două părți: **partea de rețea** și **partea de host**. Dispozitivele care au aceeași parte de rețea se află în aceeași rețea și pot comunica **direct**, fără ruter.

Granița dintre cele două părți nu se poate deduce din adresă - trebuie spusă separat, prin **masca de rețea**: un șir continuu de 1 urmat de un șir continuu de 0.

**Operațiile ȘI și SAU pe biți, în treizeci de secunde.** Nu au nimic misterios. Se aplică **bit cu bit**, pe poziții corespunzătoare: ABA ȘI BA SAU B 0000 0101 1001 1111 De reținut, practic: **ȘI cu 1 păstrează bitul, ȘI cu 0 îl șterge. SAU cu 1 aprinde bitul, SAU cu 0 îl lasă în pace.** Atât vă trebuie pentru tot restul cursului.

### Observație

**Cele două operații fundamentale. Adresa de rețea** = adresa IP **ȘI** masca. Masca are 1 pe partea de rețea, deci acei biți se păstrează, și 0 pe partea de host, deci acei biți se șterg. Rezultatul: toți biții de host puși pe 0.

**Adresa de broadcast** = adresa IP **SAU** inversa măștii. Inversa are 0 pe partea de rețea (nu schimbă nimic) și 1 pe partea de host (aprinde tot). Rezultatul: toți biții de host puși pe 1.

Niciuna dintre ele nu poate fi atribuită unei stații.

### 5.5.1 Notăția CIDR

Notăția zecimală a măștii - 255.255.255.0 - e greoaie. S-a introdus prescurtarea **CIDR**: se scrie pur și simplu **numărul de biți de 1** din mască, precedat de bară.

Tabelul acesta merită învățat - nu pe de rost, ci prin logică. Fiecare rând înjumătățește numărul de adrese față de cel de deasupra, iar „utilizabile” e mereu „totale minus 2” (rețea și broadcast).

**Experimentele de făcut aici.** Schimbați prefixul de la /26 la /24 și priviți cum biții trec dintr-o culoare în alta. Mutarea graniței cu **un singur bit** dublează numărul de stații și înjumătățește numărul de subrețele. Puneți adresa 192.168.10.64 cu prefix /26. Observați că adresa introdusă este chiar adresa de rețea - deci nu poate fi dată unei stații. Încercați /31 și /32. Sunt cazuri speciale: /31 se folosește pe legături punct-la-punct între rutere, /32

Tabela 5.3 Notăția CIDR.

| Prefix | Mască zeci-<br>mală | Biți de host | Adrese totale | Adrese utiliza-<br>bile |
|--------|---------------------|--------------|---------------|-------------------------|
| /24    | 255.255.255.0       | 8            | 256           | 254                     |
| /25    | 255.255.255.128     | 7            | 128           | 126                     |
| /26    | 255.255.255.192     | 6            | 64            | 62                      |
| /27    | 255.255.255.224     | 5            | 32            | 30                      |
| /28    | 255.255.255.240     | 4            | 16            | 14                      |
| /29    | 255.255.255.248     | 3            | 8             | 6                       |
| /30    | 255.255.255.252     | 2            | 4             | 2                       |

ADRESĂ IP: 192.168.10.77  
 PREFIX /: 26  
  
 11000000.10101000.00001010.01001101  
 mască 255.255.255.192 (/26)  
 adresă de rețea 192.168.10.64  
 adresă de broadcast 192.168.10.127  
 interval utilizabil 192.168.10.65 – 192.168.10.126  
 adrese totale / utilizabile 64 / 62  
 wildcard 0.0.0.63  
  
 ■ biți de rețea ■ biți de host

Figura 5.3 Calculator de subrețea.

pentru o singură adresă (o interfață de loopback).

### Exemplu

Trei stații sunt conectate la același switch: A cu 192.168.17.64/26, B cu 192.168.17.35/26, C cu 192.168.17.31/27. Există probleme?

**Rezolvare.** Da, **trei** probleme distincte: **A** are configurată o **adresă de rețea**. 64 în binar este 01000000, iar cu /26 ultimii 6 biți sunt de host și sunt toți zero. O adresă de rețea nu poate fi atribuită unei interfețe. **C** are configurată o **adresă de broadcast**. Cu /27, rețeaua lui este 192.168.17.0/27, iar broadcast-ul acesteia este exact .31. **C** mai are și o **mască diferită** de a celorlalte. Chiar și cu adrese corecte, A și B s-ar considera în rețeaua 192.168.17.0/26, iar C în 192.168.17.0/27 - comunicația ar funcționa asimetric și derutant. Verificați fiecare caz în calculatorul de mai sus. Introduceți adresa și prefixul și comparați cu ce afișează.

## 5.6 Clase, adrese private, adrese rezervate

Istoric, adresele erau împărțite în cinci clase identificate după primii biți, fiecare cu masca ei implicită și fixă.

**De ce s-a renunțat la clase.** Sistemul era rigid până la absurd. O organizație care avea nevoie de 300 de adrese primea fie o clasă C cu 254 - insuficient - fie o clasă B cu **65 534**, din care irosea 65 000.



Tabela 5.4 Clase, adrese private, adrese rezervate.

| Clasă | Primii biți | Interval                    | Mască implicată | Statut azi              |
|-------|-------------|-----------------------------|-----------------|-------------------------|
| A     | 0...        | 0.0.0.0 – 127.255.255.255   | /8              | istoric                 |
| B     | 10...       | 128.0.0.0 – 191.255.255.255 | /16             | istoric                 |
| C     | 110...      | 192.0.0.0 – 223.255.255.255 | /24             | istoric                 |
| D     | 1110...     | 224.0.0.0 – 239.255.255.255 | -               | <b>multicast, în uz</b> |
| E     | 1111...     | 240.0.0.0 – 255.255.255.255 | -               | rezervat, experimental  |

Înlocuitorul se numește **CIDR** (Classless Inter-Domain Routing) și face un lucru simplu: masca poate fi orice, nu doar /8, /16 sau /24. Pentru 300 de adrese primiți un /23, cu 510 adrese utilizabile.

Clasele au dispărut din echipamente, dar au rămas în vocabular. Când cineva spune „o rețea de clasă C”, înțelegeți „un /24” - și nu vă mirați.

### 5.6.1 Adrese private

RFC 1918 a alocat trei spații pentru uz privat. Ele **nu pot fi rutate în Internet**, deci pot fi refolosite de oricine - și sunt, în orice locuință și în orice firmă din lume. Ieșirea în Internet se face prin translatare (NAT), subiectul cursului 9.

Tabela 5.5 Adrese private.

| Bloc privat    | Dimensiune              | Unde apare de obicei              |
|----------------|-------------------------|-----------------------------------|
| 10.0.0.0/8     | 16,7 milioane de adrese | rețele mari de întreprindere      |
| 172.16.0.0/12  | 1 milion de adrese      | rețele medii                      |
| 192.168.0.0/16 | 65 536 de adrese        | rețele mici, echipamente de acasă |

### 5.6.2 Adrese cu semnificație specială

**Un simptom care spune totul.** Dacă o stație are o adresă din 169.254.x.x, nu are rost să căutați problema în altă parte: **serverul DHCP nu a răspuns**. Fie nu există, fie cablul e în alt VLAN, fie relay-ul nu e configurat, fie rezervorul de adrese s-a epuizat.

Tabela 5.6 Adrese cu semnificație specială.

| Bloc            | Ce înseamnă                                                                                                  |
|-----------------|--------------------------------------------------------------------------------------------------------------|
| 127.0.0.0/8     | <b>Loopback.</b> 127.0.0.1 se referă la calculatorul însuși. Un ping aici testează stiva locală, nu rețeaua. |
| 169.254.0.0/16  | <b>APIPA.</b> Adresa pe care și-o dă singură o stație când serverul DHCP nu răspunde.                        |
| 0.0.0.0         | „această rețea” sau „orice rețea”, după context. Ca destinație într-o rută, înseamnă ruta implicită.         |
| 255.255.255.255 | Broadcast limitat - nu trece niciodată de un ruter.                                                          |
| 224.0.0.0/4     | Multicast. 224.0.0.5 = toate ruterele OSPF, 224.0.0.6 = DR și BDR.                                           |

## 5.7 Subnetarea

### Observație

**Subnetare.** Împărțirea unui bloc de adrese în blocuri mai mici, care să acopere un set de cerințe. Se face „împrumutând” biți din partea de host și adăugându-i la partea de rețea.

Într-o rețea modernă, un „subnet” nu e cu nimic diferit de o rețea; termenul indică doar că face parte din spațiul unei rețele mai mari.

**De ce am vrea să împărțim.** Ați primit blocul 192.168.10.0/24 - 254 de adrese utilizabile. Aveți trei departamente. Ați putea pune toate calculatoarele într-o singură rețea plată, dar atunci reveniți la toate problemele din cursul 4: un singur domeniu de broadcast, nicio separare, nicio politică diferențiată. Subnetarea vă dă trei rețele separate, fiecare cu gateway-ul ei, între care traficul trece controlat, printr-un ruter.

### 5.7.1 Varianta simplă: subrețele egale

Se împrumută atâția biți câți sunt necesari pentru numărul dorit de subrețele. Pentru trei subrețele avem nevoie de 2 biți, pentru că  $2^2 = 4 \geq 3$ .

**Exemplu:** 192.168.10.0/24 pentru trei rețele cu 60, 30 și 15 stații.

rețeaua inițială

11000000.10101000.00001010

.

00000000

/24

subrețeaua 1

11000000.10101000.00001010.00

000000

/26 → 192.168.10.0

subrețeaua 2

11000000.10101000.00001010.01

000000

/26 → 192.168.10.64

subrețeaua 3

11000000.10101000.00001010.10

000000

/26 → 192.168.10.128

(subrețeaua 4)

11000000.10101000.00001010.11

000000

/26 → 192.168.10.192

Fiecare /26 oferă 62 de adrese utilizabile, deci toate trei cerințele sunt acoperite. Risipa e însă considerabilă:

Tabela 5.7 Varianta simplă: subrețele egale.

| Cerință              | Alocat              | Utilizabile | Risipite |
|----------------------|---------------------|-------------|----------|
| 60 stații            | /26                 | 62          | 2        |
| 30 stații            | /26                 | 62          | 32       |
| 15 stații            | /26                 | 62          | 47       |
| <b>Total risipit</b> | <b>81 de adrese</b> |             |          |

### Exemplu

De ce pentru 15 stații nu ajunge un /28, care oferă 14 adrese utilizabile?

**Rezolvare.** Pentru că  $14 < 15$ . Trebuie mers la /27, care oferă 30. Și mai important: la cele 15 stații trebuie adăugat și **gateway-ul**, care consumă și el o adresă utilizabilă. Deci cerința reală e 16, iar /28 e cu atât mai insuficient. Aceasta e greșeala numărul unu la examen: se uită gateway-ul.

## 5.8 VLSM: subnetarea optimă

**VLSM** (Variable Length Subnet Mask) renunță la constrângerea ca toate subrețelele să aibă aceeași mască. Fiecare subrețea primește exact cât îi trebuie.

**Regula de aur a VLSM.** Alocarea se face **de la cea mai mare cerință spre cea mai mică**. Ordinea nu e o convenție estetică - e obligatorie.

Motivul: un bloc de dimensiune  $2^n$  trebuie să înceapă la o adresă **aliniată** la  $2^n$ . Dacă alocați mai întâi câteva blocuri mici, alinierea se pierde, iar următorul bloc mare nu mai poate începe decât mult mai departe - spațiul dintre ele devine inutilizabil.

Aceleași cerințe ca mai sus, rezolvate cu VLSM. Calculul biților de host necesari:

$$1. \text{ 60 de stații} + \text{gateway} = 61 \rightarrow 2^6 - 2 = 62 \geq 61 \rightarrow \text{6 biți de host} \rightarrow /26$$

2. **30 de stații** + gateway = 31  $\rightarrow 2^5 - 2 = 30 < 31 \rightarrow$  nu ajunge;  $2^6 - 2 = 62 \rightarrow /26$ .

Dacă cerința e strict 30 *inclusiv* gateway-ul, /27 este suficient.

3. **15 stații** + gateway = 16  $\rightarrow 2^5 - 2 = 30 \geq 16 \rightarrow$  **5 biți**  $\rightarrow /27$

Tabela 5.8 VLSM: subnetarea optimă.

| Cerință              | Subrețea alocată       | Adrese utilizabile | Risipă |
|----------------------|------------------------|--------------------|--------|
| 60 stații            | 192.168.10.0/26        | 62                 | 2      |
| 30 stații            | 192.168.10.64/27       | 30                 | 0      |
| 15 stații            | 192.168.10.96/27       | 30                 | 15     |
| <b>Total risipit</b> | <b>17 în loc de 81</b> |                    |        |

În plus, rămâne liber tot blocul de la 192.168.10.128 în sus - 128 de adrese pentru extinderi viitoare, față de niciuna în varianta cu subrețele egale.

| BLOC DE ADRESE  |                   | STAȚII NECESARE (SEPARATE PRIN VIRGULĂ) |                |        |
|-----------------|-------------------|-----------------------------------------|----------------|--------|
| 172.18.240.0/23 |                   | 200,90,20,20,6,4,4,4                    |                |        |
| CERINȚĂ         | SUBREȚEA          | INTERVAL UTILIZABIL                     | BROADCAST      | RISIPĂ |
| 200 stații      | 172.18.240.0/24   | 172.18.240.1 – 172.18.240.254           | 172.18.240.255 | 54     |
| 90 stații       | 172.18.241.0/25   | 172.18.241.1 – 172.18.241.126           | 172.18.241.127 | 36     |
| 20 stații       | 172.18.241.128/27 | 172.18.241.129 – 172.18.241.158         | 172.18.241.159 | 10     |
| 20 stații       | 172.18.241.160/27 | 172.18.241.161 – 172.18.241.190         | 172.18.241.191 | 10     |
| 6 stații        | 172.18.241.192/29 | 172.18.241.193 – 172.18.241.198         | 172.18.241.199 | 0      |
| 4 stații        | 172.18.241.200/29 | 172.18.241.201 – 172.18.241.206         | 172.18.241.207 | 2      |
| 4 stații        | 172.18.241.208/29 | 172.18.241.209 – 172.18.241.214         | 172.18.241.215 | 2      |
| 4 stații        | 172.18.241.216/29 | 172.18.241.217 – 172.18.241.222         | 172.18.241.223 | 2      |

spațiu consumat **480 / 512 adrese**  
 adrese de stație risipite **116**  
 rămâne liber de la **172.18.241.224**

Alocarea se face de la cea mai mare cerință spre cea mai mică – singura ordine care nu lasă găuri în spațiul de adrese.

Figura 5.4 Împărțitor VLSM - verificați-vă calculele.

### Exemplu

Subnetati optim 172.18.240.0/23 pentru: o rețea cu 200 de stații, una cu 90, două cu 20, una cu 6 și trei cu 4 stații.

**Rezolvare. Pasul 1 - ordonăm descrescător:** 200, 90, 20, 20, 6, 4, 4, 4. **Pasul 2 - calculăm biții de host pentru fiecare:** CerințăNecesar cu gatewayBiți

de hostPrefixSubrețeaBroadcast 2002018 → 254 utilizabile/24172.18.240.0172.18.240.255  
 90917 → 126/25172.18.241.0172.18.241.127 20215 → 30/27172.18.241.128172.18.241.159  
 20215 → 30/27172.18.241.160172.18.241.191 673 → 6/29172.18.241.192172.18.241.199  
 453 → 6/29172.18.241.200172.18.241.207 453 → 6/29172.18.241.208172.18.241.215 453  
 → 6/29172.18.241.216172.18.241.223 **Pasul 3 - verificarea:** fiecare subrețea începe  
 exact acolo unde s-a terminat cea dinainte (broadcast + 1). Dacă undeva există o gaură  
 sau o suprapunere, ați greșit calculul. **Rămâne liber** blocul de la 172.18.241.224 în  
 sus - 32 de adrese.

**Legăturile dintre rutere.** Pentru o legătură punct-la-punct între două rutere folosiți /30 - două adrese utilizabile, exact cât trebuie. Pe echipamentele care îl suportă, /31 este și mai eficient: două adrese, fără rețea și broadcast.

Un /24 alocat unei legături între două rutere risipește 252 de adrese pentru absolut nimic. Este risipa clasică din proiectele de începător.

## 5.9 ARP: de la adresa IP la adresa MAC

Iată o problemă practică pe care nu am rezolvat-o încă. O stație vrea să trimită un pachet și cunoaște adresa IP a destinației. Dar cadrul Ethernet în care trebuie încapsulat pachetul cere o **adresă MAC** - și pe aceea nu o are.

Protocolul care face traducerea se numește **ARP** (Address Resolution Protocol).

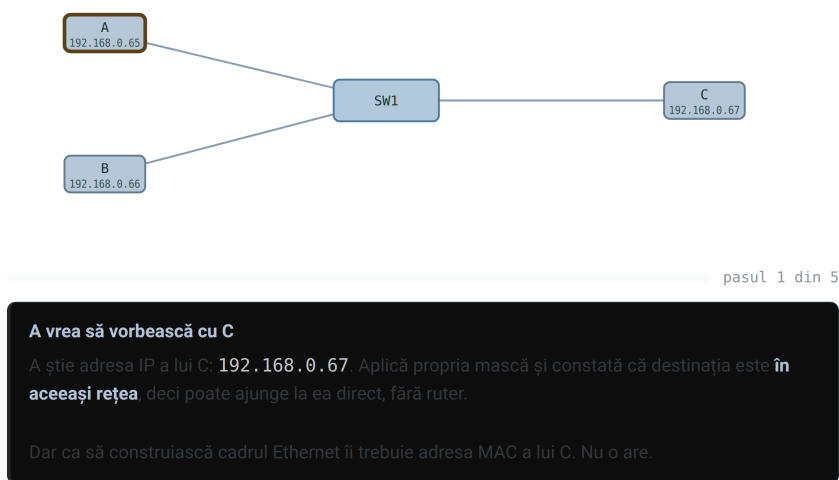


Figura 5.5 Un schimb ARP complet.

### 5.9.1 Ce se întâmplă când destinația e în altă rețea

Aici e partea care produce cea mai multă confuzie la început.

1. Sursa verifică dacă destinația e în aceeași rețea, aplicând propria mască peste ambele adrese.

2. Dacă **da** - emite ARP pentru adresa IP a destinației, ca mai sus.
3. Dacă **nu** - emite ARP pentru adresa IP a **propriului default gateway**. Pentru că acolo trebuie să ajungă cadrul, fizic.

**Regula de aur a celor două adrese.** Adresele **IP** sursă și destinație rămân **neschimbate** de la un capăt la altul al comunicației.

Adresele **MAC** sursă și destinație se **rescriu pe fiecare segment** Ethernet parcurs.

Consecință practică: dacă vedeți într-o captură un pachet cu adresa IP sursă din altă rețea și adresa MAC sursă a ruterului local, totul este perfect normal. Dacă vedeți adresa **IP** sursă rescrisă, atunci acolo se face NAT - subiectul cursului 9.

#### Observație

**Proxy ARP.** Un ruter poate răspunde cu **propria sa adresă MAC** la o cerere ARP pentru o adresă IP din afara rețelei emițătorului, dacă are o rută spre acea destinație. Stația crede că vorbește direct cu destinația, dar cadrele ajung, de fapt, la ruter. Mecanismul salvează configurații greșite - de exemplu o stație fără gateway configurat - dar maschează probleme reale și de aceea este dezactivat implicit pe multe echipamente moderne.

Câte tabele ARP are un ruter? Câte **una pentru fiecare interfață multiacces activă**. Un switch, în schimb, are **zero**: el nu se uită la adrese IP, deci nu are ce traduce. Excepția e adresa IP de management a switch-ului, care aparține interfeței virtuale, nu funcției de comutare.

## 5.10 DHCP: adrese fără configurare manuală

Configurarea manuală a adreselor este realistă pentru rutere și servere. Pentru sutele de laptopuri și telefoane care intră și ies dintr-o rețea în fiecare zi, este imposibilă.

**DHCP** (Dynamic Host Configuration Protocol) automatizează procesul prin patru mesaje, ușor de memorat sub acronimul **DORA**.

### 5.10.1 DHCP relay

Mesajul Discover este un broadcast către 255.255.255.255, iar ruterele **nu propagă broadcast-uri** - asta le e treaba. Într-o rețea cu mai multe VLAN-uri și un singur server DHCP central, cererea nu ar ajunge niciodată la el.

Soluția este configurarea unui **DHCP relay** pe ruterul din rețeaua locală: acesta preia broadcast-ul, îl transformă într-un unicast către adresa serverului și îi atașează informația despre rețeaua din care provine - ca serverul să știe din ce rezervor să aloce.

Exemplu 5.1 DHCP relay pe interfața dinspre clienți copiază

```
1
2 R1(config)# interface gigabitEthernet 0/0.20
3 R1(config-subif)# ip helper-address 10.0.100.5
```

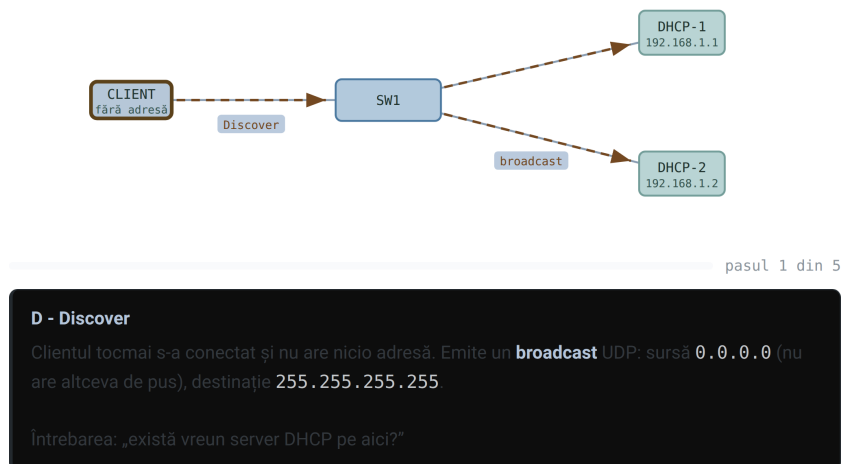


Figura 5.6 DORA: cele patru mesaje DHCP.

```
PC> ?
Comenzi disponibile aici:
ipconfig
ipconfig /all
ipconfig /release
ipconfig /renew
arp -a
arp -d
ping 192.168.1.1
ping 192.168.2.10
tracert 192.168.2.10
```

PC>

Comenzile de mai jos sunt cele pe care le veți rula la laborator, pe o stație. Încercați-le pe rând.

Figura 5.7 Terminal: comenzile de diagnostic pe o stație.

## 5.11 Greșeli frecvente

### Observație

#### Greșeli frecvente.

- **„Am uitat gateway-ul din calculul de adrese"** Gateway-ul consumă și el o adresă utilizabilă. Pentru 30 de stații **plus** gateway vă trebuie un /26, nu un /27. Cerința reală = numărul de stații + 1.
- **„Am alocat subrețelele VLSM în ordinea din enunț"** Ordinea trebuie să fie **descrescătoare după dimensiune**. Altfel alinierea se pierde și blocurile mari nu mai încap. Ordonati întotdeauna înainte de a alocă.
- **„Măștile pot fi diferite pe stațiile aceluiași segment"** Nu produc o eroare vizibilă la configurare, dar produc comportament asimetric: o stație crede că destinația e locală, cealaltă crede că e la distanță. Aceeași mască pe toate stațiile unui segment, fără excepție.
- **„Stația trimite ARP pentru adresa IP a destinației finale"** Doar dacă

destinația e în aceeași rețea. Altfel trimite ARP pentru **gateway**, pentru că acolo trebuie să ajungă fizic cadrul. Întâi se aplică masca, apoi se decide pe cine se întreabă.

- „**169.254.x.x e o adresă validă, deci rețeaua merge**” Este semnul clar că DHCP-ul nu a răspuns. Două stații cu astfel de adrese se pot chiar vedea între ele, ceea ce sporește confuzia. Vedeți  $169.254 \rightarrow$  căutați serverul DHCP sau relay-ul, nimic altceva.

## 5.12 Rezumat și glosar

1. Adresa IP e **ierarhică**, iar asta face rutarea posibilă la scară globală - spre deosebire de adresa MAC, care e plată.
2. Masca separă partea de rețea de partea de host. **ȘI cu masca** dă adresa de rețea; **SAU cu inversa** dă broadcast-ul.
3. **VLSM** alocă fiecărei subrețele exact cât îi trebuie, în ordine descrescătoare.
4. **ARP** traduce  $IP \rightarrow MAC$  în interiorul unui segment. Adresele IP rămân constante, cele MAC se rescriu la fiecare salt.
5. **DHCP** distribuie automat adrese, în patru mesaje. Peste granițele de broadcast e nevoie de **relay**.

## 5.13 Ce urmează și unde citiți mai departe

Avem adrese ierarhice și mecanismele care le leagă de lumea fizică. Cursul următor arată cum se folosește ierarhia: cum decide un ruter, pentru fiecare pachet în parte, pe unde să îl trimită mai departe.

La **laboratorul 4** proiectați planul complet de adresare al unei firme, cu cifre reale, și îl automatizați prin DHCP.

- RFC 791 - Internet Protocol
- RFC 826 - Address Resolution Protocol
- RFC 1918 - spații de adrese pentru rețele private
- RFC 2131 - Dynamic Host Configuration Protocol
- RFC 4632 - CIDR: arhitectura de alocare și agregare a adreselor
- RFC 3927 - adresarea link-local (APIPA)

## Întrebări de verificare

1. Care este adresa de broadcast a rețelei  $192.168.17.0/26$ ?
  - a) 192.168.17.255
  - b) 192.168.17.63
  - c) 192.168.17.127



- d) 192.168.17.64
2. De ce se alocă subrețelele VLSM începând cu cea mai mare cerință?
- a) Ca să fie mai ușor de citit tabelul
  - b) Pentru că un bloc de dimensiune  $2^n$  trebuie să înceapă la o adresă aliniată la  $2^n$
  - c) Pentru că protocoalele de rutare cer această ordine
  - d) Nu contează ordinea
3. O stație vrea să trimită un pachet către o adresă IP din altă rețea. Pentru cine emite cererea ARP?
- a) Pentru adresa IP a destinației finale
  - b) Pentru adresa IP a propriului default gateway
  - c) Pentru adresa de broadcast a rețelei destinație
  - d) Nu emite nicio cerere ARP
4. Ce se întâmplă cu adresele MAC ale unui pachet care traversează trei rutere?
- a) Rămân neschimbate, ca și cele IP
  - b) Se rescriu pe fiecare segment Ethernet
  - c) Se șterg complet după primul ruter
  - d) Sunt înlocuite cu adrese IP
5. O stație are adresa 169.254.13.7. Ce s-a întâmplat?
- a) A primit adresa de la DHCP
  - b) Nu a primit răspuns la Discover și și-a atribuit singură o adresă APIPA
  - c) Gateway-ul este configurat greșit
  - d) Mască este prea mică
6. De ce mesajul DHCP Request se trimite în broadcast, deși clientul știe deja adresa serverului?
- a) Pentru că protocolul cere așa
  - b) Ca să afle și serverele respinse și să își elibereze rezervările
  - c) Pentru că este mai rapid
  - d) Pentru că serverul nu are încă adresa clientului
7. Câte adrese utilizabile oferă un /29?

- a) 8
- b) 6
- c) 4
- d) 14

8. La ce folosește câmpul TTL din antetul IPv4?

- a) Măsoară latența pachetului
- b) Este decrementat de fiecare router și oprește buclele de rutare
- c) Indică prioritatea pachetului
- d) Conține suma de control



## 6. Rutare

---

---

|                                         |
|-----------------------------------------|
| Din cursul trecut                       |
| Ce este, de fapt, un ruter              |
| Ce este o rută și ce e metrica          |
| De unde vin rutele                      |
| Distanța administrativă                 |
| Tabela de rutare și procesul de decizie |
| Ruta implicită                          |
| Rute statice                            |
| Traseul complet al unui pachet          |
| Sumarizarea rutelor și interfața nulă   |
| Greșeli frecvente                       |
| Rezumat și glosar                       |
| Ce urmează și unde citiți mai departe   |
| Întrebări de verificare                 |

---

---

Un ruter nu cunoaște Internetul. Nu are o hartă a lumii și nu întreabă pe nimeni unde e destinația. Are doar o listă de direcții și o regulă de alegere, pe care le aplică din nou și din nou, pentru fiecare pachet în parte. Cursul acesta arată cum arată acea listă, cum ajung rutele în ea și de ce ordinea liniilor contează atât de mult.

### 6.1 Din cursul trecut

**Ce trebuie să aveți în minte.** Adresa IP e **ierarhică**: partea de rețea plus partea de host, separate de **maskă**. **ȘI între adresă și maskă** dă adresa de rețea. Această operație e inima cursului de azi. Adresele IP rămân **constante** de la un capăt la altul; cele MAC se rescriu la fiecare salt. **TTL** este decrementat de fiecare ruter și oprește buclele.

#### 6.1.1 Ce veți putea face după acest curs

**Ce veți putea face după acest capitol.**

- Să explicați ce e o rută și ce rol are metrica
- Să spuneți, pentru orice rută, din ce sursă provine și ce distanță administrativă are
- Să citiți o tabelă de rutare și să prevedeați exact ce se întâmplă cu un pachet dat
- Să configurați rute statice, cu next-hop sau cu interfață de ieșire, și să știți când merge fiecare

- Să scrieți o rută implicită și o rută flotantă de rezervă
- Să calculați o rută sumarizată și să folosiți interfața Null0

## 6.2 Ce este, de fapt, un ruter

Din punct de vedere arhitectural, un ruter este un **calculator specializat**: are procesor, memorie și dispozitive de intrare/ieșire. Diferențele față de un calculator obișnuit sunt însă semnificative, iar toate servesc aceluiași scop - să ia decizii de dirijare cât mai repede.

Tabela 6.1 Ce este, de fapt, un ruter.

| Componentă | Rol într-un ruter                                                            | Se pierde la repornire? |
|------------|------------------------------------------------------------------------------|-------------------------|
| ROM        | codul de pornire (bootstrap) și o imagine de urgență                         | nu - e permanentă       |
| Flash      | sistemul de operare, adică imaginea IOS                                      | nu                      |
| RAM        | configurația <b>activă</b> , tabela de rutare, tabela ARP, cozile de pachete | <b>da - tot</b>         |
| NVRAM      | configurația <b>de pornire</b>                                               | nu                      |
| ASIC-uri   | circuite dedicate care execută comutarea pachetelor în hardware              | -                       |

### Observație

**Distincția care salvează o configurație.** `running-config` trăiește în **RAM** și este ce rulează chiar acum. `startup-config` trăiește în **NVRAM** și este ce se va încărca la următoarea pornire.

O configurație perfectă, nesalvată cu `copy running-config startup-config`, dispare la prima pană de curent - sau la prima repornire făcută de altcineva. Este eroarea numărul unu în laboratoare și, din păcate, și în rețele reale.

**De ce un ruter e mai lent decât un switch.** Un switch citește primii 12 octeți ai cadrului - cele două adrese MAC - și caută într-o tabelă cu potrivire **exactă**. Operația se face în hardware, într-un singur ciclu. Un ruter trebuie să decapsuleze nivelul 2, să citească antetul IP, să **decrementeze TTL**, să **recalculeze suma de control**, să caute în tabela de rutare cu potrivire de **prefix** - nu exactă - și apoi să reîncapsuleze cadrul cu alte adrese MAC, ceea ce poate necesita o rezoluție ARP. Ruterele moderne fac și asta în hardware, dar rămâne mai multă muncă. De aici și diferența de preț per port.

Pe lângă funcția de bază, ruterele moderne mai fac: filtrarea traficului cu ACL-uri (cursul 7), translatarea adreselor cu NAT și PAT (cursul 9), stabilirea de tuneluri, atribuirea de adrese ca server DHCP și proxy ARP.

### 6.3 Ce este o rută și ce e metrica

#### Observație

**Ruta.** Tripletul (**destinație, direcție, distanță**).

„Pentru a ajunge la Brașov pot trece prin Ploiești; distanța totală va fi de 120 km.” - (Brașov, Ploiești, 120 km).

Tripletul e suficient nu doar pentru a merge, ci și pentru a **alege** între mai multe posibilități.

În rețele, „distanța” în kilometri nu spune nimic util. Mărimea asociată unei rute se numește **metrică** și poate fi oricare dintre următoarele, sau o combinație:

Tabela 6.2 Ce este o rută și ce e metrica.

| Metrică              | Ce măsoară                                | Cine o folosește |
|----------------------|-------------------------------------------|------------------|
| Hop count            | numărul de rutere până la destinație      | RIP              |
| Cost                 | invers proporțional cu lățimea de bandă   | OSPF             |
| Metrică compusă      | bandă, întârziere, încărcare, fiabilitate | EIGRP            |
| Atribute de politică | preferințe administrative, nu performanță | BGP              |

#### Exemplu

Ruterul S are două căi spre aceeași destinație: una prin A, cu 2 hopuri, dar prin două legături seriale de 64 kbps; alta prin B, cu 3 hopuri, dar prin fibră de 10 Gbps. Ce alege RIP?

**Rezolvare.** RIP alege calea prin A, cu 2 hopuri - pentru că este singurul lucru pe care îl măsoară. Rezultatul: traficul trece prin legături de 64 kbps, în loc de fibră de 10 Gbps. Diferența este de peste 150 000 de ori. Exact această slăbiciune a dus la înlocuirea RIP cu OSPF, care măsoară costul invers proporțional cu banda. Detalii în cursul 10.

### 6.4 De unde vin rutele

Când un ruter neconfigurat pornește, nu cunoaște **nicio** rută. Rutele apar din trei direcții:

**Diferența dintre C și L.** Când configurați 192.168.1.1/24 pe o interfață, ruterul creează **două** intrări: C 192.168.1.0/24 - „rețeaua asta e direct atașată la mine, pe interfața X” L 192.168.1.1/32 - „adresa asta sunt chiar eu” A doua există ca ruterul să știe imediat, fără niciun calcul, când un pachet îi este destinat lui însuși - de exemplu un ping către interfața lui, sau o conexiune SSH de administrare.

Tabela 6.3 De unde vin rutele.

| Simbol     | Sursă     | Cum apare                                                         | Efort                              |
|------------|-----------|-------------------------------------------------------------------|------------------------------------|
| C          | Connected | automat, pentru fiecare interfață activă cu adresă IP configurată | niciunul                           |
| L          | Local     | automat, adresa /32 a interfeței însăși                           | niciunul                           |
| S          | Static    | scrisă manual de administrator                                    | o linie per rețea, per ruter       |
| R, D, O, B | dinamic   | învățată automat de la vecini prin RIP, EIGRP, OSPF, BGP          | configurare inițială, apoi automat |

```

R1# ?
Comenzi disponibile aici:
  show ip route
  show ip route static
  show ip interface brief
  show ip protocols
  show arp
  ping 192.168.5.10
  traceroute 192.168.5.10
  show running-config
  copy running-config startup-config

```

R1#

Comenzile de mai jos sunt cele pe care le veți folosi la fiecare laborator de aici înainte.

Figura 6.1 Terminal: explorați tabela de rutare a unui ruter.

## 6.5 Distanța administrativă

Metrica permite compararea a două rute care provin din **aceeași** sursă. Dar cum comparăm o rută OSPF cu metrica 20 cu o rută RIP cu metrica 3?

**De ce numerele nu sunt comparabile.** Metrica RIP este un **număr de rutere**: 3 înseamnă „trei salturi”. Metrica OSPF este un **cost** calculat din lățimea de bandă: 20 poate însemna „două legături de 100 Mbps” sau „o legătură serială lentă”. Sunt unități complet diferite, ca și cum ați compara 3 kilograme cu 20 de secunde. Numerele nu au nicio relație între ele.

Este nevoie de un al doilea criteriu, aplicat **înaintea** metricii: **distanța administrativă** (AD), care exprimă cât de multă încredere are ruterul în sursa rutei. Cu cât mai mică, cu atât mai multă încredere.

### 6.5.1 Regula completă de selecție

1. **Prefixul cel mai lung câștigă întotdeauna.** O rută /26 bate o rută /8, indiferent de sursă, indiferent de metrică. Acesta e criteriul suprem și nu se negociază.
2. La **același prefix**, câștigă **distanța administrativă** mai mică.

Tabela 6.4 Distanța administrativă.

| Simbol | Sursă                        | AD  | Logica din spate                                               |
|--------|------------------------------|-----|----------------------------------------------------------------|
| C      | Connected                    | 0   | ruterul vede el însuși rețeaua - nimic nu e mai sigur          |
| S      | Static                       | 1   | administratorul a decis explicit; se presupune că știe ce face |
| D      | EIGRP                        | 90  | protocol proprietar, metrică bogată                            |
| O      | OSPF                         | 110 | link-state, vedere completă a topologiei                       |
| i      | IS-IS                        | 115 | tot link-state                                                 |
| R      | RIP                          | 120 | distance-vector, metrică simplistă                             |
| -      | necunoscută / neîncrezătoare | 255 | ruta nu va fi folosită niciodată                               |

3. La **aceeași sursă**, câștigă **metrica** mai mică.

4. Dacă și metrica e egală, ruterul păstrează **ambele** căi și distribuie traficul între ele - mecanism numit *equal-cost load balancing*.

### Exemplu

Un ruter cunoaște următoarele rute. Care ajung în tabela de rutare?

#Rută      1C (10.0.0.8/30, Fa0/0)      2R (141.85.37.0/24, metrică 3)      3C (10.0.0.4/30, Fa0/1)      4R (141.85.37.0/24, metrică 2)      5D (200.0.0.0/16, metrică 31452) 6R (200.0.0.0/16, metrică 3)

**Rezolvare.** Rutele 1 și 3 intră: sunt singurele către destinațiile lor, și oricum au AD 0. **Ruta 4** intră, ruta 2 nu: aceeași destinație, aceeași sursă (RIP), deci decide metrica -  $2 < 3$ . **Ruta 5** intră, ruta 6 nu - și aici e capcana. Metrica 3 pare mult mai bună decât 31452, dar cele două numere provin din protocoale diferite și **nu sunt comparabile**. Se aplică distanța administrativă: EIGRP are 90, RIP are 120, deci ruta EIGRP câștigă. Dacă vi s-a părut că 6 ar trebui să câștige, ați aplicat criteriul greșit - exact ce se întâmplă la examen.

Reținerea acestui tabel se face cel mai ușor prin logica din spatele lui: cu cât ruterul are mai multă informație de primă mână despre o rută, cu atât numărul e mai mic. O interfață proprie e adevărul absolut (0). O rută scrisă de un om e o afirmație verificabilă (1). Un protocol link-state, care are harta, e mai credibil (110) decât unul care repetă zvonuri (120).



| SURSA RUTEI                      | DISTANȚA ADMINISTRATIVĂ                        |
|----------------------------------|------------------------------------------------|
| interfață direct conectată       | 90 - protocol propriu, considerat de încredere |
| rută statică                     | 120 - vechi, metrică slabă                     |
| EIGRP intern                     | 0 - nimic nu bate realitatea fizică            |
| OSPF                             | 1 - administratorul a spus explicit            |
| RIP                              | 110 - standard deschis, link-state             |
| rută necunoscută / inutilizabilă | 255 - nu se instalează niciodată               |

Apasă un element din stânga, apoi perechea lui din dreapta.

Figura 6.2 Sursa rutei și încrederea în ea.

## 6.6 Tabela de rutare și procesul de decizie

Tabela de rutare este harta ruterului către rețeaua din jur. Este organizată de la rutele cele mai **specifice** (mască lungă) spre cele mai **generale**, iar procesul de dirijare a unui pachet are exact doi pași.

1. **Primirea pachetului.** Se decapsulează antetul de nivel 2, se citește adresa IP destinație din antetul de nivel 3, se decrementează TTL și se recalculează suma de control.
2. **Găsirea rutei.** Se parcurge tabela; pentru fiecare rută se face ȘI între masca ei și adresa destinație, iar rezultatul se compară cu adresa de rețea a rutei. **Prima potrivire câștigă.** Dacă nu se găsește nimic, pachetul e aruncat și se trimite un ICMP *destination unreachable*.

**Cea mai lungă potrivire de prefix.** Ordonarea tabelii de la specific la general nu e o convenție estetică - este chiar algoritmul. Parcurgând în această ordine și oprindu-se la prima potrivire, ruterul găsește automat **cea mai specifică** rută care se potrivește.

De aceea o rută /26 bate una /8: apare mai devreme în tabelă. Iar ruta implicită, /0, e ultima - se aplică doar dacă nimic altceva nu s-a potrivit.

**Experimentele de făcut aici.** 13.1.0.5 - se potrivește atât cu /14, cât și cu /8. Câștigă prefixul mai lung. Urmăriți coloana din dreapta ca să vedeți de ce. 13.128.0.1 - nu mai încapă în /14, deci cade pe /8. Aceeași rețea „13”, altă rută. 192.168.3.7 - se potrivește și cu /24 (spre Null0), și cu /20. Câștigă /24, deci pachetul este **aruncat** deliberat. 8.8.8.8 - nu se potrivește cu nimic specific și cade pe ruta implicită. 172.30.14.9 - nu se potrivește cu niciuna dintre cele două rute conectate. Verificați de ce, calculând ȘI-ul.

**Cele două legi ale rutării. 1.** Rutarea se face **individual, pentru fiecare pachet în parte**. Două pachete ale aceleiași conexiuni pot lua căi diferite, dacă între timp s-a schimbat ceva.

IP DESTINAȚIE

13.1.0.5

| SURSĂ    | DESTINAȚIE         | DIRECȚIE           | TEST              |
|----------|--------------------|--------------------|-------------------|
| C        | 172.30.14.0/30     | Fa0/0              | 13.1.0.4 ≠        |
| C        | 172.30.14.4/30     | Fa0/1              | 13.1.0.4 ≠        |
| S        | 192.168.3.0/24     | Null0              | 13.1.0.0 ≠        |
| S        | 192.168.0.0/20     | 172.30.14.2        | 13.1.0.0 ≠        |
| R        | 12.0.0.0/20        | 172.30.14.6        | 13.1.0.0 ≠        |
| <b>R</b> | <b>13.0.0.0/14</b> | <b>172.30.14.2</b> | <b>13.0.0.0 =</b> |
| R        | 13.0.0.0/8         | 172.30.14.6        | 13.0.0.0 =        |
| S        | 0.0.0.0/0          | 172.30.14.6        | 0.0.0.0 =         |

rezultat **13.0.0.0/14** via **172.30.14.2**

Tabela este ordonată de la cel mai specific prefix spre cel mai general. Se aplică prima rută care se potrivește.

Figura 6.3 Consultarea tabelii de rutare - testați destinații.

2. Fiecare ruter ia decizia **exclusiv pe baza propriei tabeli**. Nu există coordonare, nu există confirmare că pachetul a ajuns și - foarte important - nu există nicio garanție că drumul de întoarcere este același.

### Exemplu

Configurați o rută pe ruterul A spre rețeaua din spatele lui C. Dați ping. Nu merge. De ce?

**Rezolvare.** Pentru că pachetul **a ajuns** la destinație. Dar destinația a răspuns, iar răspunsul nu are pe unde să se întoarcă: ruterul C nu știe unde e rețeaua lui A. Rutarea trebuie configurată **în ambele direcții**, întotdeauna. Este cea mai frecventă greșeală de început și produce simptomul cel mai derutant: „pachetul pleacă, dar nu primesc răspuns”. Cum verificați: dați ping **din C spre A**. Dacă nici acela nu merge, știți sigur unde e problema.

## 6.7 Ruta implicită

Ruta 0.0.0.0/0, numită și *quad-zero*, are masca formată numai din zerouri. Operația ȘI cu ea dă întotdeauna 0.0.0.0, care coincide întotdeauna cu adresa de rețea a rutei - deci se potrivește cu **orice** destinație.

Fiind cea mai generală, ocupă ultima poziție în tabelă și se aplică doar dacă nimic altceva nu s-a potrivit. Este ruta pe care orice rețea mică o are spre furnizorul de Internet.

### Analogie

Un birou de sortare a corespondenței are rafturi pentru destinațiile pe care le cunoaște: „Brașov”, „Cluj”, „București”. Iar la capăt are un raft mare pe care scrie

**„restul lumii”**

. Tot ce nu se potrivește pe niciun raft specific ajunge acolo și pleacă spre centrul de

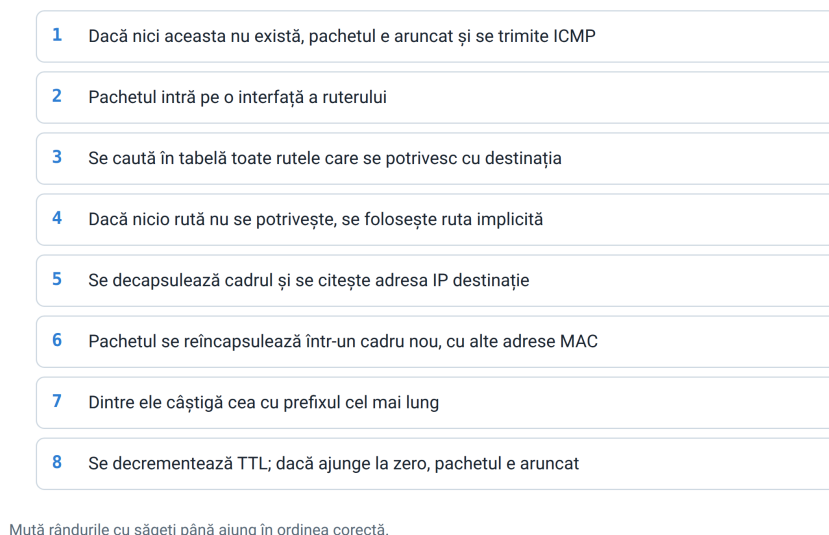


Figura 6.4 Decizia de rutare, pas cu pas.

sortare superior.

Ruta implicită este acel raft. Iar ruterul furnizorului de Internet este centrul de sortare superior.

**Cine are și cine nu are rută implicită.** O rețea de întreprindere are, de regulă, o singură rută implicită - spre furnizor.

Ruterele din **nucleul** Internetului nu au niciuna. Ele trebuie să cunoască explicit toate cele peste 900 000 de prefixe globale, pentru că nu există „mai sus” unde să trimită ce nu cunosc. Aceasta e diferența dintre a fi la marginea Internetului și a fi în mijlocul lui.

## 6.8 Rute statice

O rută statică este scrisă manual de administrator. Direcția poate fi exprimată în două feluri, iar alegerea între ele nu e indiferentă.

Exemplu 6.1 cele trei forme ale unei rute statice copiază

```

1
2 ! prin adresa urmatorului salt — merge intotdeauna
3 R1(config)# ip route 192.168.10.0 255.255.255.0 10.0.0.3
4
5 ! prin interfata de iesire — doar pe legaturi punct-la-punct
6 R1(config)# ip route 192.168.10.0 255.255.255.0 serial 0/0/0
7
8 ! ambele, varianta cea mai explicita si mai rapida
9 R1(config)# ip route 192.168.10.0 255.255.255.0 serial 0/0/0
   ↪ 10.0.0.3
10
11 ! ruta implicita
12 R1(config)# ip route 0.0.0.0 0.0.0.0 10.0.0.3

```

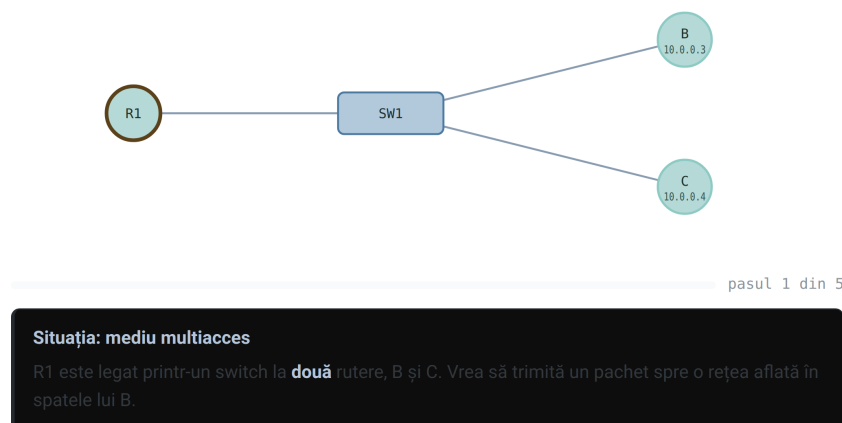


Figura 6.5 De ce next-hop-ul contează pe Ethernet.

**Există și o portiță - dar nu vă bazați pe ea.** Forma cu interfață *poate* funcționa pe Ethernet dacă vecinul are activat **proxy ARP**: el va răspunde la cererea ARP în locul destinației, cu propria adresă MAC.

A construi o rețea pe această bază înseamnă însă a construi o rețea care se strică atunci când cineva, absolut corect, dezactivează proxy ARP din motive de securitate. Scrieți next-hop-ul.

### Exemplu

Trei rutere A, B, C legate în triunghi. A are LAN-ul 172.16.0.0/16, B are 144.13.248.0/21, C are 140.20.0.0/20. Legătura A-C este 10.0.0.4/30 (A are .5, C are .6). Configurați rute statice astfel încât LAN A să comunice cu LAN C, pe calea optimă.

**Rezolvare.** configurație A(config)# ip route 140.20.0.0 255.255.240.0 10.0.0.6  
C(config)# ip route 172.16.0.0 255.255.0.0 10.0.0.5 **Ambele** rute sunt necesare - legea a doua a rutării. Pachetul ajunge la destinație cu prima, dar răspunsul nu are pe unde să se întoarcă fără a doua. Atenție și la măști: /20 înseamnă 255.255.240.0, nu 255.255.255.0. O mască greșită produce un simptom bizar: unele stații răspund, altele nu.

#### 6.8.1 Ruta flotantă

Uneori vrem o cale de rezervă, care să se activeze **doar** dacă cea principală dispare. Se obține dându-i rutei de rezervă o distanță administrativă **mai mare**.

Exemplu 6.2 rută principală și rută flotantă copiază

```

1
2 ! principala - AD implicit 1
3 R1(config)# ip route 172.20.19.0 255.255.255.0 10.0.0.6

```

```

4
5 ! rezerva — AD 200, deci nu intra in tabela cat timp prima e
   ↪ valida
6 R1(config)# ip route 172.20.19.0 255.255.255.0 10.0.0.2 200

```

Verificați cu `show ip route`: ruta cu AD 200 **nu apare** în tabelă atât timp cât cea cu AD 1 e validă. Închideți interfața principală și priviți din nou - a apărut. Redeschideți-o și verificați că se retrage.

## 6.9 Traseul complet al unui pachet

Aici punem cap la cap tot ce am învățat: ARP din cursul 5, comutarea din cursul 3 și rutarea de azi. Este exercițiul care apare, într-o formă sau alta, la fiecare examen.

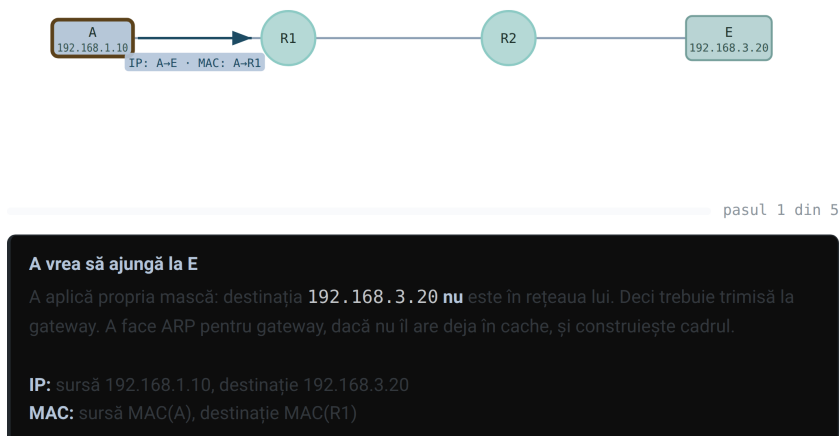


Figura 6.6 Un pachet, de la un capăt la altul.

## 6.10 Sumarizarea rutelor și interfața nulă

Un ruter care are patru rute spre patru subrețele consecutive le poate înlocui pe toate cu una singură, dacă acestea au un prefix comun. Procesul se numește **sumarizare** (sau agregare).

### 6.10.1 Cum se calculează

Se scriu adresele în binar și se observă câți biți de la stânga sunt **comuni tuturor**. Acel număr este prefixul rutei summarize.

192.168.1.0/26

192.168.1.

00

000000

192.168.1.64/26

```

192.168.1.
01
000000
192.168.1.128/26
192.168.1.
10
000000
192.168.1.192/26
192.168.1.
11
000000
-----
comun: primii 24 de biți →
192.168.1.0/24

```

### Exemplu

Care este ruta sumarizată pentru 10.4.0.0/16, 10.5.0.0/16, 10.6.0.0/16 și 10.7.0.0/16?

**Rezolvare.** Se compară al doilea octet, în binar: 4 = 00000100

5 = 00000101

6 = 00000110

7 = 00000111 Primii 6 biți sunt comuni: 000001. Primul octet contribuie cu 8 biți comuni, deci total  $8 + 6 = 14$ . Ruta sumarizată: 10.4.0.0/14. Verificare: /14 acoperă  $2^{(32-14)} = 262\,144$  adrese, adică exact patru blocuri /16. Corect.

| PREFIXUL SUBREȚELOR (BIȚI) | PREFIXUL RUTEI SUMARIZATE (BIȚI) |
|----------------------------|----------------------------------|
| /26 - sferturi de clasă C  | /24                              |

biți economisiți **2.00 biți**  
 rute înlocuite cu una singură **4.00 rute**  
 adrese acoperite de ruta sumarizată **256.00 adrese**  
 linii economisite în tabelă **3.00 linii**

Figura 6.7 Cât economisește o sumarizare.

Alegeți /26 sumarizat la /24: patru rute devin una. Alegeți /30 sumarizat la /22 - cazul legăturilor punct-la-punct dintr-o rețea mare - și veți vedea de ce sumarizarea nu este un moft: **256** de linii înlocuite cu una singură, în tabela fiecărui ruter din rețea.

**De ce contează, la scară.** Avantajul imediat: tabele de rutare mai mici, căutări mai rapide, mai puține actualizări atunci când o subrețea apare sau dispare.

La scara Internetului, sumarizarea este **singurul motiv** pentru care tabela globală de rutare are sute de mii de intrări în loc de sute de milioane. Fără ea, Internetul pur și simplu nu ar funcționa la dimensiunea actuală.

### 6.10.2 Efectul secundar și interfața nulă

Sumarizarea are un preț. Dacă din cele patru subrețele una - să zicem 192.168.1.128/26 - nu mai există, ruta sumarizată 192.168.1.0/24 continuă să atragă traficul pentru ea și îl trimite mai departe, unde va fi aruncat abia la capătul lanțului. În cazul unei bucle, traficul poate chiar oscila între rutere până la expirarea TTL-ului.

Soluția este o rută statică spre **Null0**, interfața nulă: o destinație care aruncă imediat tot ce primește. Fiind mai specifică decât ruta sumarizată, ea câștigă și oprește traficul chiar la sursă.

Exemplu 6.3 ruta „groapă de gunoi” copiază

```
1
2 R1(config)# ip route 192.168.1.128 255.255.255.192 null0
```

Echivalentul din lumea Linux este `/dev/null`; ideea e aceeași - un loc în care lucrurile dispar, la cerere și fără eroare.

### 6.11 Greșeli frecvente

#### Observație

#### Greșeli frecvente.

- **„Am configurat ruta, dar ping-ul nu merge”** În nouă cazuri din zece, lipsește ruta de **întoarcere**. Pachetul ajunge; răspunsul nu are pe unde să vină înapoi. Rutarea se configurează întotdeauna în ambele direcții.
- **„Am comparat metricile a două protocoale diferite”** Nu sunt comparabile. Se aplică întâi distanța administrativă. Prefix mai lung → AD mai mic → metrică mai mică. În această ordine.
- **„Am scris ruta cu interfața de ieșire, pe Ethernet”** Ruterul nu știe ce adresă MAC să pună în cadru, pentru că pe segment sunt mai mulți candidați. Pe Ethernet, scrieți întotdeauna next-hop-ul.
- **„Am salvat configurația... cred”** running-config e în RAM. Fără copy running-config startup-config, se pierde la repornire. Salvați după fiecare bloc de modificări, nu la final.
- **„Am sumarizat și acum unele destinații nu mai răspund”** Ruta sumarizată acoperă și subrețele care nu există sau nu sunt ale voastre. Adăugați rute spre Null0 pentru porțiunile neutilizate ale blocului.

### 6.12 Rezumat și glosar

1. O **rută** este tripletul (destinație, direcție, distanță). Distanța se numește **metrică** și depinde de protocol.
2. Selecția se face în ordine: **prefix mai lung** → **AD mai mic** → **metrică mai mică**.
3. Tabela e ordonată de la specific la general, iar **prima potrivire câștigă**. Ruta impli-

cită e ultima.

4. Pe Ethernet, o rută statică are nevoie de **next-hop**, pentru că altfel lipsește adresa MAC destinație.
5. Adresele IP rămân constante pe tot traseul; cele MAC se rescriu la fiecare salt; TTL scade cu 1 la fiecare ruter.
6. **Sumarizarea** micșorează tabelele; **Null0** oprește traficul spre porțiunile neutilizate.

### 6.13 Ce urmează și unde citiți mai departe

Rutele statice sunt exacte și previzibile, dar nu reacționează la nicio schimbare și nu scalează. Cursul 10 introduce protocoalele care întrețin tabela singure. Până atunci, cursurile 7 și 9 arată ce **altceva** mai poate face un ruter cu pachetele care trec prin el: să le filtreze și să le rescrie adresele.

La **laboratorul 5** configurați exact topologia de aici, întâi cu rute statice și apoi cu OSPF, apoi tăiați un cablu și cronometrați cine își revine mai repede.

- RFC 1812 - cerințele pentru rutere IPv4
- RFC 4632 - CIDR și agregarea rutelor
- Cisco - *IP Routing: Protocol-Independent Configuration Guide*
- Radia Perlman, *Interconnections: Bridges, Routers, Switches and Internetworking Protocols*

### Întrebări de verificare

1. Ce înseamnă [110/65] într-o intrare din tabela de rutare?
  - a) 110 hopuri și 65 de secunde
  - b) Distanța administrativă 110 și costul cumulativ 65
  - c) Costul 110 și metrica 65
  - d) Adresa 110.0.0.65
2. O rută statică are AD 200 și există și o rută OSPF spre aceeași destinație. Care intră în tabelă?
  - a) Cea statică, pentru că rutele statice au prioritate
  - b) Cea OSPF, pentru că 110 este mai mic decât 200
  - c) Amandouă, pentru echilibrarea traficului
  - d) Depinde de metrică
3. Un pachet pentru 13.1.0.5 găsește în tabelă atât 13.0.0.0/8 cât și 13.0.0.0/14. Ce se întâmplă?
  - a) Se folosește prima găsită în tabelă



- b) Castiga /14, pentru ca prefixul este mai lung
  - c) Castiga /8, pentru ca acopera mai mult
  - d) Pachetul este duplicat pe ambele cai
4. O ruta statica prin interfata de iesire nu functioneaza pe Ethernet. De ce?
- a) Pentru ca Ethernet nu suporta rute statice
  - b) Pentru ca ruterul nu stie ce adresa MAC destinatie sa puna in cadru, pe un mediu multiacces
  - c) Pentru ca interfata trebuie sa fie in stare shutdown
  - d) Pentru ca Ethernet nu are TTL
5. Care este ruta sumarizata pentru 10.4.0.0/16, 10.5.0.0/16, 10.6.0.0/16 si 10.7.0.0/16?
- a) 10.0.0.0/8
  - b) 10.4.0.0/14
  - c) 10.4.0.0/16
  - d) 10.4.0.0/22
6. La ce foloseste o ruta statica spre Null0?
- a) Pentru a crea o bucla controlata
  - b) Pentru a arunca imediat traficul spre o subreea inexistentă, acoperita de o ruta sumarizata
  - c) Pentru a redirectiona traficul spre firewall
  - d) Pentru a testa interfata de loopback
7. Unde se afla configuratia care se pierde la repornirea ruterului?
- a) In Flash
  - b) In NVRAM
  - c) In RAM - este running-config
  - d) In ROM
8. Un ping catre o retea indepartata nu primeste raspuns, desi ruta e configurata pe ruterul local. Ce verificati intai?
- a) Cablul
  - b) Daca exista ruta de intoarcere pe ruterul indepartat
  - c) Viteza interfetei
  - d) Adresa MAC a destinatiei

## 7. Liste de acces

---

---

- Din cursul trecut
- Ce este o listă de acces
- Ruter cu ACL sau firewall dedicat?
- Cum se citește o listă de reguli
- Unde se aplică o listă
- Masca wildcard
- Listele standard
- Listele extinse
- Listele cu nume și editarea lor
- Liste care țin cont de context
- Verificare, jurnalizare, comentarii
- Protejarea accesului la echipament
- Greșeli frecvente
- Rezumat și glosar
- Ce urmează
- Întrebări de verificare

---

---

Un ruter, lăsat în pace, dirijează tot ce primește. Nu are opinii despre trafic: dacă are o rută, trimite pachetul mai departe. Lista de acces este mecanismul prin care îi dăm, pentru prima dată, dreptul de a spune nu. Este totodată limbajul în care se descrie traficul pe echipamentele Cisco - aceleași liste apar mai târziu la NAT, la calitatea serviciilor și la filtrarea rutelor. Cursul acesta pornește de la zero: ce este o regulă, cum se citește o listă, de ce ordinea liniilor decide totul și de ce masca folosită aici este pe dos față de tot ce ați învățat până acum.

### 7.1 Din cursul trecut

**Ce trebuie să aveți în minte.** Ruterul ia o decizie **pentru fiecare pachet în parte**, pe baza tabelii de rutare. Decizia se ia prin **longest prefix match**: cea mai specifică rută câștigă. Un pachet fără rută este **aruncat**. Ruterul nu îl păstrează și nu îl ghicește. Ruterul **decapsulează** cadrul, citește antetul IP, apoi **reîncapsulează** pachetul într-un cadru nou pe interfața de ieșire. Rutarea trebuie configurată **în ambele sensuri** - altfel răspunsul nu are pe unde să vină înapoi.

Ultimul punct revine astăzi, într-o formă nouă și mult mai neplăcută: nu doar rutarea

trebuie gândită în ambele sensuri, ci și **filtrarea**. O listă care permite corect traficul dus, dar uită traficul întors, produce exact același simptom - „nu merge” - dintr-un motiv complet diferit.

### 7.1.1 Ce veți putea face după acest curs

**Ce veți putea face după acest capitol.**

- Să explicați ce este un ACL și ce poate identifica dintr-un pachet
- Să parcurgeți mental o listă de reguli și să spuneți exact ce se întâmplă cu un pachet dat
- Să calculați o mască wildcard și să spuneți ce interval de adrese acoperă
- Să alegeți între o listă standard și una extinsă și să știți unde se plasează fiecare
- Să scrieți liste cu nume și să editați o singură linie fără să rescrieți tot
- Să permiteți traficul de întoarcere fără să deschideți rețeaua din exterior
- Să depanați o listă folosind contorul de potriviri

## 7.2 Ce este o listă de acces

**Ideea, înainte de orice sintaxă.** Imaginați-vă că ruterul are, pe fiecare ușă, o foaie de hârtie cu propoziții de forma: „*dacă pachetul arată așa* → *lasă-l*” sau „*dacă pachetul arată așa* → *aruncă-l*”. Fiecare propoziție are deci **două părți**: un **criteriu** (cum arată pachetul) și o **acțiune** (permite sau deny). O propoziție se numește **regulă** sau **ACE** (Access Control Entry). Foaia întreagă se numește **listă de acces**, sau **ACL**. Atât. Restul cursului este despre cum se scriu criteriile, în ce ordine se citesc propozițiile și pe care dintre uși se lipește foaia.

### Observație

**Access Control List (ACL).** O succesiune ordonată de condiții definite de administrator, folosită pentru **identificarea** unor tipuri de trafic. Traficul identificat poate fi apoi filtrat, marcat, translatat, criptat sau asociat cu alte acțiuni. Lista în sine nu face nimic până nu este **aplicată** undeva.

Ultima propoziție merită subliniată, pentru că e sursa primei confuzii de laborator: o listă scrisă în configurație și neaplicată pe nicio interfață **nu filtrează absolut nimic**. Este doar un text. Aplicarea se face separat, cu o a doua comandă.

### 7.2.1 Filtrarea nu este singura utilizare

ACL-ul este, pe un echipament Cisco, modul standard de a răspunde la întrebarea „despre care trafic vorbim?”. De aceea îl veți întâlni în locuri care nu au nicio legătură cu securitatea:

**Consecință practică.** Într-un ACL folosit pentru NAT sau pentru VPN, **permite** nu înseamnă „lasă să treacă”, ci „acesta este traficul despre care vorbesc”. Iar **deny** nu aruncă nimic - doar exclude traficul din selecție. Aceeași sintaxă, cu totul alt înțeles al cuvintelor.

Tabela 7.1 Filtrarea nu este singura utilizare.

| Utilizare                  | Ce face lista acolo                                                              | Unde apare în curs |
|----------------------------|----------------------------------------------------------------------------------|--------------------|
| <b>Filtrare</b>            | decide ce trece și ce nu - singurul caz în care <b>deny</b> chiar aruncă pachete | cursul de azi      |
| <b>NAT</b>                 | identifică adresele interne care au voie să fie translate                        | cursul 9           |
| <b>QoS</b>                 | identifică traficul care primește prioritate în cozi                             | cursul 4 (802.1p)  |
| <b>Acces la echipament</b> | de pe ce adrese se poate deschide o sesiune SSH pe ruter                         | secțiunea 12       |
| <b>Filtrarea rutelor</b>   | ce actualizări de rutare se acceptă de la vecini                                 | cursul 10          |
| <b>VPN</b>                 | ce trafic „interesant” declanșează construirea tunelului criptat                 | cursul 9           |

### 7.2.2 Ce poate „vedea” o listă dintr-un pachet

Criteriile disponibile acoperă nivelurile 3 și 4 ale modelului OSI - exact antetele pe care ruterul le citește oricum:

Tabela 7.2 Ce poate „vedea” o listă dintr-un pachet.

| Nivel | Criteriu                                   | Exemplu de scriere  |
|-------|--------------------------------------------|---------------------|
| 3     | adresa IP <b>sursă</b>                     | host 192.168.1.7    |
| 3     | adresa IP <b>destinație</b>                | 10.0.0.0 0.0.0.255  |
| 3     | <b>protocolul</b> încapsulat               | ip, icmp, ospf, gre |
| 4     | <b>portul</b> sursă sau destinație         | eq 80, range 20 21  |
| 4     | <b>tipul</b> mesajului ICMP                | echo, echo-reply    |
| 4     | <b>starea</b> conexiunii TCP (aproximativ) | established         |

Ce **nu** poate vedea un ACL clasic: numele utilizatorului, conținutul paginii cerute, numele domeniului, faptul că un fișier conține un virus. Toate acestea sunt la nivelurile 5–7 și cer un echipament de alt tip.

#### Analogie

Un portar la intrarea unei clădiri de birouri poate verifica de unde veniți, unde vreți să ajungeți și la ce etaj - pentru că scrie pe legitimație. Nu poate verifica ce aveți în geantă și nici ce intenții aveți. Un ACL este acest portar. Un firewall modern este portarul

#### plus

scannerul de bagaje.

### 7.3 Router cu ACL sau firewall dedicat?

Întrebarea apare imediat ce cineva realizează că ruterul poate filtra: dacă poate, de ce mai cumpărăm firewall-uri? Răspunsul e că fac lucruri diferite, iar diferența nu e de calitate, ci de rol.

**Ruter cu liste de acces.** Funcția principală este **rutarea**; filtrarea e un supliment. Decizii pe baza nivelurilor **3 și 4 Fără stare**: fiecare pachet e judecat izolat, ca și cum ar fi primul. Nu costă nimic în plus - echipamentul există deja. Perfect pentru filtre simple și pentru protejarea echipamentului însuși.

#### Firewall dedicat

- Funcția principală este **filtrarea**; poate ruta, dar limitat
- Decizii pe baza nivelurilor **3–7**, cu inspecție de conținut
- **Cu stare**: ține minte sesiunile deschise și acceptă automat răspunsurile
- Criptare hardware, jurnalizare serioasă, interfață de administrare
- Necesari la marginea oricărei rețele care expune servicii

Un ACL costă latență și încărcare pe procesorul ruterului: fiecare pachet trebuie comparat, în cel mai rău caz, cu toate regulile din listă. Pe un ruter mic, o listă de 300 de linii aplicată pe interfața de Internet se simte.

#### 7.3.1 Unde se pun serviciile expuse public

Arhitectura clasică plasează un firewall între ruterul de margine și rețeaua internă și scoate serviciile publice - server web, server de mail - într-o zonă separată, numită **DMZ** (Demilitarized Zone).

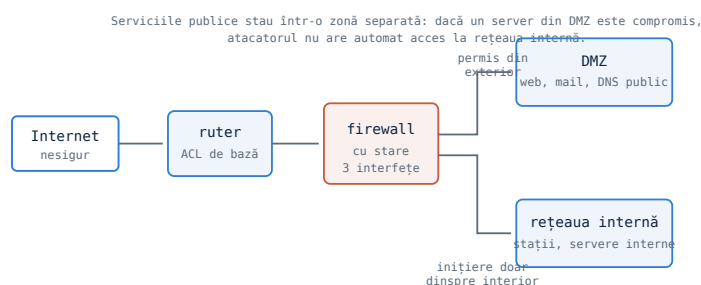


Figura 7.1 Poziția ACL-ului în arhitectura de margine. Ruterul face prima triere, grosieră; firewall-ul face deciziile fine și ține evidența sesiunilor.

### 7.4 Cum se citește o listă de reguli

Aceasta este secțiunea care, dacă e înțeleasă bine, face restul cursului aproape banal. Sunt trei propoziții.

**Observație**

**Cele trei propoziții.** 1. Regulile se testează **secvențial, de sus în jos**.

2. La **prima potrivire** se aplică acțiunea, iar restul listei **nu se mai citește**.

3. Dacă nicio regulă nu se potrivește, se aplică **deny any** - o regulă **invizibilă**, aflată automat la sfârșitul oricărei liste.

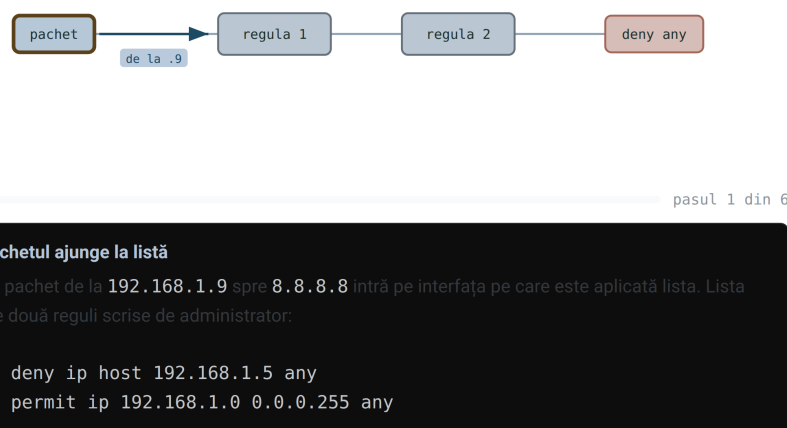


Figura 7.2 Parcurgerea unei liste, pas cu pas.

**Observație**

**Lecția care se învață o singură dată.** O listă aplicată din greșeală pe interfața prin care sunteți conectat prin SSH vă va deconecta **instantaneu** de la echipament, iar reconectarea nu va mai fi posibilă. Într-un laborator este o lecție de zece secunde; într-o rețea reală, pe un ruter aflat la 200 de kilometri, este o deplasare la fața locului.

Obiceiul profesional: pe echipamente aflate la distanță, aplicați lista cu un **reload** în 5 programat înainte. Dacă vă blocați singur, ruterul repornește peste cinci minute cu configurația veche. Dacă totul merge, anulați repornirea cu **reload cancel**.

#### 7.4.1 Ordinea de parcurgere, ca exercițiu

#### 7.5 Unde se aplică o listă

O listă scrisă nu face nimic. Ca să aibă efect, trebuie legată de un loc precis, iar locul are **trei coordonate**.

**„In” și „out” se judecă din perspectiva ruterului.** Aceasta e sursa a jumătate din greșelile de laborator. **in** și **out** nu se referă la rețeaua dumneavoastră, ci la **ruter**. **in** = traficul care **intră în ruter** prin acea interfață, venind dinspre cablu. **out** = traficul care **iese din ruter** prin acea interfață, plecând spre cablu. Deci același pachet, care merge de la LAN spre Internet, este **in** pe interfața dinspre LAN și **out** pe interfața dinspre Internet.

- 1 Pachetul intră pe interfața pe care este aplicată lista
- 2 Dacă nu se potrivește, se trece la regula următoare
- 3 Se testează criteriul primei reguli
- 4 Restul regulilor nu se mai evaluează
- 5 La prima potrivire se aplică permit sau deny
- 6 Dacă lista se termină fără potrivire, se aplică deny any implicit

Mută rândurile cu săgeți până ajung în ordinea corectă.

Figura 7.3 Puneți pașii în ordine.

Tabela 7.3 Unde se aplică o listă.

| Coordonată              | Ce înseamnă                                         | Valori                     |
|-------------------------|-----------------------------------------------------|----------------------------|
| <b>protocolul rutat</b> | pentru ce familie de adrese se aplică               | IPv4, IPv6                 |
| <b>interfața</b>        | pe ce ușa a ruterului stă foaia                     | Gi0/0, Se0/1/0, VLAN 10... |
| <b>direcția</b>         | traficul care <b>intră</b> sau cel care <b>iese</b> | in, out                    |

Puteți alege să îl opriți în oricare dintre cele două locuri - dar în locuri diferite, cu efecte diferite.

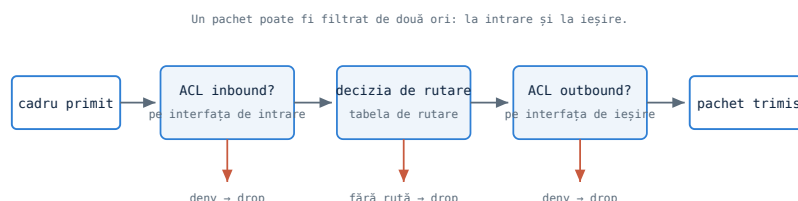


Figura 7.4 Traseul unui pachet prin ruter. ACL-ul inbound este verificat înainte de decizia de rutare, deci un pachet respins acolo nici măcar nu consumă timp de căutare în tabelă.

**De ce inbound este, de obicei, mai eficient.** Un pachet oprit la intrare este aruncat înainte ca ruterul să caute în tabela de rutare, să decrementeze TTL și să recalculeze suma de control. Pentru trafic pe care oricum nu îl vrem, e muncă economisită.

Excepția: dacă o singură regulă trebuie să se aplice traficului venit din **mai multe** interfețe de intrare, o singură listă **out** pe interfața de ieșire înlocuiește trei sau patru liste **in**.

### Exemplu

Un ruter cu 2 interfețe rulează dual stack (IPv4 și IPv6). Care este numărul maxim de ACL-uri de filtrare care pot fi aplicate simultan?

**Rezolvare.** Regula generală: pe o interfață, pentru un protocol rutat și o direcție, poate exista o **singură** listă activă. Deci: 2 interfețe  $\times$  2 protocoale rutate  $\times$  2 direcții = **8 liste**. Corolarul practic: dacă aplicați o a doua listă pe aceeași interfață, direcție și protocol, ea nu se adaugă la prima - o **înlocuiește**, tăcut. Un motiv în plus să folosiți liste cu nume, unde adăugarea unei reguli se face în interiorul listei existente.

## 7.6 Masca wildcard

Aici trebuie să vă dezvățați ceva. Într-o listă de acces, adresele nu se însoțesc de mască de rețea, ci de **mască wildcard**, iar regula ei este exact inversă.

### Observație

**Regula wildcard, în două rânduri.** Bitul **0** înseamnă „acest bit **trebuie** să se potrivească”.

Bitul **1** înseamnă „**ignoră** acest bit, poate fi orice”.

**De ce e pe dos și cum să nu vă mai încurcați.** La masca de rețea, **1** înseamnă „partea de rețea” - adică partea fixă. La wildcard, partea fixă e marcată cu **0**. Trucul de memorare: la wildcard, **1 = wild = liber = orice**. Cifra 1 seamănă cu un semn de exclamare: „nu mă interesează!”. Iar pentru cazurile obișnuite există o scurtătură aritmetică: **wildcard = 255.255.255.255 minus masca**. Scădeți octet cu octet: 255.255.255.255 - 255.255.255.0 = 0.0.0.**255** 255.255.255.255 - 255.255.248.0 = 0.0.**7**.255

Wildcard-ul poate însă face lucruri pe care o mască de rețea nu le poate. Masca cere ca biții de 1 să fie **contigui**, de la stânga. Wildcard-ul nu are această restricție: putem lăsa un bit de 0 pe ultima poziție și selecta astfel **doar adresele pare** dintr-o rețea. E rar folosit, dar explică de ce cele două nu sunt același lucru cu semnele schimbate.

| ADRESĂ DE REFERINȚĂ | WILDCARD  | ADRESĂ DE TESTAT |
|---------------------|-----------|------------------|
| 172.16.8.0          | 0.0.7.255 | 172.16.13.4      |

```

referință 10101100.00010000.00001000.00000000
wildcard  00000000.00000000.00000111.11111111
testată   10101100.00010000.00001101.00000100
biți ignorați 11 → acoperă 2048 adrese
verdict   match

```

Figura 7.5 Testor de mască wildcard.

Porniți de la ce e deja în casetă: 172.16.8.0 0.0.7.255 acoperă opt rețele de clasă C consecutive, de la 172.16.8.0 la 172.16.15.255. Uitați-vă la reprezentarea binară: cei trei biți liberi din al treilea octet produc exact  $2^3 = 8$  combinații.

Apoi încercați trei lucruri:



- wildcard 0.0.0.0 peste 10.1.1.7 - o singură adresă, echivalentul lui **host**;
- wildcard 0.0.0.1 peste 11.2.2.90 - exact două adrese, .90 și .91; e trucul prin care două reguli devin una;
- wildcard 255.255.255.255 peste orice adresă - absolut tot, echivalentul lui **any**.

### 7.6.1 Cuvintele cheie care scurtează scrierea

Tabela 7.4 Cuvintele cheie care scurtează scrierea.

| Cuvânt cheie          | Echivalent complet      | Sens                                    |
|-----------------------|-------------------------|-----------------------------------------|
| <b>any</b>            | 0.0.0.0 255.255.255.255 | orice adresă                            |
| <b>host X</b>         | X 0.0.0.0               | exact adresa X                          |
| adresă scrisă singură | X 0.0.0.0               | tot exact adresa X, în listele standard |

### 7.6.2 Calculator: de la prefix la wildcard și înapoi

LUNGIMEA PREFIXULUI (BIȚI)

/22 - patru rețele

biți liberi (host) **10.00 biți**  
 adrese acoperite **1024.00 adrese**  
 wildcard, ultimul octet **255.00**  
 wildcard, octetul 3 **3.00**  
 rețele de clasă C acoperite **4.00 buc**

Figura 7.6 Prefix → wildcard.

Alegeți /22 și citiți rezultatul: wildcard-ul este 0.0.3.255, adică patru rețele de clasă C. Alegeți /30 - cazul legăturilor între rutere - și obțineți 0.0.0.3: patru adrese, din care două utilizabile.

#### Exemplu

Scrieți criteriul care selectează **doar** subrețelele 192.168.16.0/24 până la 192.168.19.0/24, într-o singură linie.

**Rezolvare.** Sunt patru rețele de clasă C consecutive, deci un bloc de  $4 \times 256 = 1024$  adrese, adică un /22. Verificarea alinierii: 16 împărțit la 4 dă rest 0, deci blocul este aliniat și poate fi scris ca prefix unic. (Dacă am fi pornit de la .17, nu s-ar fi putut.) Wildcard-ul pentru /22 este 0.0.3.255. Criteriul se scrie: 192.168.16.0 0.0.3.255 Verificare: al treilea octet are ultimii doi biți liberi, deci ia valorile 16, 17, 18, 19. Exact ce voiam.

## 7.7 Listele standard

O listă standard răspunde la o singură întrebare: **de unde vine pachetul?** Nu poate spune nimic despre destinație, protocol sau port. E puțin - dar e suficient pentru cazul

Tabela 7.5 Listele standard.

| Tip             | Identificare                  | Criterii                          | Unde se plasează                     |
|-----------------|-------------------------------|-----------------------------------|--------------------------------------|
| <b>Standard</b> | numere 1–99 și 1300–1999      | doar adresa <b>sursă</b>          | cât mai aproape de <b>destinație</b> |
| <b>Extinsă</b>  | numere 100–199 și 2000–2699   | sursă, destinație, protocol, port | cât mai aproape de <b>sursă</b>      |
| <b>Cu nume</b>  | un nume ales de administrator | standard sau extinsă              | după tipul ei                        |

frecvent „stația aceasta nu are voie în segmentul acela”.

Exemplu 7.1 anatomia unei reguli standard copiază

```

1
2 R(config)# access-list 50 deny host 172.16.1.1
3           |      |      |
4           |      |      +--- criteriul: ce sursa
5           |      +----- actiunea: permit sau deny
6           +----- numarul listei (1–99 =>
                        ↪ standard)

```

Exemplu 7.2 exemplu complet: blochează o stație, permite restul rețelei copiază

```

1
2 R(config)# access-list 50 remark blocheaza statia problematica
3 R(config)# access-list 50 deny host 172.16.1.1
4 R(config)# access-list 50 permit 172.16.0.0 0.0.255.255
5
6 R(config)# interface fastEthernet 0/1
7 R(config-if)# ip access-group 50 in

```

### Observație

**Ordinea decide totul.** Dacă cele două reguli ar fi inversate, stația 172.16.1.1 s-ar potrivi cu `permit 172.16.0.0 0.0.255.255` - pentru că face parte din acea rețea - ar fi acceptată, iar regula de `deny` nu ar fi verificată niciodată.

**Regula mai specifică trebuie să fie întotdeauna deasupra celei mai generale.** Aceasta este singura regulă de aur a listelor de acces.

#### 7.7.1 De ce lista standard se plasează lângă destinație

#### 7.8 Listele extinse

Lista extinsă răspunde la patru întrebări deodată: **de unde, unde, cu ce protocol și pe ce port**. Prețul este o sintaxă mai lungă și obligația de a nu greși ordinea argumentelor.

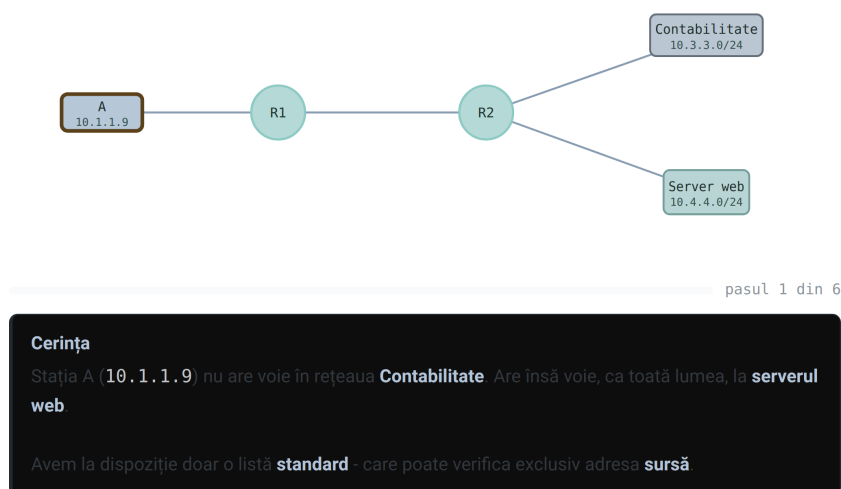


Figura 7.7 De ce lista standard merge lângă destinație.

### Exemplu 7.3 anatomia unei reguli extinse copiază

```

1
2 access-list 101 deny tcp host 172.16.6.1 192.168.1.0 0.0.0.255 eq
   ↪ 23
3
4      |      |      |      |      |      |      |
5      |      |      |      |      |      |      +----- destinația +
6      |      |      |      |      |      |      +----- sursa (aici: o
7      |      |      |      |      |      |      +----- singura stație)
8      |      |      |      |      |      |      +----- protocolul: ip,
9      |      |      |      |      |      |      +----- tcp, udp, icmp...
10     |      |      |      |      |      |      +----- acțiunea
11     +-----+-----+-----+-----+-----+-----+-----+
12     ↪ => extinsă)

```

**Ordinea argumentelor: sursă înainte de destinație, întotdeauna.** Este greșeala numărul unu la liste extinse. `deny tcp A B eq 80` înseamnă „de la A spre B, portul 80 al lui B”. Dacă inversați A și B, lista este sintactic corectă și complet inutilă.

Iar dacă vreți să filtrați după **portul sursă**, acesta se scrie **imediat după sursă**: `permit tcp any eq 80 192.168.0.0 0.0.0.255` înseamnă „de la orice server web, spre rețeaua noastră”.

Tabela 7.6 Protocoale și porturi uzuale.

| Serviciu      | Protocol     | Port   | Cuvânt cheie IOS      |
|---------------|--------------|--------|-----------------------|
| Web           | TCP          | 80     | eq www sau eq 80      |
| Web securizat | TCP          | 443    | eq 443                |
| SSH           | TCP          | 22     | eq 22                 |
| Telnet        | TCP          | 23     | eq telnet             |
| FTP (control) | TCP          | 21     | eq ftp                |
| DNS           | UDP (și TCP) | 53     | eq domain             |
| DHCP          | UDP          | 67, 68 | eq bootps / eq bootpc |
| Ping          | ICMP         | -      | echo, echo-reply      |

### 7.8.1 Protocoale și porturi uzuale

Operatorii disponibili pentru porturi: **eq** (egal), **neq** (diferit), **lt** (mai mic), **gt** (mai mare) și **range** 20 21 (interval închis).

Exemplu 7.4 patru reguli extinse, citite în cuvinte copiază

```

1
2 ! permite tot traficul IP de la o singura statie , catre orice
   ➔ destinatie
3 access-list 101 permit ip host 10.0.0.1 any
4
5 ! opreste tot traficul cu originea in reteaua 10.0.0.0/24
6 access-list 101 deny ip 10.0.0.0 0.0.0.255 any
7
8 ! opreste Telnet de la o statie anume catre o retea anume
9 access-list 101 deny tcp host 172.16.6.1 192.168.1.0 0.0.0.255 eq
   ➔ 23
10
11 ! permite Telnet de la toate statiile dintr-o retea , spre oriunde
12 access-list 101 permit tcp 172.16.6.0 0.0.0.255 any eq telnet

```

### 7.8.2 Evaluador interactiv

Editați lista din caseta de sus și urmăriți cum se schimbă verdictele. Trei observații merită făcute cu atenție.

**Prima.** Regula 4, **deny ip host 192.168.10.66 any**, pare să blocheze stația problematică - dar tabelul arată că pachetul ei este **permis**, de regula 3. Motivul este ordinea: **permit icmp 192.168.10.0 0.0.0.255 any** se potrivește prima, iar restul listei nu se mai verifică. Mutați regula 4 deasupra regulii 3 și priviți cum se schimbă verdictul. Aceasta este, în două rânduri, întreaga capcană a listelor de acces: **o regulă corectă, plasată prea jos, nu există.**

| lista de acces                                                                                                                                                                                                                |               |            |          |          |          |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|------------|----------|----------|----------|
| <pre> permit tcp 192.168.10.0 0.0.0.255 host 10.0.0.50 eq 80 deny tcp 192.168.10.0 0.0.0.255 host 10.0.0.50 permit icmp 192.168.10.0 0.0.0.255 any deny ip host 192.168.10.66 any permit ip 192.168.10.0 0.0.0.255 any </pre> |               |            |          |          |          |
| PACHET                                                                                                                                                                                                                        | SURSĂ         | DESTINAȚIE | PROTOCOL | LINIA    | REZULTAT |
| web din LAN                                                                                                                                                                                                                   | 192.168.10.12 | 10.0.0.50  | tcp:80   | linia 1  | permit   |
| SSH spre server                                                                                                                                                                                                               | 192.168.10.12 | 10.0.0.50  | tcp:22   | linia 2  | deny     |
| ping spre Internet                                                                                                                                                                                                            | 192.168.10.12 | 8.8.8.8    | icmp     | linia 3  | permit   |
| statia interzisa                                                                                                                                                                                                              | 192.168.10.66 | 8.8.8.8    | icmp     | linia 3  | permit   |
| din alta retea                                                                                                                                                                                                                | 172.16.0.9    | 10.0.0.50  | tcp:80   | implicit | deny     |

La finalul oricărei liste există un deny any implicit, care nu se scrie și nu se vede în configurație.

Figura 7.8 Evaluador de listă de acces.

**A doua.** Ultimul pachet, venit din altă rețea, nu se potrivește cu nicio regulă și cade pe deny implicit - deși nimeni nu a scris acea linie nicăieri.

**A treia.** Ștergeți ultima linie, permit ip 192.168.10.0 0.0.0.255 any, și priviți câte pachete rămân permise. Este demonstrația practică a faptului că o listă fără o linie de permisiune finală este un zid.

### 7.8.3 Unde se plasează lista extinsă

**De ce regulile de plasare sunt inversate.** O listă standard nu poate specifica destinația. Plasată lângă sursă, ar bloca traficul acelei surse **spre orice destinație**. Plasarea lângă destinație limitează dauna la exact segmentul vizat.

O listă **extinsă** descrie precis și sursa, și destinația, deci nu există risc de daună colaterală. Atunci e mai eficient să opriți traficul **cât mai devreme**, ca să nu consume bandă traversând rețeaua degeaba.

| SITUAȚIA                                         | CE FOLOSIȚI                              |
|--------------------------------------------------|------------------------------------------|
| blochez o stație spre un singur departament      | listă extinsă pe interfața de margine    |
| blochez accesul la un server web anume           | listă standard aplicată pe liniile VTY   |
| blochez tot traficul unei rețele spre Internet   | listă standard, lângă destinație, out    |
| permit doar SSH spre ruter, de la administratori | listă extinsă, lângă sursă, in           |
| identific traficul care trebuie translatat       | listă folosită de NAT, fără access-group |

Apasă un element din stânga, apoi perechea lui din dreapta.

Figura 7.9 Ce tip de listă, unde.

## 7.9 Listele cu nume și editarea lor

Listele numerotate au două neajunsuri serioase, iar al doilea e cel care doare.

### Listă numerotată

- Se cheamă „254” - peste șase luni nimeni nu mai știe ce face
- **Nu se poate șterge o singură linie.** `no access-list 254` șterge **toată** lista
- Regulile noi se adaugă obligatoriu la **sfârșit** - deci sub **permit**-ul general
- O modificare înseamnă: șterge tot, rescrie tot, sperând că nu ai uitat nimic

**Listă cu nume.** Se cheamă `FILTRU_DMZ_IN` - se înțelege din nume Fiecare regulă are un **număr de secvență**, iar `no 30` șterge doar regula 30 O regulă nouă se poate **insera** între altele, dându-i un număr intermediar Este forma obligatorie pentru liste reflexive și pentru IPv6

**De ce numerele de secvență merg din 10 în 10.** IOS numerotează automat regulile cu 10, 20, 30... tocmai ca să vă lase loc între ele. Dacă mai târziu aveți nevoie de o regulă între 10 și 20, îi dați numărul 15 și ea se așază exact acolo. Dacă ați fi numerotat din 1 în 1, nu ați mai fi avut unde insera nimic - și ar fi trebuit să renumerați lista cu `ip access-list resequence`.

### Exemplu 7.5 crearea și editarea unei liste cu nume copiază

---

```

1
2 R(config)# ip access-list extended FILTRU_LAN_IN
3 R(config-ext-nacl)# 20 permit ip any any
4
5 ! ne dam seama ca lipsesc doua reguli, si trebuie sa fie INAINTE
6 R(config-ext-nacl)# 5 permit icmp host 10.0.0.1 any
7 R(config-ext-nacl)# 10 deny icmp any any
8
9 ! regula de pe linia 5 avea adresa gresita: o stergem punctual
10 R(config-ext-nacl)# no 5
11 R(config-ext-nacl)# 5 permit icmp host 10.0.0.2 any
12
13 ! aplicarea pe interfata - la fel ca la listele numerotate
14 R(config)# interface fastEthernet 0/1
15 R(config-if)# ip access-group FILTRU_LAN_IN in

```

---

**Metoda de siguranță pentru liste lungi.** Chiar și cu liste cu nume, pentru o modificare majoră profesioniștii lucrează așa: copiază lista din `show running-config` într-un editor de text, o modifică acolo, o testează mental, apoi șterge lista veche și lipește noua versiune într-o singură operație. Astfel rețeaua nu petrece niciun moment cu o listă pe jumătate scrisă.

## 7.10 Liste care țin cont de context

### 7.10.1 Problema traficului de întoarcere

Vrem ca stațiile noastre să navigheze pe web, dar nimeni din exterior să nu poată iniția o conexiune spre ele. Sună simplu. Nu este.

**De ce un ACL clasic nu poate exprima asta.** Un ACL vede **pachete**, nu **conexiuni**. Când serverul web răspunde, pachetul lui vine dinspre exterior, spre rețeaua noastră - exact ca un pachet trimis de un atacator. Ruterul nu ține minte că noi am întrebat primii. Fiecare pachet e judecat ca și cum ar fi primul din lume. Așa că orice regulă care lasă răspunsul să intre lasă și atacul să intre. Aceasta este definiția lui **fără stare** (stateless), și e diferența fundamentală față de un firewall.

### 7.10.2 Soluția minimală: established

Opțiunea **established** se potrivește cu pachetele TCP care au bitul **ACK** sau **RST** setat - adică cele care fac parte dintr-o conversație deja pornită. Primul pachet al unei conexiuni noi, cel cu SYN singur, nu are ACK, deci nu se potrivește și este oprit.

Exemplu 7.6 ieșire liberă, intrare doar ca răspuns copiază

---

```

1
2 R0(config)# ip access-list extended IESIRE_HTTP
3 R0(config-ext-nacl)# 10 permit tcp 192.168.0.0 0.0.0.255 any eq
   ↪ www
4
5 R0(config)# ip access-list extended INTRARE_HTTP
6 R0(config-ext-nacl)# 10 permit tcp any eq www 192.168.0.0
   ↪ 0.0.0.255 established
7
8 R0(config)# interface fa1/0
9 R0(config-if)# ip access-group IESIRE_HTTP out
10 R0(config-if)# ip access-group INTRARE_HTTP in

```

---

Observați structura celei de-a doua reguli: sursa este **any eq www** - orice server web - iar destinația este rețeaua noastră. Este oglinda primei reguli, cu sursa și destinația inversate, plus cuvântul magic la final.

#### Limitele lui established

- Verifică doar biții ACK și RST - un atacator poate **fabrica** un pachet cu ACK setat, fără să existe vreo conexiune
- Funcționează **exclusiv pentru TCP**; UDP-ul și ICMP-ul nu au biți de control, deci nu au echivalent
- Nu se descurcă cu aplicații care negociază porturi dinamic (FTP activ, SIP, jocuri online)

### 7.10.3 Listele reflexive

Un ACL reflexiv construiește **dynamic** regulile de întoarcere, pe baza sesiunilor observate la ieșire. Când o stație internă deschide o conexiune, ruterul creează automat, temporar, regula inversă - și o șterge când sesiunea se încheie sau expiră.

Se definesc doar prin liste extinse cu nume și folosesc două cuvinte cheie: **reflect**, care marchează traficul de urmărit, și **evaluate**, care aplică regulile generate din el.

---

#### Exemplu 7.7 ACL reflexiv pentru HTTP și ICMP copiază

---

```

1
2 R0(config)# ip access-list extended OUTBOUND
3 R0(config-ext-nacl)# 10 permit tcp 192.168.0.0 0.0.0.255 any eq
   ↪ www reflect TRAFIC_HTTP
4 R0(config-ext-nacl)# 20 permit icmp 192.168.0.0 0.0.0.255 any
   ↪ reflect TRAFIC_ICMP
5
6 R0(config)# ip access-list extended INBOUND
7 R0(config-ext-nacl)# 10 evaluate TRAFIC_HTTP
8 R0(config-ext-nacl)# 20 evaluate TRAFIC_ICMP
9
10 R0(config)# interface fastEthernet 1/0
11 R0(config-if)# ip access-group OUTBOUND out
12 R0(config-if)# ip access-group INBOUND in

```

---

Acesta este primul pas real spre un firewall cu stare: ruterul nu mai judecă fiecare pachet izolat, ci ține minte ce sesiuni au fost inițiate din interior. Spre deosebire de **established**, funcționează și pentru UDP și ICMP, pentru că se bazează pe perechea de adrese și porturi, nu pe biții TCP.

### 7.10.4 Liste condiționate temporal

Uneori regula nu depinde de trafic, ci de ceas: acces permis doar în programul de lucru, backup permis doar noaptea.

---

#### Exemplu 7.8 Telnet permis doar în programul de lucru copiază

---

```

1
2 R(config)# time-range PROGRAM_LUCRU
3 R(config-time-range)# periodic Monday Tuesday Wednesday Thursday
   ↪ Friday 9:00 to 18:00
4
5 R(config)# ip access-list extended ACL_ORAR
6 R(config-ext-nacl)# 10 permit tcp any 192.168.1.0 0.0.0.255 eq
   ↪ telnet time-range PROGRAM_LUCRU
7

```



## 8 R# show time-range

**Observație**

**Ceasul ruterului trebuie să fie corect.** O listă temporală depinde de ora echipamentului. Un ruter repornit își pierde ceasul și pornește, de regulă, de la 1 martie 1993 - moment în care regula dumneavoastră nu se mai aplică niciodată, sau se aplică permanent. Sincronizarea prin **NTP** nu este un moft, ci o precondiție.

**7.11 Verificare, jurnalizare, comentarii**

Trei mecanisme fac diferența între un ACL scris și un ACL înțeles.

**Comentariile** - **remark**, maximum 100 de caractere - explică **intenția** fiecărui grup de reguli, nu sintaxa lor. Peste șase luni, când altcineva citește configurația, un comentariu valorează cât zece minute de reconstituit raționamentul.

**Jurnalizarea** - cuvântul cheie **log** la finalul unei reguli - generează un mesaj care conține numărul listei, acțiunea, adresa sursă și numărul de pachete. Mesajul apare la primul pachet care se potrivește și apoi din 5 în 5 minute, mecanism gândit exact ca să nu inunde jurnalele.

```
R1# ?
Comenzi disponibile aici:
  show access-lists
  show ip access-lists
  show ip interface gi0/0
  show ip interface brief
  show time-range
  clear access-list counters
  show running-config | include access
```

R1#

Scrieți sau alegeți o comandă. Contorul de potriviri, din prima comandă, este cel mai util instrument de depanare din tot cursul.

Figura 7.10 Terminal: comenzile de verificare.

**Contorul de potriviri este cel mai bun instrument de depanare.** Rulați **show access-lists** și uitați-vă la numărul din paranteze de lângă fiecare regulă.

**Zero potriviri** pe o regulă care ar trebui să prindă trafic înseamnă că traficul nu ajunge acolo: fie lista e pe interfața greșită, fie pe direcția greșită, fie o regulă de deasupra a prins deja traficul. În exemplul din terminal, regula 40 are **0 matches** - și e chiar cazul discutat mai devreme, în care regula 30 prinde traficul înaintea ei.

**Multe potriviri** pe un **deny** pe care nu îl așteptați înseamnă că filtrați ceva ce nu voiati. Ambele situații se văd în trei secunde.

**7.12 Protejarea accesului la echipament**

Până acum am filtrat trafic care **trece prin** ruter. Ultima aplicație este diferită: filtrăm traficul care se **oprește la** ruter - sesiunile de administrare.

Un ruter accesibil prin SSH de oriunde din Internet este o parolă distanță de un dezastru.

Soluția standard nu este o listă pe interfață, ci una aplicată pe **liniile virtuale** (VTY), cu o comandă diferită: `access-class` în loc de `ip access-group`.

Exemplu 7.9 SSH permis doar din rețeaua administratorilor copiază

```

1
2 R(config)# access-list 5 remark doar statiile administratorilor
3 R(config)# access-list 5 permit 192.168.99.0 0.0.0.255
4
5 R(config)# line vty 0 15
6 R(config-line)# access-class 5 in
7 R(config-line)# transport input ssh
8 R(config-line)# exec-timeout 5 0

```

Tabela 7.7 Protejarea accesului la echipament.

| Comandă                      | Unde se aplică | Ce filtrează                           |
|------------------------------|----------------|----------------------------------------|
| <code>ip access-group</code> | pe o interfață | traficul care <b>trece prin</b> router |
| <code>access-class</code>    | pe liniile VTY | cine se poate <b>conecta la</b> router |

Avantajul acestei abordări: lista se aplică o singură dată și acoperă toate interfețele, indiferent pe unde ar veni încercarea de conectare. Un ACL pe interfață ar trebui replicat pe fiecare ușă.

### 7.13 Greșeli frecvente

#### Observație

##### Greșeli frecvente.

- **„Am scris lista, dar nu filtrează nimic”** Lista există în configurație, dar nu a fost aplicată pe nicio interfață. Este doar text. Verificați cu `show ip interface` ce listă e aplicată și pe ce direcție.
- **„Am pus regula, dar nu are efect”** O regulă de deasupra prinde deja traficul. Regula dumneavoastră nu e citită niciodată. `show access-lists` → 0 matches. Mutați regula mai sus. Specificul deasupra generalului.
- **„Am blocat tot, deși voiam să blochez o singură stație”** Lista are doar reguli `deny`. După ele vine `deny any` implicit. Adăugați o linie `permit` finală pentru restul traficului.
- **„Ping-ul merge într-un sens, dar nu se întoarce”** Ați permis traficul dus și ați uitat răspunsul. Ca și la rutare, filtrarea se gândește în ambele sensuri. Fie o regulă pentru `echo-reply`, fie `established`, fie o listă reflexivă.
- **„Am confundat in cu out”** Direcția se judecă din perspectiva ruterului, nu a rețelei. Întrebați-vă: pachetul **intră în** router pe această interfață, sau **iese din** el?

- „Am folosit masca de rețea în loc de wildcard" permit 10.0.0.0 255.255.255.0 se acceptă fără eroare și selectează cu totul altceva. Wildcard = 255.255.255.255 minus masca. Verificați cu `show access-lists`, unde IOS afișează explicit wildcard bits.
- „Am aplicat lista pe interfața prin care eram conectat" Sesiunea se închide instantaneu și nu se mai poate redeschide. Pe echipamente la distanță: `reload in 5` înainte, `reload cancel` după ce confirmați că merge.

## 7.14 Rezumat și glosar

**Ce trebuie să rămână.** Un ACL este o listă **ordonată** de perechi criteriu-acțiune, citită de sus în jos, până la **prima potrivire**. La sfârșitul oricărei liste există un **deny any invizibil**. Lista trebuie **aplicată** pe o interfață și o direcție ca să aibă efect. `in` și `out` se judecă din perspectiva **ruterului**. Wildcard: **0** = **trebuie să se potrivească**, **1** = **ignoră**. Este inversul măștii. **Standard** = doar sursa → lângă **destinație**. **Extinsă** = sursă, destinație, protocol, port → lângă **sursă**. Listele cu **nume** permit editarea unei singure linii; cele numerotate, nu. Traficul de întoarcere cere **established** sau o listă **reflexivă**. `show access-lists` și **contorul de potriviri** rezolvă majoritatea problemelor.

## 7.15 Ce urmează

Am învățat să descriem traficul și să îl oprim pe un ruter Cisco, fără nicio noțiune de stare și fără niciun fel de criptografie. Cursul următor privește aceeași problemă din perspectiva unui firewall cu stare și a unei mașini Linux, și adaugă exact dimensiunea care lipsește complet aici: cum protejăm conținutul, nu doar traseul.

**Laboratorul 6** configurează liste standard și extinse pe topologia firmei, împreună cu NAT - și veți vedea acolo, în practică, ce înseamnă „aceeași listă, alt înțeles al cuvântului permit".

- Cisco - *Security Configuration Guide: Access Control Lists*
- RFC 3871 - cerințe operaționale de securitate pentru infrastructura IP
- RFC 2827 (BCP 38) - filtrarea la intrare împotriva adreselor sursă falsificate
- Manualul `iptables(8)`, pentru comparație cu abordarea Linux

## Întrebări de verificare

1. Un ACL conține o singura regula: `deny host 10.0.0.5`. Ce trafic trece prin interfața pe care este aplicat?
  - a) Tot, în afara de cel de la 10.0.0.5
  - b) Niciun trafic, din cauza lui `deny any` implicit de la finalul listei
  - c) Doar traficul de la 10.0.0.5
  - d) Depinde de direcția aplicării
2. Unde se plasează o lista de acces standard?

- a) Cat mai aproape de sursa
  - b) Cat mai aproape de destinatie
  - c) Pe interfata de loopback
  - d) Nu conteaza
3. Ce inseamna wildcard-ul 0.0.0.3 aplicat adresei 192.168.1.4?
- a) Doar adresa 192.168.1.4
  - b) Adresele 192.168.1.4 pana la 192.168.1.7
  - c) Toata retea 192.168.1.0/24
  - d) Nicio adresa
4. Care este wildcard-ul echivalent unui prefix /26?
- a) 0.0.0.63
  - b) 0.0.0.255
  - c) 0.0.0.31
  - d) 255.255.255.192
5. Pe o interfata, cate liste de filtrare IPv4 pot fi active simultan?
- a) Una singura
  - b) Doua: una in si una out
  - c) Cate reguli are lista
  - d) Nelimitat
6. Ce face cuvantul cheie established intr-o regula extinsa?
- a) Deschide o conexiune noua
  - b) Permite doar pachetele TCP care au bitul ACK sau RST setat, adica cele dintr-o conversatie deja pornita
  - c) Crijteaza traficul
  - d) Aplica regula doar in orele de program
7. O regula de permit nu pare sa aiba efect. Ce verificati intai?
- a) Daca ruterul are suficienta memorie
  - b) Contorul de potriviri din show access-lists si daca o regula de deasupra prinde deja traficul
  - c) Daca interfata este in stare shutdown

- d) Versiunea de IOS
8. Ce comanda restrictioneaza cine se poate conecta prin SSH la ruter?
- a) ip access-group 5 in, pe fiecare interfata
  - b) access-class 5 in, pe liniile VTY
  - c) ip access-list standard SSH
  - d) transport input ssh
9. De ce se prefera listele cu nume celor numerotate?
- a) Sunt mai rapide
  - b) Permit stergerea si inserarea unei singure reguli, prin numere de secventa
  - c) Ocupa mai putina memorie
  - d) Se aplica automat pe toate interfetele

## 8. Securizarea rețelei

---

---

- Din cursul trecut
- Nivelul transport: de ce există porturi
- Antetul TCP, câmp cu câmp
- Deschiderea și închiderea unei conexiuni
- Cele trei familii de atacuri
- iptables: tabele și lanțuri
- Anatomia unei reguli iptables
- Politica implicită și un firewall complet
- Arhitectura perimetrului
- SSH și cele trei concepte de securitate
- Greșeli frecvente
- Rezumat și glosar
- Ce urmează
- Întrebări de verificare

---

---

Până acum am construit o rețea care funcționează. Cursul acesta pornește de la observația neplăcută că Internetul nu este un loc prietenos, și adaugă trei lucruri care lipsesc. Întâi nivelul transport - pentru că aproape toate atacurile și aproape toate apărările se poartă acolo, la nivelul porturilor și al flag-urilor TCP. Apoi filtrarea cu iptables, adică ce se întâmplă când firewall-ul nu mai e un ruter Cisco, ci o mașină Linux. Și, la final, criptografia din spatele unei simple sesiuni SSH: cum reușesc două calculatoare care nu s-au întâlnit niciodată să se pună de acord asupra unei chei secrete, în văzul tuturor.

### 8.1 Din cursul trecut

**Ce trebuie să aveți în minte.** Un ACL este o listă **ordonată**, citită până la **prima potrivire**. La final există un **deny any invizibil**. Lista trebuie **aplicată** pe o interfață și o direcție ca să existe. Listele extinse pot filtra după **protocol și port** - adică după nivelul 4. Un ACL este **fără stare**: nu ține minte că am întrebat noi primii.

Ultimul punct este exact locul unde ne oprisem, nemulțumiți. Astăzi vedem ce înseamnă un echipament care **ține minte** sesiunile - dar, ca să înțelegem ce anume ține minte, trebuie mai întâi să înțelegem ce este o sesiune.

### 8.1.1 Ce veți putea face după acest curs

#### Ce veți putea face după acest capitol.

- Să explicați ce adaugă nivelul transport peste IP și de ce există porturi
- Să alegeți între TCP și UDP pentru o aplicație dată, cu argumente
- Să citiți o strângere de mână în trei pași și să spuneți ce face fiecare flag
- Să recunoașteți cele trei familii de atacuri și să explicați cum funcționează un SYN flood
- Să scrieți reguli iptables corecte, pe lanțul potrivit
- Să construiți un firewall complet, cu politică implicită DROP
- Să explicați ce garantează, separat, autentificarea, confidențialitatea și integritatea
- Să descrieți schimbul Diffie-Hellman și să spuneți ce nu rezolvă el

### 8.2 Nivelul transport: de ce există porturi

**Problema pe care o rezolvă nivelul 4.** Până acum, un pachet ajunge la **calculatorul** potrivit. Adresa IP face exact atât: identifică mașina. Dar pe mașina aceea rulează, în același timp, un browser, un client de mail, un joc și trei actualizări în fundal. Când sosește un pachet, sistemul de operare trebuie să știe **cărei aplicații** îi aparține. Adresa IP nu spune nimic despre asta. De aceea nivelul 4 adaugă încă un număr: **portul**. Fiecare nivel introduce un nou tip de adresă - nivelul 2 adresa MAC, nivelul 3 adresa IP, nivelul 4 portul. Fără porturi, un calculator ar putea rula o singură aplicație de rețea la un moment dat.

#### Observație

**Socket.** Perechea **adresă IP + număr de port**, scrisă 192.168.1.10:49152. O conexiune este identificată complet de **patru** numere: socketul sursă și socketul destinație. Două conexiuni pot avea aceeași destinație și aceeași adresă sursă, dar porturi sursă diferite - și de aceea puteți deschide zece file spre același site fără să se amestece.

**UDP nu este „TCP stricat”.** Tabelul de mai sus enumeră ce *poate* face nivelul transport. UDP face doar primele trei - și o face deliberat. La un apel video, un pachet pierdut retransmis peste 200 de milisecunde este inutil: momentul lui a trecut. Mai bine lipsește. Uneori viteza contează mai mult decât garanțiile.

**TCP - Transmission Control Protocol.** Orientat pe conexiune: se stabilește o sesiune înainte de transfer Sigur: datele ajung garantat și în ordine Control de flux, de congestie și de eroare Antet de **minimum 20 de octeți** SSH, HTTP, SMTP, transfer de fișiere, baze de date

#### UDP - User Datagram Protocol

- Fără conexiune: se trimite și gata
- Nesigur: segmentele se pot pierde sau sosi în altă ordine
- Fără control de flux sau congestie
- Antet de **8 octeți** - sursă, destinație, lungime, sumă de control
- DNS, DHCP, VoIP, IPTV, jocuri, streaming

Tabela 8.1 Nivelul transport: de ce există porturi.

| Rol al nivelului transport | Ce înseamnă                                                   | TCP | UDP       |
|----------------------------|---------------------------------------------------------------|-----|-----------|
| Segmentare                 | împarte fluxul aplicației în bucăți care încap într-un pachet | da  | da        |
| Adresare                   | identifică aplicația prin numărul de port                     | da  | da        |
| Multiplexare               | mai multe aplicații pe aceeași adresă IP                      | da  | da        |
| Siguranța transmisiei      | retransmite ce s-a pierdut, reordonează ce a sosit anapoda    | da  | <b>nu</b> |
| Controlul fluxului         | împiedică un emițător rapid să înece un receptor lent         | da  | <b>nu</b> |
| Inițierea conexiunilor     | stabilește o sesiune înainte de transferul propriu-zis        | da  | <b>nu</b> |

### 8.2.1 Intervalele de porturi

Tabela 8.2 Intervalele de porturi.

| Interval      | Denumire                  | Cine le folosește                                                                                                       |
|---------------|---------------------------|-------------------------------------------------------------------------------------------------------------------------|
| 0 – 1023      | <b>well-known</b>         | servicii standard: 22 SSH, 25 SMTP, 53 DNS, 80 HTTP, 443 HTTPS. Pe Linux, doar procesele privilegiate pot asculta aici. |
| 1024 – 49151  | <b>înregistrate</b>       | aplicații cunoscute, dar neoficiale: 3306 MySQL, 3389 RDP, 8080 proxy HTTP                                              |
| 49152 – 65535 | <b>dinamice</b> / efemere | porturile <b>sursă</b> alese automat de sistemul de operare la fiecare conexiune nouă                                   |

**Consecința pentru filtrare.** Când stația dumneavoastră deschide o pagină web, conexiunea arată așa: 192.168.1.10:51423 → 93.184.216.34:80. Portul **destinație** este cunoscut (80), dar portul **sursă** este imprevizibil.

De aceea o regulă de firewall pentru traficul de întoarcere nu poate specifica portul destinație - el se schimbă la fiecare conexiune. Trebuie fie **established**, fie urmărirea sesiunilor. Aceasta e, în concret, limitarea de care ne-am lovit la finalul cursului trecut.

Priviți a doua comandă. Cele trei conexiuni au **aceeași adresă locală**, dar porturi sursă



```

user@statie:~$ ?
Comenzi disponibile aici:
ss -tuln
ss -tn state established
netstat -tuln
cat /proc/sys/net/ipv4/ip_local_port_range
nmap -sT 192.168.1.10

user@statie:~$

```

Comenzi care arată ce socketuri sunt deschise chiar acum pe o mașină Linux.

Figura 8.1 Terminal: porturi și socketuri.

diferite - și tocmai asta le ține separate. Iar ultima linie e o conexiune în celălalt sens: cineva e conectat prin SSH la mașina noastră, deci portul local este 22, cel binecunoscut.

### 8.3 Antetul TCP, câmp cu câmp

Aproape toate mecanismele discutate mai jos - retransmisia, controlul de flux, strângerea de mână, atacul SYN flood, opțiunea **established** - sunt scrise, la propriu, în acest antet de 20 de octeți.

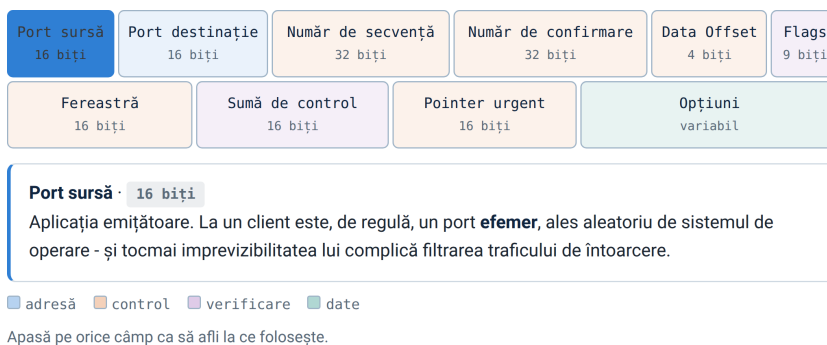


Figura 8.2 Antetul TCP - apăsați pe câmpuri.

Tabela 8.3 Antetul TCP, câmp cu câmp.

| Flag | Rol                                                            | Unde îl întâlnești            |
|------|----------------------------------------------------------------|-------------------------------|
| SYN  | cere sincronizarea numerelor de secvență - deschide conexiunea | SYN flood, --syn în iptables  |
| ACK  | confirmă datele primite; activează câmpul de confirmare        | <b>established</b> în ACL-uri |
| FIN  | anunță că emițătorul a terminat de transmis                    | închiderea normală            |
| RST  | închide brutal conexiunea; răspunsul la un pachet nevalid      | <b>REJECT</b> în iptables     |

## 8.4 Deschiderea și închiderea unei conexiuni

O conexiune TCP se deschide printr-un schimb de **trei** segmente, în care fiecare capăt își comunică numărul de secvență inițial și îl confirmă pe al celuilalt. Se închide, în schimb, prin **patru** - și motivul acestei asimetrii spune ceva despre cum este gândit TCP.

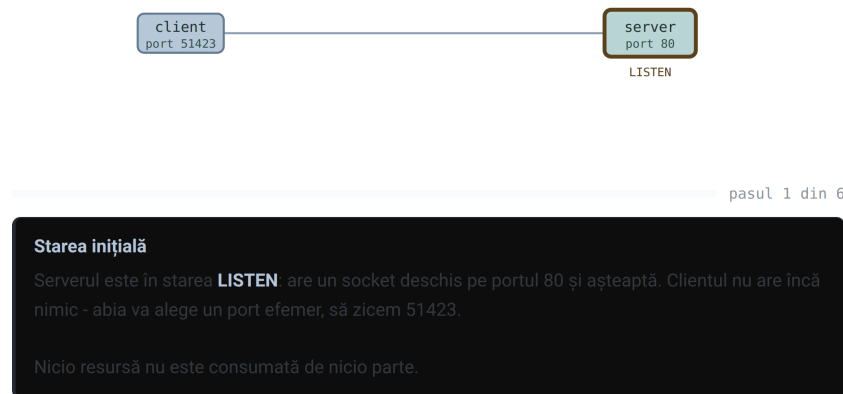


Figura 8.3 Strângerea de mână în trei pași.

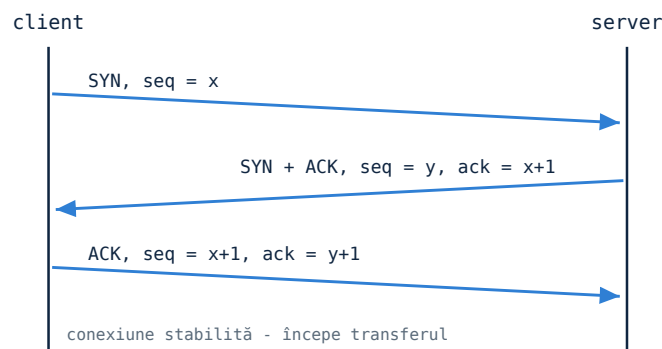


Figura 8.4 Cele trei segmente ale deschiderii unei conexiuni TCP, în forma clasică de diagramă temporală. Panta liniilor reprezintă timpul de propagare.

**De ce numărul de secvență inițial este aleatoriu.** Dacă ar fi previzibil, un atacator care cunoaște adresele și porturile ar putea fabrica segmente care par să facă parte din conversație - fără să vadă traficul. Se numește **TCP sequence prediction** și a fost o vulnerabilitate reală în anii '90, când multe sisteme porneau numerotarea de la o valoare crescută cu un pas fix.

## 8.5 Cele trei familii de atacuri

Orice rețea conectată la Internet este, permanent, ținta unor tentative automatizate. Nu e nevoie să fiți interesant: scanările sunt continue și oarbe. Atacurile se împart în trei familii, după scop.

Tabela 8.4 Cele trei familii de atacuri.

| Familie                       | Scop                              | Exemple                                                  | Ce ajută                              |
|-------------------------------|-----------------------------------|----------------------------------------------------------|---------------------------------------|
| <b>De recunoaștere</b>        | aflarea a ce există și ce rulează | ping sweep, port scan, sniffing, interogări DNS și whois | filtrarea ICMP, DROP în loc de REJECT |
| <b>De blocare</b> (DoS/-DDoS) | indisponibilizarea serviciului    | SYN flood, smurf, amplificare DNS                        | limitarea ratei, SYN cookies          |
| <b>De acces</b>               | obținerea de drepturi nemeritate  | spargere de parole, buffer overflow, man-in-the-middle   | autentificare puternică, criptare     |

**De ce DROP e mai bun decât REJECT în fața Internetului.** REJECT aruncă pachetul și **trimite un răspuns**: „port închis”. Politicos - și extrem de util pentru cel care scanează, pentru că îi confirmă că mașina există. DROP aruncă pachetul **tăcut**. Scannerul nu primește nimic și trebuie să aștepte expirarea temporizatorului pentru fiecare port. O scanare care ar dura o secundă durează minute. În interiorul rețelei, însă, REJECT este preferabil: aplicațiile proprii primesc o eroare imediată în loc să aștepte 30 de secunde degeaba.

### 8.5.1 Cum funcționează un SYN flood



Figura 8.5 SYN flood, pas cu pas.

#### Observație

**Firewall.** Una sau mai multe mașini al căror scop este prevenirea accesului neautorizat la o rețea. Controlează accesul la servicii atât dinspre exterior spre interior, cât și invers. Poate fi un dispozitiv dedicat, o funcție a unui ruter sau un program pe o stație.

Un firewall nu rezolvă toate cele trei familii, dar taie eficient primele două: poate bloca porturile vulnerabile, poate opri inițierea conexiunilor din exterior, poate refuza răspunsul

Tabela 8.5 Cum funcționează un SYN flood.

| Tip                               | Până la ce nivel se uită    | Ține minte sesiunile?                   | Exemplu                           |
|-----------------------------------|-----------------------------|-----------------------------------------|-----------------------------------|
| <b>Stateless</b>                  | 3–4                         | nu - fiecare pachet este judecat izolat | ACL-uri pe ruter                  |
| <b>Stateful</b>                   | 3–4, cu tabelă de conexiuni | da                                      | iptables cu conn-track, Cisco ASA |
| <b>De nivel aplicație (proxy)</b> | până la 7                   | da, plus conținutul                     | proxy HTTP, WAF                   |

la ICMP echo, poate limita numărul de sesiuni pe jumătate deschise și poate opri broadcast-urile direcționate care fac posibil atacul smurf. Împotriva atacurilor de acces ajută mai puțin: acolo lucrează criptografia și autentificarea, din secțiunea 10.

## 8.6 iptables: tabele și lanțuri

**iptables** este utilitarul din spațiul utilizator care configurează **Netfilter**, cadrul de filtrare din nucleul Linux. Permite unei mașini Linux să filtreze pachete, să translateze adrese și să rescrie câmpuri - adică să fie, simultan, ruter și firewall.

### Analogie

Netfilter e ca o clădire cu mai multe puncte de control, dispuse pe traseul obligatoriu al oricărui vizitator. La fiecare punct de control există un teanc de instrucțiuni.

### Lanțul

este punctul de control -

*unde*

pe traseu.

### Tabela

este tipul de instrucțiuni -

*ce fel de decizie*

se ia acolo: filtrare, traducere de adrese sau modificare. Un punct de control poate avea mai multe teancuri.

Tabela 8.6 iptables: tabele și lanțuri.

| Tabelă        | Ce conține                                                                                          |
|---------------|-----------------------------------------------------------------------------------------------------|
| <b>filter</b> | regulile care decid ce trece și ce se aruncă; <b>tabela implicită</b> , folosită dacă nu scrieți -t |
| <b>nat</b>    | regulile de traducere a adreselor - subiectul cursului 9                                            |
| <b>mangle</b> | alterarea specializată a pachetelor: TTL, marcaje, ToS                                              |
| <b>raw</b>    | excepții de la urmărirea conexiunilor                                                               |

Lanțurile sunt punctele din traseul unui pachet în care se aplică reguli. Cheia înțelegerii lui iptables este să știți **prin ce lanțuri trece** fiecare tip de pachet - și sunt doar trei tipuri.

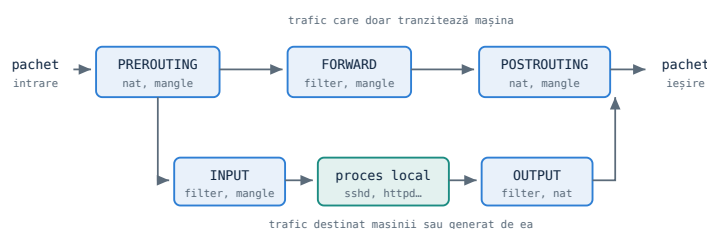


Figura 8.6 Traseul pachetelor prin lanțurile Netfilter. Un pachet care tranzitează mașina trece prin FORWARD; unul destinat ei trece prin INPUT; unul generat de ea trece prin OUTPUT.

### Observație

**Confuzia INPUT / FORWARD.** Este cea mai frecventă sursă de reguli care nu au niciun efect. Întrebarea de pus, de fiecare dată: **cine este destinatarul din antetul IP?**

Dacă destinatarul este chiar mașina Linux (o sesiune SSH către ea, o pagină servită de ea) → **INPUT**.

Dacă destinatarul este altcineva, iar mașina doar rutează → **FORWARD**.

O regulă pusă pe INPUT pentru trafic care tranzitează nu se va aplica niciodată - și nici nu veți primi vreo eroare.

| CE FILTRAȚI                                       | LANȚUL CORECT           |
|---------------------------------------------------|-------------------------|
| cineva se conectează prin SSH la ruterul Linux    | FORWARD                 |
| o stație din LAN accesează un site din Internet   | INPUT                   |
| ruterul însuși își descarcă actualizările         | PREROUTING, tabela nat  |
| traducerea adresei sursă la ieșirea spre Internet | POSTROUTING, tabela nat |
| redirectarea portului 80 spre un server intern    | OUTPUT                  |

Apasă un element din stânga, apoi perechea lui din dreapta.

Figura 8.7 Ce lanț folosiți?.

## 8.7 Anatomia unei reguli iptables

O regulă are întotdeauna două părți: un **șablon** - ce condiții trebuie să îndeplinească pachetul - și o **țintă** - ce se întâmplă cu el.

Exemplu 8.1 o regulă completă, descompusă copiază

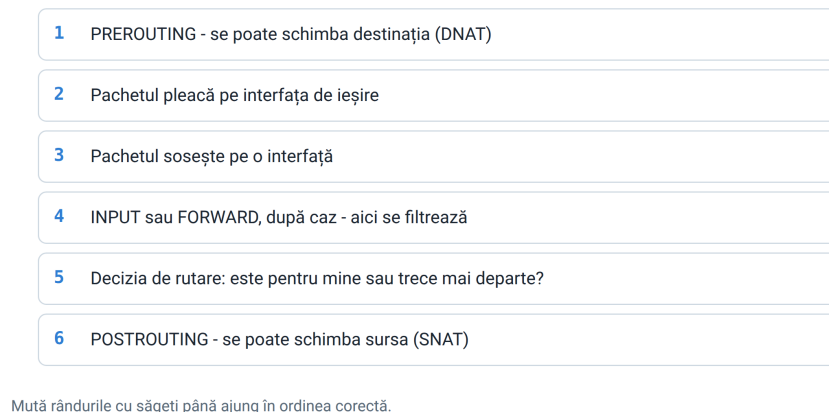


Figura 8.8 Traseul unui pachet prin Netfilter.

```

2 iptables -t filter -A INPUT -s 10.0.0.0/8 -p icmp -j DROP
3 #          + tabela  + op      + lant  +-----+ conditii -----+ +
      ↪ tinta
4 #
5 # -t filter   tabela (implicita , se poate omite)
6 # -A INPUT    adauga la finalul lantului INPUT
7 # -s 10...    sursa: aici masca este NORMALA, nu wildcard!
8 # -p icmp     protocolul
9 # -j DROP     jump: ce se face cu pachetul

```

**Atenție la mască.** La Cisco am folosit **wildcard** (0.0.0.255). În iptables se folosește notația **CIDR normală** (/24) sau masca clasică. Este una dintre puținele locuri unde trecerea de la un ecosistem la altul chiar produce greșeli tăcute.

Tabela 8.7 Anatomia unei reguli iptables.

| Operație                | Formă lungă | Ce face                                            |
|-------------------------|-------------|----------------------------------------------------|
| -A                      | --append    | adaugă o regulă la <b>finalul</b> lanțului         |
| -I                      | --insert    | inserează la o poziție dată (implicit prima)       |
| -D                      | --delete    | șterge o regulă                                    |
| -L -n -v --line-numbers | --list      | afișează regulile, cu con-toare și numere de linie |
| -F                      | --flush     | golește lanțul                                     |
| -P                      | --policy    | schimbă politica implicită                         |

**LOG** este singura țintă care nu oprește parcurgerea. Toate celelalte sunt **terminale**: odată aplicate, pachetul a primit un verdict și lanțul nu se mai citește. La fel ca la ACL-urile Cisco. **LOG** nu decide nimic - doar notează și lasă pachetul să meargă mai departe prin listă. De aceea o regulă de jurnalizare se pune **imediat deasupra** celei de DROP pe care vrea să o documenteze, cu exact aceleași condiții.

Tabela 8.8 Anatomia unei reguli iptables.

| Condiție                             | Ce verifică                                     |
|--------------------------------------|-------------------------------------------------|
| -i eth0 / -o eth1                    | interfața de intrare / de ieșire                |
| -s / -d                              | adresa sursă / destinație, în notație CIDR      |
| -p tcp --dport 22                    | protocolul și portul destinație                 |
| -p tcp --syn                         | doar segmentele de <b>inițiere</b> a conexiunii |
| -m state --state ESTABLISHED,RELATED | starea sesiunii, din tabela conntrack           |
| -m limit --limit 25/s                | rata de sosire - pentru apărarea anti-flood     |
| ! -s 10.0.0.0/8                      | negarea oricărei condiții, cu semnul exclamării |

Tabela 8.9 Anatomia unei reguli iptables.

| Țintă  | Ce face                                                          |
|--------|------------------------------------------------------------------|
| ACCEPT | lasă pachetul să treacă; evaluarea lanțului se oprește           |
| DROP   | aruncă <b>tăcut</b> - expeditorul nu află nimic                  |
| REJECT | aruncă și trimite un ICMP sau un RST TCP                         |
| LOG    | înregistrează în jurnalul de sistem și <b>continuă</b> evaluarea |

```

root@fw:~# ?
Comenzi disponibile aici:
iptables -L -n -v --line-numbers
iptables -S
conntrack -L | head
iptables -Z
iptables-save > /etc/iptables/rules.v4
dmesg | grep IPTABLES

root@fw:~#

```

Comenzile de bază pentru inspectarea unui firewall Linux.

Figura 8.9 Terminal: inspectarea unui firewall Linux.

Priviți prima comandă. Coloana **pkts** este echivalentul exact al contorului de potriviri de la ACL-urile Cisco, și se folosește la fel: regula 4 din INPUT are **0 pachete**, ceea ce înseamnă că nimeni nu a accesat încă serverul web - sau că o regulă de deasupra prinde deja traficul.

### Observație

**Regulile iptables nu supraviețuiesc repornirii.** Sunt ținute în memoria nucleului. Fără iptables-save (sau serviciul netfilter-persistent), la următorul boot mașina pornește cu lanțurile goale și politicile pe ACCEPT - adică fără firewall. Partea bună: dacă v-ați blocat singuri, o repornire vă salvează. Partea proastă: dacă ați

crezut că ați configurat un firewall, nu l-ați configurat.

## 8.8 Politica implicită și un firewall complet

Fiecare lanț predefinit are o **politică** - acțiunea aplicată pachetelor care nu s-au potrivit cu nicio regulă. Este echivalentul lui **deny any** implicit de la Cisco, cu o diferență importantă: aici este configurabilă, iar implicit este **ACCEPT**.

### Politica **ACCEPT** - permisivă

- „Permite tot ce nu e interzis explicit”
- Trebuie să anticipați fiecare lucru rău
- Orice serviciu pornit din greșeală e expus
- Un port uitat rămâne deschis pentru totdeauna

**Politica **DROP** - restrictivă.** „Interzice tot ce nu e permis explicit” Trebuie să enumerați ce vreți să meargă Un serviciu nou nu merge până nu îl deschideți - **și e bine așa** Singura abordare acceptabilă în fața Internetului

### Observație

**Ordinea comenzilor, la propriu.** Setati **întâi** regulile de **ACCEPT** și abia apoi politica **DROP**. Dacă faceți invers și lucrați prin SSH, prima comandă vă deconectează instantaneu de la mașină.

Lanțurile create de utilizator nu pot avea politică implicită - evaluarea lor se întoarce, pur și simplu, în lanțul apelant.

### Exemplu 8.2 un firewall complet, pentru un ruter Linux copiază

```

1
2 # 1. Curatam si pornim de la zero
3 iptables -F
4 iptables -X
5
6 # 2. Traficul pe interfata de bucla – mereu primul, altfel se
   ➔ strica servicii locale
7 iptables -A INPUT -i lo -j ACCEPT
8 iptables -A OUTPUT -o lo -j ACCEPT
9
10 # 3. Regula care rezolva TOT traficul de intoarcere, dintr-o linie
11 iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
12 iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
13
14 # 4. Serviciile pe care le expunem noi
15 iptables -A INPUT -i eth1 -s 192.168.1.0/24 -p tcp --dport 22 -j
   ➔ ACCEPT
16 iptables -A INPUT -p tcp --dport 80 -j ACCEPT

```



```

17
18 # 5. LAN-ul are voie spre Internet
19 iptables -A FORWARD -i eth1 -o eth0 -s 192.168.1.0/24 -j ACCEPT
20
21 # 6. Jurnalizam ce urmeaza sa aruncam (LOG nu opreste parcurgerea)
22 iptables -A INPUT -m limit --limit 5/min -j LOG --log-prefix "
    ↪ IPTABLES-DROP: "
23
24 # 7. ABIA ACUM politicile
25 iptables -P INPUT DROP
26 iptables -P FORWARD DROP
27 iptables -P OUTPUT ACCEPT
28
29 # 8. Salvam, altfel totul dispare la repornire
30 iptables-save > /etc/iptables/rules.v4

```

**Linia care merită reținută.** `-m state --state ESTABLISHED,RELATED -j ACCEPT` este exact ce ne lipsea la ACL-urile Cisco. Nucleul ține o tabelă de conexiuni (conntrack); orice pachet care aparține unei sesiuni deja stabilite este acceptat automat, indiferent de port.

`RELATED` merge mai departe: acceptă și traficul **înrudit** - de exemplu conexiunea de date a unui FTP activ, sau mesajele ICMP de eroare legate de o sesiune existentă. Aceasta este, în concret, diferența dintre stateless și stateful.

### Exemplu

O rețea privată are un ruter Linux. Configurați o politică antispoofing: pachetele cu adrese private nu au voie nici să intre din Internet, nici să iasă spre el.

**Rezolvare.** antispoofing iptables -A FORWARD -i eth0 -s 192.168.0.0/16 -j DROP  
iptables -A FORWARD -i eth0 -s 172.16.0.0/12 -j DROP iptables -A FORWARD -i  
eth0 -s 10.0.0.0/8 -j DROP iptables -A FORWARD -o eth0 -d 192.168.0.0/16 -j DROP  
iptables -A FORWARD -o eth0 -d 172.16.0.0/12 -j DROP iptables -A FORWARD -o  
eth0 -d 10.0.0.0/8 -j DROP Regulile se pun pe FORWARD, pentru că este vorba de trafic  
care **tranzitează** ruterul. Condiția de interfață este obligatorie: `-i eth0` înseamnă  
„venit dinspre Internet”. Fără ea, ați bloca și traficul intern legitim dintre două subrețele  
private - adică exact traficul pe care ruterul trebuie să îl ducă.

### Exemplu

Topologie: un ruter Linux cu `eth0` spre Internet, `eth1` spre un server (142.31.16.9) și `eth2` spre LAN (142.31.16.128/25). Administratorul lucrează de la 214.13.177.2, din Internet. Configurația de plecare este: INPUT: politică ACCEPT, regula `-i eth0 -j DROP`; FORWARD: politică DROP, regula `-i eth0 -j ACCEPT`.

**Rezolvare. #1 - Stațiile din LAN nu ajung la server.** Cauza: politica lanțului FORWARD este DROP, iar singura regulă permite doar traficul venit pe eth0 - adică din Internet. soluția iptables -P FORWARD ACCEPT

**#2 - Doar stațiile din LAN au voie la HTTP-ul serverului.** soluția iptables -F FORWARD iptables -A FORWARD -s 142.31.16.128/25 -d 142.31.16.9 -p tcp --dport 80 -j ACCEPT iptables -A FORWARD -d 142.31.16.9 -p tcp --dport 80 -j DROP Ordinea este obligatorie: regula permisivă, mai specifică, trebuie să fie deasupra celei restrictive. Exact aceeași regulă ca la ACL-uri.

**#3 - Doar administratorul are voie la SSH pe ruter.** Fiind trafic *destinat ruterului*, lanțul este INPUT, nu FORWARD. soluția iptables -F INPUT iptables -A INPUT -s 214.13.177.2 -p tcp --dport 22 -j ACCEPT iptables -A INPUT -p tcp --dport 22 -j DROP

**#4 - Sesiunile TCP să poată fi inițiate doar dinspre interior.** Prima variantă, iptables -A FORWARD -p tcp --syn -j DROP, este greșită: blochează inclusiv conexiunile pornite din LAN. Trebuie restrânsă la traficul care *intră* dinspre Internet: soluția iptables -A FORWARD -i eth0 -p tcp --syn -j DROP Observați că soluția modernă ar fi cu totul alta - o singură linie cu --state ESTABLISHED,RELATED - și că ea acoperă și UDP-ul, și ICMP-ul, pe care --syn nu le atinge deloc.

## 8.9 Arhitectura perimetrului

Unde anume se pune firewall-ul față de ruterul de margine? Decizia este, cel mai adesea, impusă de furnizorul de Internet, dar principiile sunt clare.

Tabela 8.10 Arhitectura perimetrului.

| Arhitectură                      | Când se folosește                    | Observații                                                                                       |
|----------------------------------|--------------------------------------|--------------------------------------------------------------------------------------------------|
| Ruter în față, firewall în spate | rețele medii și mari                 | cea mai frecventă: ruterul acoperă cerințele de rutare pe care firewall-ul nu le poate îndeplini |
| Firewall în față, ruter în spate | cerințe de securitate foarte stricte | ruterul intern este protejat, dar firewall-ul trebuie să facă și rutare                          |
| Un singur echipament             | rețele mici, rutare statică          | simplu și ieftin; punct unic de defectare                                                        |

### 8.9.1 IDS și IPS

Un firewall decide pe baza **antetelor**. Pentru inspecția **conținutului** se adaugă un echipament separat, iar diferența dintre cele două variante este exclusiv de poziționare.

**IPS - Intrusion Prevention System.** Plasat în calea traficului, în serie Poate **opri** pachetele periculoase Devine un potențial punct de gâtuire Dacă pică, poate întrerupe tot traficul Un fals pozitiv taie trafic legitim

#### IDS - Intrusion Detection System

- Primește o **copie** a traficului, în paralel

- Doar **semnalează**; nu poate opri nimic
- Nu influențează performanța rețelei
- Dacă pică, traficul curge nestingherit
- Un fals pozitiv generează doar o alarmă

Alegerea nu este „care e mai bun”, ci „ce se întâmplă când greșește”. Într-o rețea unde disponibilitatea contează mai mult decât riscul, IDS. Într-una unde un singur pachet greșit e inacceptabil, IPS.

## 8.10 SSH și cele trei concepte de securitate

Telnet transmite totul în clar, inclusiv parola. Oricine ascultă traficul o obține - nu e nevoie de niciun efort, doar de acces la un port de switch oglindit sau la o rețea wireless. SSH rezolvă problema pe trei planuri distincte, care merită înțelese **separat**, pentru că se regăsesc, cu alte nume, în orice discuție de securitate.

Tabela 8.11 SSH și cele trei concepte de securitate.

| Concept                  | Întrebarea la care răspunde          | Mecanismul din SSH                     |
|--------------------------|--------------------------------------|----------------------------------------|
| <b>Autentificare</b>     | Sunt cei doi cine pretind că sunt?   | parolă sau chei asimetrice             |
| <b>Confidențialitate</b> | Poate altcineva să citească mesajul? | criptare simetrică (AES, ChaCha20)     |
| <b>Integritate</b>       | A fost mesajul modificat pe drum?    | MAC - cod de autentificare a mesajului |

**De ce sunt trei probleme și nu una.** Sunt independente, iar rezolvarea uneia nu o atinge pe cealaltă. Un mesaj poate fi **criptat** perfect și totuși trimis de un impostor - criptarea nu spune cine scrie. Un mesaj poate fi **semnat** corect și totuși citit de toată lumea. Iar un mesaj criptat poate fi **modificat** în tranzit fără ca atacatorul să înțeleagă ce modifică, ceea ce e suficient ca să facă rău. De aceea orice protocol serios le tratează pe toate trei, cu mecanisme diferite.

### 8.10.1 Confidențialitatea: problema cheii comune

Criptarea simetrică - aceeași cheie la ambele capete - este rapidă și eficientă. Are însă o problemă care pare imposibilă: cum ajung cele două capete să dețină aceeași cheie, dacă singurul canal disponibil este chiar cel pe care vor să îl protejeze?

Răspunsul este **schimbul Diffie-Hellman**, publicat în 1976, și e una dintre cele mai elegante idei din informatică: cele două capete construiesc o cheie comună **fără ca aceasta să fie transmisă vreodată**.

#### Analogie cu vopseluri

Doi oameni vor o culoare secretă comună, dar tot ce își trimit este văzut de toată lumea.

Se pun de acord public asupra unei culori de bază - galben. Fiecare adaugă în secret o culoare proprie și trimite amestecul. Un observator vede cele două amestecuri, dar

### nu poate separa

culorile din ele.

Fiecare adaugă apoi propria culoare secretă în amestecul primit. Ambii ajung la exact aceeași culoare finală - galben plus secretul lui plus secretul ei. Observatorul are galbenul și cele două amestecuri, dar nu poate produce culoarea finală, pentru că i-ar trebui unul dintre secrete.

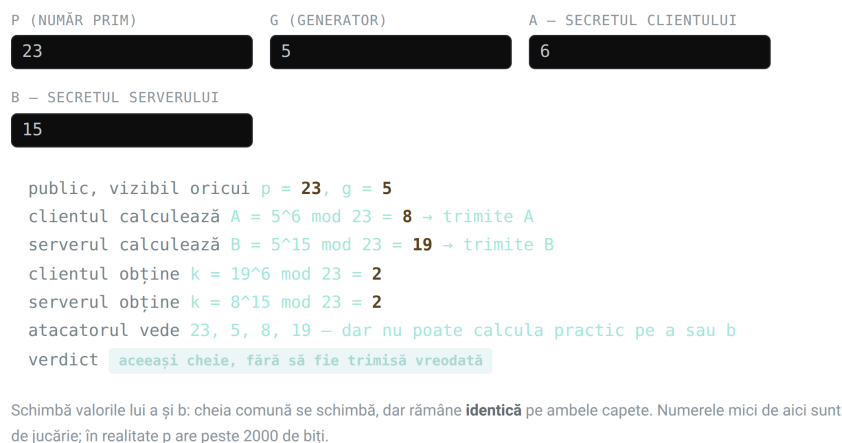


Figura 8.10 Schimbul Diffie-Hellman.

Jucați-vă cu valorile. Schimbați secretul clientului: cheia comună se schimbă - dar rămâne **identică** pe ambele capete, deși cele două secrete nu au părăsit niciodată calculatoarele respective. Ce circula pe canal sunt doar  $p$ ,  $g$ ,  $A$  și  $B$ .

Securitatea stă în faptul că, deși  $A = g^a \bmod p$  se calculează instantaneu, drumul invers - aflarea lui  $a$  din  $A$  - este **problema logaritmului discret**, pentru care nu se cunoaște niciun algoritm rapid. Cu numerele de jucărie de mai sus se sparge prin încercări; cu un  $p$  de 2048 de biți, nu.

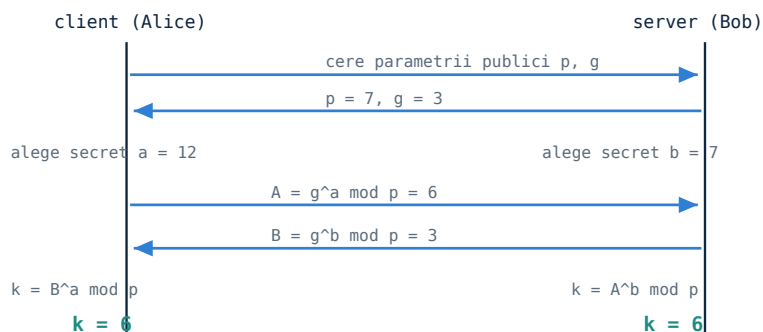


Figura 8.11 Schimbul Diffie-Hellman ca diagramă temporală. Cheia comună  $k$  nu circula niciodată pe canal; ea este calculată independent, la ambele capete.

**Observație**

**Ce NU rezolvă Diffie-Hellman.** Schimbul garantează că nimeni **care doar ascultă** nu află cheia. Nu garantează absolut nimic despre **cine** este la celălalt capăt.

Un atacator plasat în mijloc poate face două schimburi DH separate - unul cu clientul, altul cu serverul - și să decripteze, citească și recripteze tot traficul. Este atacul **man-in-the-middle**, iar soluția nu ține de DH, ci de **autentificare**: de aceea SSH vă întreabă, la prima conectare, dacă recunoașteți amprenta cheii serverului. Mesajul acela pe care toată lumea îl acceptă fără să citească este exact apărarea împotriva acestui atac.

### 8.10.2 Autentificarea: parolă sau cheie

Autentificarea are loc **după** ce canalul criptat există. Pare să rezolve problema parolelor - dar nu complet:

- serverul trebuie să vadă parola ca să o valideze; dacă serverul este compromis, parola este compromisă - și, de regulă, e refolosită și altundeva;
- parolele suficient de sigure sunt greu de reținut, deci utilizatorii aleg parole slabe;
- o parolă poate fi ghicită prin încercări repetate, iar Internetul are răbdare infinită.

De aceea se preferă **cheile asimetrice**. O pereche de chei are proprietatea că ceea ce este criptat cu una poate fi decriptat doar cu cealaltă:  $K^+(K^-(M)) = M$  și  $K^-(K^+(M)) = M$ . Clientul își păstrează cheia privată - care nu părăsește niciodată calculatorul lui - iar cheia publică este configurată pe server.

1. Se stabilește sesiunea sigură prin Diffie-Hellman.
2. Clientul cere autentificarea pentru un anumit utilizator.
3. Serverul trimite un **challenge**: un șir aleator pe care clientul trebuie să îl semneze.
4. Clientul îl criptează cu cheia sa privată. Rezultatul se numește **authenticator**.
5. Serverul verifică folosind cheia publică preconfigurată. Dacă rezultatul coincide cu șirul trimis, clientul deține cheia privată - deci este cine pretinde.

Observați ce nu s-a întâmplat niciodată în cei cinci pași: **cheia privată nu a fost transmisă**. Serverul nu o cunoaște și nu o poate afla. Un server compromis nu vă compromite identitatea pe alte servere - spre deosebire de o parolă.

**De ce doar challenge-ul este criptat asimetric.** Criptarea asimetrică este cu ordine de mărime mai lentă decât cea simetrică. A cripta astfel tot traficul ar fi inacceptabil. Cheia privată este folosită strict pentru operația de autentificare; restul sesiunii merge pe cheia simetrică obținută prin Diffie-Hellman.

Același principiu - **asimetric pentru schimbul de chei, simetric pentru date** - stă la baza TLS și a oricărei conexiuni HTTPS pe care o deschideți.

### 8.10.3 Integritatea: codul MAC

Un **MAC** (Message Authentication Code) este, în esență, un hash calculat **cu o cheie**. Emițătorul îl calculează peste mesaj și îl trimite alături; receptorul îl recalculează cu aceeași cheie și compară. Dacă valorile diferă, mesajul a fost modificat pe drum.

Cheia este esențială. Un hash simplu nu ar ajuta: atacatorul ar modifica mesajul și ar recalcula hash-ul. Fără cheie, nu poate produce un MAC valid pentru mesajul modificat.

**Ce se vede și ce nu, într-o captură SSH.** Un Wireshark pornit pe o sesiune SSH arată adresele IP, porturile, momentele și dimensiunile pachetelor - deci *că* există o conversație, cu cine și cât de intensă este. Se numesc **metadata**, și ele scapă criptării.

Nu arată conținutul, nu arată parola, nu arată comenzile executate și nu permite modificarea traficului fără ca ambele capete să observe imediat.

### 8.11 Greșeli frecvente

#### Observație

#### Greșeli frecvente.

- **„Regula mea iptables nu are niciun efect”** Aproape sigur e pe lanțul greșit: INPUT în loc de FORWARD, sau invers. Întrebați cine e destinatarul din antetul IP. Mașina însăși → INPUT. Altcineva → FORWARD.
- **„Am pus politica DROP și m-am deconectat”** Politica s-a aplicat înaintea regulii care permitea SSH-ul dumneavoastră. Întâi toate regulile ACCEPT, apoi -P DROP. Pe mașini la distanță, testați cu o a doua sesiune deschisă.
- **„Firewall-ul a dispărut după repornire”** Regulile trăiesc în memoria nucleului și nu se salvează singure. iptables-save sau serviciul netfilter-persistent.
- **„Am folosit wildcard-ul Cisco în iptables”** -s 10.0.0.0/0.0.0.255 nu înseamnă ce credeți. iptables folosește masca normală sau CIDR. -s 10.0.0.0/24. Verificați cu iptables -L -n, care afișează prefixul rezultat.
- **„Am blocat ICMP complet, ca să fiu în siguranță”** Ați blocat și mesajele *fragmentation needed*, esențiale pentru descoperirea MTU-ului. Rezultatul: conexiuni care se deschid și apoi îngheață inexplicabil. Blocați doar **echo-request** dinspre exterior; lăsați ICMP-ul de eroare să treacă.
- **„Diffie-Hellman mă protejează de tot”** Vă protejează de cineva care ascultă. Nu de cineva care se așază în mijloc. Verificați amprenta cheii la prima conectare. Este singurul moment în care contează.
- **„Am confundat DROP cu REJECT”** REJECT confirmă atacatorului că mașina există. DROP spre exterior, REJECT spre interior - ca aplicațiile proprii să nu aștepte degeaba.

### 8.12 Rezumat și glosar

**Ce trebuie să rămână.** Nivelul 4 adaugă **portul**; perechea IP + port se numește **socket**. TCP e sigur și cu conexiune; UDP e rapid și fără garanții. Alegerea depinde

de aplicație. Strângerea de mână are **trei** segmente; închiderea are **patru**, pentru că fiecare sens se închide separat. Un **SYN flood** umple coada de conexiuni pe jumătate deschise; apărarea sunt SYN cookies și limitarea ratei. În iptables, **lanțul** spune *unde*, **tabela** spune *ce fel de decizie*. **INPUT** = pentru mașină, **FORWARD** = prin mașină, **OUTPUT** = de la mașină. Politica corectă este DROP, iar regulile ACCEPT se pun **înaintea** ei. `--state ESTABLISHED,RELATED` rezolvă tot traficul de întoarcere într-o linie. **Autentificare, confidențialitate, integritate** sunt trei probleme distincte, cu trei mecanisme distincte. Diffie-Hellman rezolvă schimbul de chei, **nu** identitatea celui alt capăt.

### 8.13 Ce urmează

Cursul următor folosește exact aceleași unelte - ACL-uri pe Cisco, iptables pe Linux - dar pentru altceva decât filtrare: **translatarea adreselor** și construirea de **tuneluri**. Veți vedea acolo tabela **nat** pe care am amânat-o azi, și veți înțelege de ce o rețea privată poate naviga pe Internet cu o singură adresă publică.

**Laboratorul 7** aplică practic partea de securitate: SSH pe echipamente, plus apărarea împotriva atacurilor de nivel 2 din cursul 12.

- RFC 9293 - Transmission Control Protocol (versiunea consolidată a lui RFC 793)
- RFC 4987 - TCP SYN Flooding Attacks and Common Mitigations
- RFC 4251–4254 - arhitectura SSH
- Manualele `iptables(8)` și `iptables-extensions(8)`
- William Stallings, *Cryptography and Network Security* - capitolele despre Diffie-Hellman și MAC

### Întrebări de verificare

1. O regula iptables trebuie să blocheze traficul destinat chiar mașinii Linux. Ce lanț folosiți?
  - a) FORWARD
  - b) INPUT
  - c) OUTPUT
  - d) POSTROUTING
2. De ce deschiderea unei conexiuni TCP are trei segmente, iar închiderea patru?
  - a) Pentru că închiderea e mai lentă
  - b) Pentru că la deschidere SYN-ul și ACK-ul serverului încap în același segment, iar la închidere între ACK și FIN mai poate curge trafic
  - c) Pentru că închiderea folosește UDP
  - d) Este o convenție arbitrară
3. Ce exploatează un atac SYN flood?

- a) O eroare de programare din serverul web
  - b) Faptul ca serverul alocă resurse la pasul 2 al strangerii de mana, înainte sa primească ACK-ul final
  - c) Lipsa criptării
  - d) Dimensiunea mica a antetului TCP
4. Ce garantează schimbul Diffie-Hellman?
- a) Ca serverul este cine pretinde ca este
  - b) Ca cele doua capete obtin o cheie comuna fara ca aceasta sa fie transmisa pe canal
  - c) Ca mesajul nu a fost modificat
  - d) Ca parola este suficient de lunga
5. Ce diferentiaza un firewall stateful de unul stateless?
- a) Viteza de procesare
  - b) Faptul ca tine minte sesiunile stabilite si poate recunoaste traficul de intoarcere
  - c) Numarul de interfete
  - d) Faptul ca poate cripta traficul
6. In ce ordine se dau comenzile, cand configurati un firewall prin SSH?
- a) Intai politica DROP, apoi regulile ACCEPT
  - b) Intai regulile ACCEPT, apoi politica DROP
  - c) Nu conteaza ordinea
  - d) Intai se salveaza, apoi se configureaza
7. Ce face tinta LOG in iptables?
- a) Arunca pachetul si il inregistreaza
  - b) Inregistreaza pachetul si continua evaluarea lantului
  - c) Opreste conexiunea si trimite un RST
  - d) Salveaza regulile pe disc
8. De ce se prefera DROP in loc de REJECT pe interfata dinspre Internet?
- a) Este mai rapid de procesat
  - b) Pentru ca REJECT confirma atacatorului ca masina exista, iar DROP il obliga sa astepte expirarea temporizatorului
  - c) Pentru ca REJECT nu functioneaza pe TCP



- d) Pentru ca DROP inregistreaza automat in jurnal
9. De ce nu se cripteaza asimetric intreaga sesiune SSH?
- a) Pentru ca nu este sigura
  - b) Pentru ca este cu ordine de marime mai lenta decat criptarea simetrica
  - c) Pentru ca nu functioneaza pe fisiere mari
  - d) Pentru ca ar necesita doua chei publice

## 9. NAT și tunelare

---

---

- Din cursurile trecute
- Problema: de ce s-au terminat adresele
- Ce face, concret, translatarea
- Cele trei tipuri de translatare
- Cei patru termeni care încurcă pe toată lumea
- NAT pe echipamente Cisco
- NAT în Linux
- Ce ne costă NAT
- Conceptul de tunelare
- Tunelul GRE
- Tunelul SSH
- IPv6, pe scurt
- Tunelul 6to4
- Greșeli frecvente
- Rezumat și glosar
- Ce urmează
- Întrebări de verificare

---

---

IPv4 are patru miliarde de adrese, iar lumea are de mult mai multe ori mai multe dispozitive. Aritmetica aceasta ar fi trebuit să oprească Internetul prin anul 2000. Nu s-a întâmplat, iar cursul acesta explică de ce: translatarea adreselor a prelungit viața IPv4 cu peste două decenii, cu prețul unei încălcări deliberate a uneia dintre regulile fundamentale ale rețelelor. A doua jumătate a cursului este despre tunelare - arta de a face un protocol să circule printr-o infrastructură care nu îl înțelege, inclusiv IPv6 printr-un Internet care încă vorbește IPv4.

### 9.1 Din cursurile trecute

**Ce trebuie să aveți în minte.** Adresele IP sursă și destinație rămân **constante** de la un capăt la altul. Doar cele MAC se rescriu. Adresele **private** din RFC 1918 nu sunt rutabile în Internet. În iptables, **lanțul** spune unde pe traseu, iar **tabela** ce fel de decizie. **PREROUTING** este înainte de decizia de rutare, **POSTROUTING** după ea. Un **socket** este perechea adresă + port; o conexiune e identificată de patru numere.

Primul punct este cel pe care îl demolăm astăzi. Tot cursul 6 a insistat că adresele IP nu se schimbă pe traseu - și acum vom vedea un echipament care le schimbă intenționat, la

fiecare pachet, în ambele direcții. Nu e o contradicție: e o excepție construită cu grijă, care are propriile consecințe neplăcute.

### 9.1.1 Ce veți putea face după acest curs

#### Ce veți putea face după acest capitol.

- Să explicați de ce s-au epuizat adresele IPv4 și ce a amânat criza
- Să urmăriți un pachet prin translatare, dus și întors
- Să alegeți între NAT static, NAT dinamic și PAT
- Să folosiți corect termenii inside local, inside global, outside local, outside global
- Să configurați NAT pe Cisco și pe Linux
- Să enumerați ce strică NAT-ul și de ce
- Să explicați diferența dintre stratificare și tunelare
- Să scrieți și să citiți o adresă IPv6 și să explicați cum funcționează 6to4

## 9.2 Problema: de ce s-au terminat adresele

IPv4 folosește adrese de 32 de biți, deci aproximativ 4,3 miliarde de valori - din care o parte considerabilă este rezervată și nefolosibilă. Ultimele blocuri libere au fost distribuite de IANA în **februarie 2011**.

**Nu numărul a fost problema, ci risipa.** Alocarea inițială se făcea **pe clase**, iar clasele veneau în trei mărimi: 256, 65 536 sau 16 milioane de adrese. Nimic între ele. O organizație care avea nevoie de 300 de adrese nu încăpea într-o clasă C, deci primea o clasă B - și lăsa nefolosite 65 000 de adrese. Universități întregi au primit clase A, cu 16 milioane de adrese fiecare, în anii '80. Deci: patru miliarde de adrese, dintre care poate un sfert au ajuns efectiv la un dispozitiv. Restul s-au pierdut în rotunjiri.

Trei mecanisme au amânat criza, iar toate trei sunt încă în uz:

Tabela 9.1 Problema: de ce s-au terminat adresele.

| Mecanism                         | Ce face                                                                | Unde l-am văzut |
|----------------------------------|------------------------------------------------------------------------|-----------------|
| <b>CIDR</b>                      | elimină clasele; permite alocări de dimensiune potrivită (/22, /28...) | cursul 5        |
| <b>Adrese private</b> (RFC 1918) | trei blocuri re folosibile de oricâte organizații, în paralel          | cursul 5        |
| <b>NAT</b>                       | face adresele private utilizabile în Internet                          | <b>astăzi</b>   |

**De ce funcționează re folosirea.** Adresa 192.168.1.1 există, chiar acum, în milioane de locuințe simultan. Nu e nicio problemă, pentru că niciun ruter din Internet nu are rută spre ea: blocurile private sunt filtrate explicit la marginea oricărei rețele serioase.

Prețul: ca să iasă în Internet, aceste adrese trebuie **înlocuite** cu una publică. Exact asta face NAT.

Tabela 9.2 Problema: de ce s-au terminat adresele.

| Bloc privat                      | Prefix | Adrese        | Unde apare tipic              |
|----------------------------------|--------|---------------|-------------------------------|
| 10.0.0.0 -<br>10.255.255.255     | /8     | 16,7 milioane | rețele mari de întreprindere  |
| 172.16.0.0 -<br>172.31.255.255   | /12    | 1 milion      | rețele medii, laboratoare     |
| 192.168.0.0 -<br>192.168.255.255 | /16    | 65 536        | rețele mici, ruterul de acasă |

### 9.3 Ce face, concret, translatarea

#### Observație

**NAT - Network Address Translation.** Schimbarea adresei IP sursă sau destinație a unui pachet la trecerea printr-un ruter. În mod normal, adresele IP rămân neschimbate de la un capăt la altul; NAT încalcă **deliberat** această regulă. Pentru ca legătura să funcționeze, translatarea trebuie să aibă loc **în ambele direcții** - altfel răspunsul nu ar mai nimeri stația inițiatoare.

Ruterul ține evidența translatărilor într-o **tabelă NAT**, construită fie static de administrator, fie dinamic prin inspectarea traficului care trece.



Figura 9.1 Un pachet prin NAT, dus și întors.

### 9.4 Cele trei tipuri de translatare

#### 9.4.1 NAT static - unu la unu

**Problema:** un server are adresă privată, dar trebuie să fie accesibil din exterior printr-o adresă publică fixă.

**Soluția:** o asociere **permanentă** între adresa internă și una publică rezervată. Translatarea funcționează în ambele sensuri, deci conexiunile pot fi inițiate și dinspre Internet - este singurul tip la care asta merge fără configurări suplimentare.

### 9.4.2 NAT dinamic - n la m

**Problema:** 40 de stații interne, dar doar 20 de adrese publice disponibile.

**Soluția:** un **rezervor** (pool) de adrese publice, din care stațiile primesc temporar câte una, atât timp cât comunică. Când rezervorul se golește, stațiile următoare pur și simplu nu mai ies - comportament care se depanează greu, pentru că depinde de ora din zi. Conexiunile din exterior nu pot fi inițiate, pentru că nu se știe dinainte ce adresă are cine.

### 9.4.3 PAT - n la unu

**Problema:** 40 de stații interne și o singură adresă publică.

**Soluția:** **PAT** (Port Address Translation), numit și NAT overload la Cisco și **MASQUERADE** în Linux. Pe lângă adresă, ruterul translatează și **portul sursă**, folosindu-l ca identificator al conversației. Când răspunsul se întoarce, portul destinație spune ruterului cărei stații îi aparține.

STĂȚIE INTERNĂ: 192.168.1.10

SERVER EXTERN: 93.184.216.34:80

| INTERN                 | TRANSLATAT | DESTINAȚIE |
|------------------------|------------|------------|
| nicio traducere activă |            |            |

Ruterul păstrează portul tradus cât timp conexiunea e activă. Adresa publică este una singură: **86.120.5.7** — porturile sunt cele care deosebesc stațiile.

Figura 9.2 Simulator PAT: o singură adresă publică pentru toată rețeaua.

Deschideți mai multe conexiuni de la stații diferite și observați coloana din mijloc: adresa publică este mereu aceeași, dar portul diferă. Aceasta este întreaga idee.

ADRESE PUBLICE DISPONIBILE (BUC): 1

PORTURI UTILIZABILE PER ADRESĂ (PORTURI): doar 1024-65535 - 64512

CONEXIUNI SIMULTANE PER STĂȚIE (CONEXIUNI): 80

conversații simultane posibile **64512.00 conexiuni**  
 stații deservite simultan **806.00 stații**  
 adrese publice economisite **805.00 adrese**

Figura 9.3 Câte stații poate ascunde o adresă publică?.

Cu 16 biți de port, o singură adresă publică susține teoretic peste 64 000 de conversații - practic, câteva sute de stații, pentru că un browser modern deschide zeci de conexiuni pentru o singură pagină. Aceasta este, în cifre, explicația pentru care planeta a rămas pe IPv4 mult mai mult decât se estima în 1995.

Tabela 9.3 PAT - n la unu.

|                       | NAT static        | NAT dinamic                     | PAT                          |
|-----------------------|-------------------|---------------------------------|------------------------------|
| Raport adrese         | 1 : 1             | n : m                           | n : 1                        |
| Asocierea             | permanentă        | temporară, cât durează sesiunea | temporară, per conexiune     |
| Translatează portul   | nu                | nu                              | da                           |
| Inițiere din exterior | da                | nu                              | nu, fără port forwarding     |
| Utilizare tipică      | servere publicate | rare; a fost înlocuit de PAT    | practic orice rețea de acces |

## 9.5 Cei patru termeni care încurcă pe toată lumea

Documentația Cisco folosește patru denumiri pentru adresele implicate. Sunt logice odată înțelese, dar până atunci produc o confuzie remarcabilă.

**Cheia: două întrebări separate. Inside sau outside** răspunde la: *a cui este gazda?* A rețelei noastre, sau a Internetului? Se referă la **proprietar**, nu la poziția adresei. **Local** sau **global** răspunde la: *de unde privim?* Din interiorul rețelei, sau din Internet? Deci „inside local” înseamnă **gazda noastră, văzută dinăuntru**, iar „inside global” înseamnă **aceeași gazdă, văzută din Internet**. Sunt două nume pentru același calculator.

Tabela 9.4 Cei patru termeni care încurcă pe toată lumea.

| Termen                | A cui gazdă | Văzută de unde | Exemplu                                             |
|-----------------------|-------------|----------------|-----------------------------------------------------|
| <b>Inside local</b>   | a noastră   | dinăuntru      | 192.168.1.10 - adresa privată reală                 |
| <b>Inside global</b>  | a noastră   | din Internet   | 86.120.5.7 - adresa publică translatată             |
| <b>Outside global</b> | a lor       | din Internet   | 93.184.216.34 - adresa reală a serverului           |
| <b>Outside local</b>  | a lor       | dinăuntru      | de obicei <b>aceeași</b> ; diferă doar la NAT dublu |

**Regula practică.** În 95% din cazuri contează doar primele două, iar `show ip nat translations` le afișează exact în această ordine. Când citiți tabela, coloana „inside local” este stația reală, iar „inside global” este masca ei publică.

## 9.6 NAT pe echipamente Cisco

Configurarea are întotdeauna **trei** componente, iar lipsa oricăreia produce un NAT care nu funcționează și nu dă nicio eroare.

1. **Ce se translatează** - un ACL sau o asociere statică.
2. **În ce se translatează** - un rezervor de adrese, sau o interfață.

### 3. Care interfață e „inside” și care „outside” - pasul uitat cel mai des.

Exemplu 9.1 PAT: toată rețeaua, prin adresa interfeței de ieșire copiază

```

1
2 ! 1. ce se translateaza — un ACL obisnuit , dar aici permit
   ↪ inseamna
3 !   "acesta e traficul despre care vorbesc", nu "lasa-l sa treaca
   ↪ "
4 R1(config)# access-list 1 permit 192.168.1.0 0.0.0.255
5
6 ! 2. in ce se translateaza — adresa interfetei de iesire , oricare
   ↪ ar fi ea
7 R1(config)# ip nat inside source list 1 interface gigabitEthernet
   ↪ 0/1 overload
8
9 ! 3. cine e inauntru si cine e afara
10 R1(config)# interface gigabitEthernet 0/0
11 R1(config-if)# ip nat inside
12 R1(config)# interface gigabitEthernet 0/1
13 R1(config-if)# ip nat outside

```

Exemplu 9.2 NAT static: publicarea unui server intern copiază

```

1
2 ! serverul 192.168.1.100 este vizibil din Internet ca 86.120.5.8
3 R1(config)# ip nat inside source static 192.168.1.100 86.120.5.8
4
5 ! doar portul 80, daca nu vrem sa expunem tot serverul
6 R1(config)# ip nat inside source static tcp 192.168.1.100 80
   ↪ 86.120.5.8 80

```

```

R1# ?
Comenzi disponibile aici:
  show ip nat translations
  show ip nat statistics
  clear ip nat translation *
  show ip nat translations verbose
  debug ip nat

```

R1#

Comenzile de verificare pentru NAT. Prima este cea pe care o veți folosi la fiecare depanare.

Figura 9.4 Terminal: verificarea NAT.

Priviți ultima linie din prima comandă: are doar două coloane completate și niciun

protocol. Este asocierea **statică** - permanentă, existentă chiar dacă nimeni nu comunică. Celelalte patru sunt dinamice și dispar la expirarea temporizatorului.

#### Observație

**Simptomul clasic:** „NAT-ul nu merge deloc”. Dacă `show ip nat translations` este **gol** deși traficul circulă, aproape sigur lipsește pasul 3: interfețele nu au fost marcate `ip nat inside / ip nat outside`.

Dacă tabela are intrări, dar ping-ul tot nu merge, problema nu mai e la NAT - verificați ruta implicită și dacă furnizorul chiar rutează adresa publică folosită.

## 9.7 NAT în Linux

Translatarea se configurează cu aceleași unelte ca filtrarea, dar în tabela **nat**. Regula de aur este să știți în ce lanț intrați - și motivul este pur logic:

Tabela 9.5 NAT în Linux.

| Ce rescriem                | Lanț        | Țintă      | De ce acolo                                                                      |
|----------------------------|-------------|------------|----------------------------------------------------------------------------------|
| adresa <b>sursă</b>        | POSTROUTING | SNAT       | <b>după</b> decizia de rutare - ruta se alege pe baza adresei reale              |
| adresa <b>destinație</b>   | PREROUTING  | DNAT       | <b>înainte</b> de decizia de rutare - altfel ruterul ar ruta spre adresa greșită |
| sursa, cu IP-ul interfeței | POSTROUTING | MASQUERADE | când adresa publică este obținută prin DHCP și se poate schimba                  |

**De ce DNAT trebuie să fie înainte de rutare.** Pachetul sosește adresat lui 86.120.5.7 - adresa ruterului. Dacă ruterul ar decide ruta întâi, ar concluziona „e pentru mine” și l-ar da procesului local. Prea târziu. Rescriind destinația **înainte**, pachetul ajunge la decizia de rutare deja adresat lui 192.168.1.100, iar ruterul îl rutează normal spre LAN. Simetric, SNAT trebuie să fie **după**: dacă am rescrie sursa mai devreme, nu ar schimba nimic la rutare (ruta depinde de destinație), dar am pierde informația despre interfața pe care va ieși pachetul - exact informația de care MASQUERADE are nevoie.

Exemplu 9.3 NAT static, în ambele sensuri copiază

```

1
2 # iesire: rescrie sursa privata cu adresa publica rezervata
3 iptables -t nat -A POSTROUTING -s 192.168.1.100 -j SNAT -to-
    ↪ source 141.85.200.1
4
5 # intrare: rescrie destinatia publica cu adresa privata a
    ↪ serverului
```



```

6 iptables -t nat -A PREROUTING -d 141.85.200.1 -j DNAT --to-
  ↪ destination 192.168.1.100
7
8 # si, obligatoriu, activarea rutarii in nucleu
9 sysctl -w net.ipv4.ip_forward=1

```

**SNAT nu înseamnă „static NAT”.** Prescurtarea vine de la **Source NAT**. Este o confuzie care persistă în multe cursuri și în multe forumuri. Perechea ei este DNAT - **Destination NAT**. Un SNAT poate fi perfect dinamic.

---

#### Exemplu 9.4 NAT dinamic și PAT copiază

---

```

1
2 # rezervor de adrese publice
3 iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -j SNAT \
4     --to-source 141.85.200.2-141.85.200.6
5
6 # PAT cu adresa interfetei de iesire, oricare ar fi ea
7 iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

```

**SNAT sau MASQUERADE?.** SNAT cere adresa publică scrisă explicit - e mai rapid, pentru că nucleul nu trebuie să o caute la fiecare pachet. Se folosește când adresa este fixă.

MASQUERADE citește adresa curentă a interfetei de ieșire. E puțin mai lent, dar supraviețuiește unei schimbări de adresă - și de aceea e standardul pe conexiunile casnice, unde furnizorul dă adresa prin DHCP.

#### Exemplu

Este vreo problemă cu următorul set de reguli?

```

reguli iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -j SNAT --to-source
141.85.200.2-141.85.200.6 iptables -t nat -A POSTROUTING -s 192.168.1.100 -j SNAT
--to-source 141.85.200.1

```

**Rezolvare.** Da. A doua regulă **nu se va aplica niciodată**: adresa 192.168.1.100 face parte din 192.168.1.0/24, deci se potrivește cu prima regulă, care are țintă SNAT - o țintă terminală - și încheie procesarea lanțului. Soluția este inversarea ordinii: regula mai specifică deasupra. Este exact aceeași lecție ca la listele de acces din cursul 7, într-un context complet diferit: **ordinea regulilor este parte din semantica lor**. O veți întâlni a treia oară la rutare, sub numele de longest prefix match.

### 9.7.1 Port forwarding

Când un server intern trebuie expus doar pentru o anumită aplicație, se rescrie destinația pe baza portului. Este exact ce faceți în interfața ruterului de acasă când „deschideți un port”.

---

#### Exemplu 9.5 portul 80 al ruterului duce la serverul intern copiază

---

```

1
2 iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 \
3     -j DNAT --to-destination 192.168.1.100
4
5 # regula de filtrare trebuie sa permita si ea traficul!
6 iptables -A FORWARD -i eth0 -o eth1 -d 192.168.1.100 -p tcp --
    ↪ dport 80 -j ACCEPT
7
8 # necesar doar daca in retea exista mai multe gateway-uri:
9 iptables -t nat -A POSTROUTING -o eth1 -p tcp --dport 80 -d
    ↪ 192.168.1.100 \
10     -j SNAT --to-source 192.168.1.1

```

### Observație

**NAT nu înlocuiește filtrarea.** Tabela **nat** și tabela **filter** sunt independente. Un DNAT corect duce pachetul la serverul intern - dar dacă politica lanțului FORWARD este DROP și nu există o regulă explicită, pachetul e aruncat imediat după traducere. Simptom: `conntrack -L` arată conexiunea, dar serverul nu vede nimic.

## 9.8 Ce ne costă NAT

### Dezavantajele traducerii

- În cazul PAT, o stație din Internet **nu poate iniția** comunicația - ceea ce a rupt modelul peer-to-peer al Internetului original
- Folosește informație de **nivel 4** (portul) pentru a controla **nivelul 3** - o încălcare directă a stratificării
- Îngreunează protocoalele care transportă adrese IP **în interiorul datelor**: FTP activ, SIP, unele jocuri. Ruterul trebuie să inspecteze și să rescrie și încărcătura utilă, cu module dedicate (`nf_conntrack_ftp`)
- Gestionează dificil traficul **UDP**, care nu are noțiunea de conexiune, deci nici de sfârșit al ei - intrările expiră pe temporizator, uneori prea devreme
- **Rupe trasabilitatea**: în jurnalele unui server, sute de utilizatori apar cu aceeași adresă
- A **întârziat adoptarea IPv6**, oferind o soluție „suficient de bună” exact atunci când presiunea de migrare ar fi trebuit să crească

**CGNAT - când și adresa publică e împărțită.** Mulți furnizori nu mai dau nici măcar o adresă publică per abonat: pun un al doilea nivel de traducere în rețeaua lor (**Carrier-Grade NAT**), iar clientul primește o adresă din 100.64.0.0/10 - nici publică, nici privată clasică.

Consecința pentru dumneavoastră: port forwarding-ul de acasă **nu mai funcționează**, oricât l-ați configura corect. Ruterul dumneavoastră nu deține adresa publică pe care ar

trebui să o expună.

Merită spus și partea bună, pentru că se invocă des ca argument: NAT oferă, ca efect secundar, o formă de protecție - nimic nu ajunge la stațiile interne fără să fi fost cerut. Nu este însă un firewall și nu înlocuiește unul: nu filtrează nimic din ce iese, nu inspectează conținutul și nu vă apără de o stație internă compromisă.

## 9.9 Conceptul de tunelare

### Observație

**Tunelare.** Încapsularea datelor unui protocol (**payload protocol**) într-un alt protocol (**delivery protocol**), astfel încât primul să poată traversa o infrastructură care nu îl suportă.

**Când e stratificare și când e tunelare.** Faptul că IP încapsulează TCP, iar Ethernet încapsulează IP, **nu** este tunelare - este stratificarea normală, în care fiecare nivel îl poartă pe cel de deasupra. Tunelarea apare când un protocol este încapsulat într-unul de **același nivel sau de nivel mai înalt**: IPv6 în IPv4 (3 în 3), Ethernet în IP (2 în 3), TCP în SSH (4 în 7). Testul rapid: dacă în pachet apar **două antete de același tip** - două antete IP, de exemplu - este tunelare.

### Analogie

Vreți să trimiteți o scrisoare într-o limbă pe care poșta dintr-o anumită țară o refuză. O puneți într-un plic nou, cu adresa scrisă în limba acceptată, și o trimiteți unui prieten de acolo. El deschide plicul exterior și pune scrisoarea originală în poșta locală.

Poșta intermediară nu a știut niciodată ce transportă. Prietenul este capătul tunelului.

Tabela 9.6 Conceptul de tunelare.

| Tunel               | Transportă              | Prin                | Nivel | Scop                                    |
|---------------------|-------------------------|---------------------|-------|-----------------------------------------|
| <b>GRE</b>          | protocoale nivel 2 și 3 | de IPv4 sau IPv6    | 3     | legături logice între rutere            |
| <b>SSH</b>          | protocoale nivel 4      | de SSH              | 7     | transport securizat                     |
| <b>IPsec</b>        | IP                      | IP                  | 3     | VPN cu confidențialitate și integritate |
| <b>L2TP</b>         | PPP, ATM, Frame Relay   | UDP                 | 2     | legături PPP peste infrastructură IP    |
| <b>6to4, Teredo</b> | IPv6                    | IPv4, respectiv UDP | 3     | tranziția spre IPv6                     |

**Tunel  $\neq$  criptare.** GRE nu criptează absolut nimic: oricine vede traficul poate citi pachetul din interior. Tunelul rezolvă problema **conectivității**, nu pe cea a confidențialității.

Când e nevoie de amândouă, se combină: GRE peste IPsec este configurația clasică de VPN între sedii.

## 9.10 Tunelul GRE

GRE (Generic Routing Encapsulation) creează o legătură logică între două rutere care nu sunt vecine fizic. Ruterele intermediare nu știu nimic despre conținutul tunelului: pentru ele, este un simplu pachet IP adresat capătului celălalt.

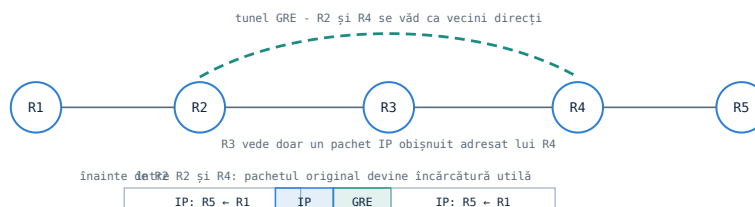


Figura 9.5 Încapsularea GRE. Pachetul original își păstrează adresele intacte; peste el se pun un antet GRE și un nou antet IP, cu adresele capetelor tunelului. Fiecare antet IP are propriul TTL, decrementat independent.

### Exemplu 9.6 configurarea unui tunel GRE, pe ambele capete copiază

```

1
2 ! pe R2
3 R2(config)# interface tunnel 0
4 R2(config-if)# ip address 172.16.0.1 255.255.255.252
5 R2(config-if)# tunnel source 203.0.113.2
6 R2(config-if)# tunnel destination 198.51.100.4
7 R2(config-if)# tunnel mode gre ip
8
9 ! pe R4 - oglinda exacta
10 R4(config)# interface tunnel 0
11 R4(config-if)# ip address 172.16.0.2 255.255.255.252
12 R4(config-if)# tunnel source 198.51.100.4
13 R4(config-if)# tunnel destination 203.0.113.2

```

De aici înainte, **tunnel 0** se comportă ca orice interfață: are adresă IP, apare în tabela de rutare, poate participa la OSPF. Ruterele R2 și R4 sunt, din punctul de vedere al rutării, **vecini direcți** - deși între ele sunt kilometri de Internet.

#### Observație

**Problema MTU, care apare mereu.** Antetele suplimentare - 20 de octeți IP plus 4 sau 8 de GRE - **consumă** din cei 1500 de octeți disponibili. Un pachet de 1500 de octeți nu mai încapă după încapsulare.

Simptomul e caracteristic și înșelător: ping-ul merge, sesiunile SSH merg, dar transferurile mari și paginile web mari se blochează. Pachetele mici trec, cele mari nu.

Soluția uzuală: `ip mtu 1400` și `ip tcp adjust-mss 1360` pe interfața de tunel.

### 9.11 Tunelul SSH

Un tunel SSH redirectează un port local printr-o sesiune SSH deja criptată. Scenariu tipic: Alice are cont pe un ruter Linux accesibil din Internet și vrea să administreze prin Telnet un server vechi din spatele lui, fără să își trimită parola în clar prin Internet.

Exemplu 9.7 redirectare locală de port copiază

```

1
2 ssh -N -L 0.0.0.0:5000:141.85.200.1:23 alice@141.85.200.19
3
4 #      -N nu deschide shell pe masina intermediara
5 #      -L tunel local: portul de intrare este pe masina locala
6 #      0.0.0.0:5000 adresa si portul pe care se asculta local
7 #      141.85.200.1:23 destinatia finala , vazuta din masina
      ↪ intermediara

```

După comandă, orice conexiune la `127.0.0.1:5000` de pe mașina lui Alice iese criptată spre serverul SSH și, de acolo, spre `141.85.200.1:23`. Aplicația lui Alice crede că vorbește cu un serviciu local.

Tabela 9.7 Tunelul SSH.

| Formă                           | Cine ascultă              | Scenariu                                        |
|---------------------------------|---------------------------|-------------------------------------------------|
| <code>-L port:gazdă:port</code> | mașina <b>locală</b>      | ajung eu la un serviciu din rețeaua îndepărtată |
| <code>-R port:gazdă:port</code> | mașina <b>îndepărtată</b> | expun un serviciu al meu spre acea rețea        |
| <code>-D port</code>            | local, ca proxy SOCKS     | tot traficul unei aplicații, prin tunel         |

#### Observație

**Unde se termină criptarea.** Traficul este protejat doar între mașina lui Alice și ruterul intermediar. De la ruter la serverul Telnet circulă **în clar**.

Un tunel securizează exact porțiunea pe care o traversează, niciun metru mai mult. Observația este valabilă pentru orice VPN - inclusiv pentru cele comerciale, care vă protejează până la serverul lor și nu mai departe.

### 9.12 IPv6, pe scurt

IPv6 nu este o versiune îmbunătățită a lui IPv4, ci un protocol nou, gândit să rezolve exact neajunsurile lui.

**Ce nu merge la IPv4**

- Adrese insuficiente
- Antet complicat, cu lungime variabilă
- Sumă de control recalculată la fiecare ruter
- Fragmentare făcută de rutere, costisitoare
- NAT introduce mai multe probleme decât rezolvă

**Ce aduce IPv6.** 128 de biți de adresă -  $3,4 \times 10^{38}$  valori Antet **fix** de 40 de octeți, rapid de procesat Fără sumă de control în antet - o face nivelul 4 Autoconfigurarea adreselor (SLAAC), fără DHCP Multicast simplificat; **broadcast-ul nu mai există**

**Cât de mari sunt 128 de biți.**  $3,4 \times 10^{38}$  adrese înseamnă aproximativ  $6 \times 10^{23}$  adrese pentru **fiecare metru pătrat** al suprafeței Pământului. Nu este o cifră care se poate epuiza prin risipă. De aceea IPv6 își permite ceva ce ar fi fost de neconceput la IPv4: fiecare segment de rețea primește un **/64** - adică 18 miliarde de miliarde de adrese - indiferent dacă are două stații sau două mii. Subnetarea nu mai este un exercițiu de economie.

### 9.12.1 Scrierea adreselor

O adresă IPv6 se scrie ca opt grupuri de câte patru cifre hexazecimale. Există două reguli de prescurtare: zerourile din **fața** fiecărui grup se pot omite, iar **un singur** șir continuu de grupuri nule se poate înlocui cu **::**.

```

ADRESĂ IPV6
2001:0db8:0000:0000:0000:0de8:7648:06e8

formă completă 2001:0db8:0000:0000:0000:0de8:7648:06e8
formă comprimată 2001:db8::de8:7648:6e8
prefixul /64 2001:db8::/64
tip global unicast

```

Figura 9.6 Comprimare și expandare de adrese IPv6.

Încercați **fe80::1, ::1** și **2002:8d55:c813::1** - ultimul este o adresă 6to4, pe care o veți recunoaște imediat după explicația din secțiunea următoare.

**De ce doar un singur .** Dacă ar fi permise două, adresa ar deveni ambiguă: **2001::25::1** nu ar spune câte grupuri nule merg în stânga și câte în dreapta. Cu un singur **::**, restul se deduce prin scădere din opt.

Convenția practică este ca primii 64 de biți să fie partea de rețea și ultimii 64 partea de interfață. Subnetarea funcționează identic cu IPv4 la nivel de bit, doar că spațiul e atât de mare încât se lucrează cu prefixe fixe: **/48** pentru un site, **/64** pentru un segment.

**Ce dispare odată cu IPv6.** Fără broadcast, **ARP nu mai există** - este înlocuit de Neighbor Discovery, care folosește multicast și deci nu deranjează toate stațiile din segment. Fără penurie de adrese, **NAT-ul nu mai are rost**, iar modelul peer-to-peer redevine posibil.

Ironia: tocmai NAT-ul, care a salvat IPv4, este motivul principal pentru care IPv6 a fost adoptat atât de încet.

Tabela 9.8 Scrierea adreselor.

| Prefix    | Tip            | Rol                                                                          |
|-----------|----------------|------------------------------------------------------------------------------|
| ::1/128   | loopback       | testarea propriei stive - echivalentul lui 127.0.0.1                         |
| 2000::/3  | global unicast | adrese rutabile în Internet                                                  |
| FE80::/10 | link-local     | comunicație în același segment; <b>generată automat</b> pe fiecare interfață |
| FC00::/7  | unique local   | echivalentul adreselor private din RFC 1918                                  |
| FF00::/8  | multicast      | transmisii către un grup                                                     |
| -         | broadcast      | <b>nu există</b> ; a fost înlocuit de multicast pe grupuri bine definite     |

### 9.13 Tunelul 6to4

Migrarea la IPv6 se face treptat: apar insule IPv6 într-o magistrală care rămâne IPv4. Tunelurile statice GRE funcționează, dar trebuie configurate manual, capăt cu capăt. Tunelul **6to4** se construiește automat, printr-un truc elegant de adresare.

1. Adresele insulei IPv6 trebuie să facă parte din 2002::/16.
2. Următorii 32 de biți ai adresei sunt **exact adresa IPv4 publică** a ruterului de la ieșirea insulei, scrisă hexazecimal. Pentru 141.85.200.19 aceasta înseamnă 8D:55:C8:13, deci prefixul 2002:8D55:C813::/48.
3. Când un ruter de capăt primește un pachet IPv6 spre o astfel de adresă, **extrage din ea** adresa IPv4 a destinației și încapsulează pachetul într-un pachet IPv4 spre acea adresă.
4. Ruterele intermediare rutează un pachet IPv4 obișnuit; câmpul Protocol are valoarea **41**, adică „IPv6 încapsulat”.
5. Ruterul de la celălalt capăt recunoaște valoarea 41, decapsulează și livrează pachetul IPv6 în insula lui.

#### Exemplu

Care este prefixul 6to4 al unui sit cu adresa publică 192.0.2.129?

**Rezolvare.** Convertim fiecare octet în hexazecimal:  $192 = C0 \cdot 0 = 00 \cdot 2 = 02 \cdot 129 = 81$  Grupăm câte doi octeți și lipim după 2002:: 2002:C000:0281::/48 Verificare: prefixul are 16 biți de 2002 plus 32 de biți de adresă IPv4 = 48. Rămân 16 biți pentru subrețelele sitului, adică 65 536 de segmente /64. Suficient.

**Ideea de reținut.** 6to4 nu are nevoie de nicio configurare a capătului celălalt, pentru că adresa IPv4 a destinației este **conținută în adresa IPv6**. Este un exemplu excelent de

proiectare în care informația de rutare este codificată chiar în identificador.

În practică, 6to4 a fost în mare parte abandonat (RFC 7526) în favoarea stivei duale - dar ideea rămâne instructivă.

## 9.14 Greșeli frecvente

### Observație

#### Greșeli frecvente.

- **„NAT-ul e configurat, dar tabela e goală"** Lipsesc marcajele `ip nat inside / ip nat outside` de pe interfețe. Fără ele, ruterul nu știe în ce direcție să translateze. Sunt al treilea pas, cel uitat.
- **„Am pus regula specifică după cea generală"** SNAT și DNAT sunt ținte terminale: prima potrivire câștigă și lanțul se oprește. Specificul deasupra generalului. Aceeași regulă ca la ACL-uri.
- **„Port forwarding-ul nu merge, deși DNAT-ul e corect"** Tabela `nat` a mutat pachetul, dar tabela `filter` l-a aruncat imediat după. Adăugați regula pe `FORWARD`. Cele două tabele nu se cunosc între ele.
- **„Tunelul GRE e sus, ping-ul merge, dar site-urile nu se încarcă"** Problemă de MTU: pachetele mici trec, cele mari nu încap după încapsulare. `ip mtu 1400` și `ip tcp adjust-mss 1360` pe interfața de tunel.
- **„Am crezut că tunelul îmi criptează traficul"** GRE nu criptează nimic. Nici 6to4, nici L2TP simplu. Pentru confidențialitate: IPsec, SSH sau WireGuard. Tunelul rezolvă conectivitatea, nu secretul.
- **„Am scris două :: într-o adresă IPv6"** Adresa devine ambiguă și e respinsă de orice implementare. Un singur `::`, pentru cel mai lung șir de grupuri nule.
- **„Port forwarding-ul de acasă nu merge deloc"** Furnizorul folosește CGNAT: adresa dumneavoastră „publică" e din `100.64.0.0/10`. Verificați ce adresă are interfața WAN a ruterului. Dacă e din acel bloc, soluția e o adresă publică de la furnizor sau un tunel către un server extern.

## 9.15 Rezumat și glosar

**Ce trebuie să rămână.** Criza IPv4 a venit din **risipa** alocării pe clase, nu doar din numărul de adrese. NAT rescrie adresele **în ambele direcții** și ține evidența într-o **tabelă**. **PAT** translatează și portul, deci n stații pot împărți o adresă publică. **Inside/outside** = a cui e gazda. **Local/global** = de unde privim. DNAT în **PREROUTING**, SNAT în **POSTROUTING** - din motive de ordine față de rutare. NAT rupe peer-to-peer, încurcă protocoalele cu adrese în date și nu este un firewall. **Tunelare** = încapsulare într-un protocol de același nivel sau mai înalt. Un tunel **nu criptează** automat; și securizează doar porțiunea pe care o traversează. IPv6 are 128 de biți, antet fix, fără broadcast și fără nevoie de NAT. **6to4** codifică adresa IPv4 chiar în adresa IPv6, deci tunelul se construiește singur.



### 9.16 Ce urmează

Cursul următor revine la rutare, dar la varianta care se întreține singură: protocoalele **link-state** și OSPF. Veți vedea de ce, într-o rețea cu douăzeci de rutere, rutele statice devin imposibil de administrat - și cum reușesc ruterele să își construiască singure o hartă completă a topologiei.

**Laboratorul 6** configurează ACL-uri, NAT și PAT pe ruterul de margine al firmei - exact combinația din ultimele două cursuri.

- RFC 3022 - NAT tradițional
- RFC 2784 - Generic Routing Encapsulation
- RFC 3056 - conectarea domeniilor IPv6 prin nori IPv4 (6to4)
- RFC 7526 - retragerea 6to4 din practica recomandată
- RFC 8200 - Internet Protocol, versiunea 6
- RFC 6598 - spațiul de adrese pentru CGNAT

### Întrebări de verificare

1. De ce PAT poate deservi zeci de mii de conexiuni cu o singura adresa publica?
  - a) Pentru ca adresele private sunt nelimitate
  - b) Pentru ca portul, pe 16 biti, devine identificator al fiecărei conversatii
  - c) Pentru ca ruterul comprima pachetele
  - d) Pentru ca foloseste IPv6 in fundal
2. In ce lant iptables se rescrie adresa destinatie si de ce?
  - a) POSTROUTING, pentru ca este ultimul
  - b) PREROUTING, pentru ca decizia de rutare trebuie luata pe baza adresei finale
  - c) INPUT, pentru ca pachetul intra in masina
  - d) FORWARD, pentru ca traficul tranziteaza
3. Ce inseamna inside global?
  - a) Adresa privata a unei statii din retea noastra
  - b) Adresa publica sub care apare in Internet o statie din retea noastra
  - c) Adresa unui server din Internet
  - d) Adresa de loopback a ruterului
4. show ip nat translations este gol, desi traficul circula. Ce verificati intai?
  - a) Daca ruterul are licenta de NAT
  - b) Daca interfetele au fost marcate cu ip nat inside si ip nat outside
  - c) Daca ACL-ul are deny la final

- d) Daca exista ruta implicita
5. Care este diferenta esentiala dintre stratificare si tunelare?
- a) Nu exista nicio diferenta
  - b) Tunelarea incapsuleaza un protocol intr-unul de acelasi nivel sau de nivel mai inalt, pentru a traversa o infrastruktura care nu il suporta
  - c) Tunelarea foloseste doar UDP
  - d) Stratificarea presupune criptare
6. Un tunel GRE este sus, ping-ul functioneaza, dar transferurile mari se blocheaza. Cauza probabila?
- a) Ruta implicita lipseste
  - b) MTU: antetele suplimentare nu mai lasa loc pachetelor de 1500 de octeti
  - c) Tunelul nu este criptat
  - d) Adresa de tunel este gresita
7. Ce prefix 6to4 corespunde adresei IPv4 publice 141.85.200.19?
- a) 2001:8D55:C813::/48
  - b) 2002:8D55:C813::/48
  - c) 2002:141:85:200::/48
  - d) FE80:8D55:C813::/48
8. Un tunel SSH intre statia A si ruterul R protejeaza traficul dintre R si serverul final?
- a) Da, tot traseul este criptat
  - b) Nu, criptarea exista doar intre A si R
  - c) Da, daca serverul are IPv6
  - d) Doar daca se foloseste UDP
9. De ce nu se poate scrie o adresa IPv6 cu doua grupuri :: ?
- a) Pentru ca ar fi prea lunga
  - b) Pentru ca adresa ar deveni ambigua: nu s-ar sti cate grupuri nule merg de fiecare parte
  - c) Pentru ca hexazecimalul nu permite
  - d) Se poate, dar doar in adrese link-local



## 10. Rutare dinamică

---

---

- Din cursul de rutare
- De ce nu ne ajung rutele statice
- Familiile de protocoale
- Distance-vector și problema buclelor
- Cum lucrează un protocol link-state
- Algoritmul lui Dijkstra
- OSPF: caracteristici de bază
- Mesajele și stările adiacenței
- DR și BDR pe rețele multiacces
- OSPF pe mai multe arii
- Configurare și verificare
- Greșeli frecvente
- Rezumat și glosar
- Ce urmează
- Întrebări de verificare

---

---

O rută statică este exactă, previzibilă și nu se schimbă niciodată - inclusiv atunci când legătura pe care o descrie a căzut acum zece minute. Într-o rețea cu trei rutere se poate trăi așa. Într-una cu douăzeci, fiecare cu câte trei ieșiri, numărul de linii de configurat depășește repede ce poate ține un om în minte, iar fiecare cablu scos cere intervenție manuală. Cursul acesta prezintă protocoalele care întrețin singure tabela de rutare: cum își descoperă ruterele vecinii, cum își construiesc o hartă comună a rețelei și cum calculează, fiecare pentru sine, drumurile cele mai scurte. Accentul cade pe familia link-state și pe reprezentantul ei standard, OSPF.

### 10.1 Din cursul de rutare

**Ce trebuie să aveți în minte.** O **rută** este tripletul destinație, direcție, distanță. **Distanța administrativă** spune cât de multă încredere are ruterul în **sursa** rutei. **Metrica** spune cât de bună este ruta, în interiorul aceleiași surse. Decizia se ia prin **longest prefix match**: prefixul mai lung câștigă, indiferent de AD. O rută statică are AD **1**; una direct conectată, **0**.

Astăzi umplem tabela altfel. Sursele noi de rute - RIP, OSPF, EIGRP - au fiecare propria distanță administrativă și propria metrică, iar ruterul trebuie să le compare pe toate. Ordinea

comparării este exact cea din cursul 6, și e bine să fie proaspătă în minte.

### 10.1.1 Ce veți putea face după acest curs

#### Ce veți putea face după acest capitol.

- Să explicați de ce rutele statice nu scalează și ce anume rezolvă un protocol dinamic
- Să distingeți între distance-vector, link-state și path-vector
- Să descrieți problema buclelor la distance-vector și mecanismele care o atenuează
- Să enumerați cei patru pași ai unui protocol link-state
- Să aplicați Dijkstra pe o topologie mică și să obțineți arborele drumurilor minime
- Să calculați costul OSPF al unei legături și să știți când formula dă rezultate greșite
- Să explicați de ce se aleg DR și BDR și cum decurge alegerea
- Să configurați OSPF single-area și să depanați o adiacență care nu se formează

### 10.2 De ce nu ne ajung rutele statice

**Aritmetica administratorului.** Într-o rețea cu  $n$  rutere, fiecare trebuie să știe drumul spre rețelele celorlalți. Dacă fiecare ruter are câte două rețele locale, sunt aproximativ  $2n$  destinații, iar fiecare dintre cele  $n$  rutere are nevoie de o linie pentru fiecare destinație care nu e a lui. Pentru 5 rutere: circa 40 de linii. Pentru 20 de rutere: circa **760**. Scrise de mână, în ambele sensuri, fără greșeli. Iar la fiecare rețea nouă adăugată undeva, trebuie modificate **toate** ruterele.

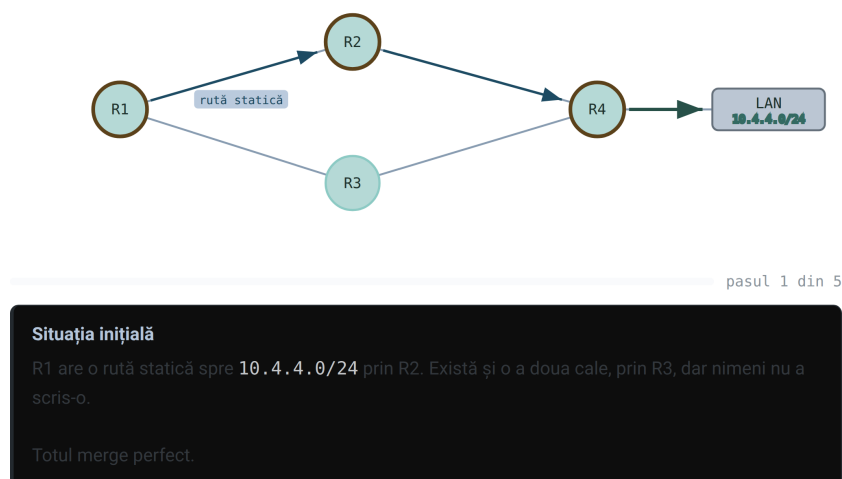


Figura 10.1 Ce se întâmplă când cade o legătură.

Cele două nu se exclud. Chiar și într-o rețea complet OSPF, ruta implicită spre furnizor este aproape întotdeauna statică - pentru că nu are cu cine să fie negociată.

### 10.3 Familiile de protocoale

Prima împărțire ține de **unde** se folosesc: în interiorul unei organizații, sau între organizații.

Tabela 10.1 De ce nu ne ajung rutele statice.

|                      | <b>Rutare statică</b>                            | <b>Rutare dinamică</b>                                   |
|----------------------|--------------------------------------------------|----------------------------------------------------------|
| Efort de configurare | crește cu pătratul dimensiunii                   | câteva linii per ruter, indiferent de mărime             |
| Reacție la defecte   | manuală                                          | automată, în secunde                                     |
| Consum de resurse    | zero                                             | procesor, memorie, bandă                                 |
| Previzibilitate      | totală                                           | depinde de starea rețelei                                |
| Securitate           | nimic de falsificat                              | anunțurile pot fi falsificate; e nevoie de autentificare |
| Unde se folosește    | rețele mici, rute implicite, legături de rezervă | orice rețea de dimensiune medie în sus                   |

Tabela 10.2 Familiile de protocoale.

| <b>Categorie</b>                       | <b>Unde</b>                       | <b>Protocoale</b>       |
|----------------------------------------|-----------------------------------|-------------------------|
| <b>IGP</b> - Interior Gateway Protocol | în interiorul unui sistem autonom | RIP, EIGRP, OSPF, IS-IS |
| <b>EGP</b> - Exterior Gateway Protocol | între sisteme autonome            | BGP                     |

**Observație**

**Sistem autonom (AS).** Un grup de rețele aflate sub o administrare unică, cu o politică de rutare comună. Fiecare AS are un număr unic la nivel mondial. Internetul este, la scară mare, o rețea de aproximativ 75 000 de sisteme autonome care își anunță reciproc rutele prin BGP.

A doua împărțire, cea care ne interesează astăzi, ține de **ce știe** fiecare ruter:

Tabela 10.3 Familiile de protocoale.

| <b>Familie</b>         | <b>Ce știe un ruter</b>                        | <b>Protocoale</b>         |
|------------------------|------------------------------------------------|---------------------------|
| <b>Distance-vector</b> | doar direcția și distanța, aflate de la vecini | RIP, IGRP, EIGRP (hibrid) |
| <b>Link-state</b>      | graful complet al topologiei                   | <b>OSPF</b> , IS-IS       |
| <b>Path-vector</b>     | lista completă de sisteme autonome traversate  | BGP                       |

**Analogie**

Un ruter

**distance-vector**

este ca un turist care întreabă trecătorii: „cât mai e până la gară?”. Primește un număr și o direcție, are încredere și pleacă. Nu vede orașul.

Un ruter

**link-state**

are harta oraşului în buzunar. Fiecare stradă închisă îi este comunicată, el o taie de pe hartă şi îşi recalculează singur traseul.

Un ruter

### path-vector

primeşte itinerarul complet: „prin Sibiu, apoi Cluj, apoi Oradea”. Poate refuza un drum pentru că trece printr-un oraş care nu îi place - şi asta e chiar ideea BGP-ului.

### Distance-vector

- Cunosc doar vectorul spre destinaţie
- Trimit **toată** tabela de rutare, periodic
- Converg lent
- Consumă puţine resurse
- Uşor de configurat
- Vulnerabile la bucle de rutare

**Link-state.** Cunosc graful întregii topologii Trimit doar **ce s-a schimbat**, când s-a schimbat Converg rapid Consumă memorie şi procesor Mai greu de configurat şi de proiectat Imune la buclele clasice, prin construcţie

| PROTOCOL | CARACTERISTICĂ DEFINITORIE                              |
|----------|---------------------------------------------------------|
| RIPv2    | path-vector, între sisteme autonome, AD 20 sau 200      |
| OSPF     | hibrid Cisco, metrică compusă, AD 90                    |
| EIGRP    | link-state, folosit de furnizorii mari                  |
| IS-IS    | distance-vector, metrică = număr de salturi, maximum 15 |
| BGP      | link-state, metrică din lăţimea de bandă, AD 110        |

Apasă un element din stânga, apoi perechea lui din dreapta.

Figura 10.2 Protocol şi caracteristică.

## 10.4 Distance-vector şi problema buclelor

Un protocol distance-vector, precum RIP, funcţionează pe un principiu simplu: fiecare ruter trimite periodic vecinilor **întreaga sa tabelă de rutare**, iar aceştia adaugă costul legăturii şi decid. Nimeni nu vede topologia; fiecare crede ce i se spune. De aici şi porecla care i s-a lipit definitiv: **routing by rumour** - rutare din zvonuri.

### 10.4.1 Numărătoarea spre infinit

## 10.5 Cum lucrează un protocol link-state

Ideea centrală este că fiecare ruter nu află **rutele** de la vecini, ci află **starea legăturilor** din întreaga reţea, îşi construieşte harta completă şi calculează singur drumurile. Patru

Tabela 10.4 Distance-vector și problema buclilor.

| Limitare          | Ce înseamnă practic                                                                                                                                 |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| Scalabilitate     | RIP consideră inaccesibilă orice destinație aflată la peste 15 salturi - o limită care într-o rețea mare devine reală                               |
| Convergență lentă | o schimbare se propagă din vecin în vecin, la fiecare interval de actualizare; pentru un diametru de 3 salturi și interval K, propagarea durează 3K |
| Vedere locală     | ruterul nu poate detecta o buclă pe care nu o vede                                                                                                  |
| Metrică simplistă | numărul de salturi ignoră complet lățimea de bandă: o cale prin trei legături de fibră pare mai proastă decât una printr-un singur modem            |

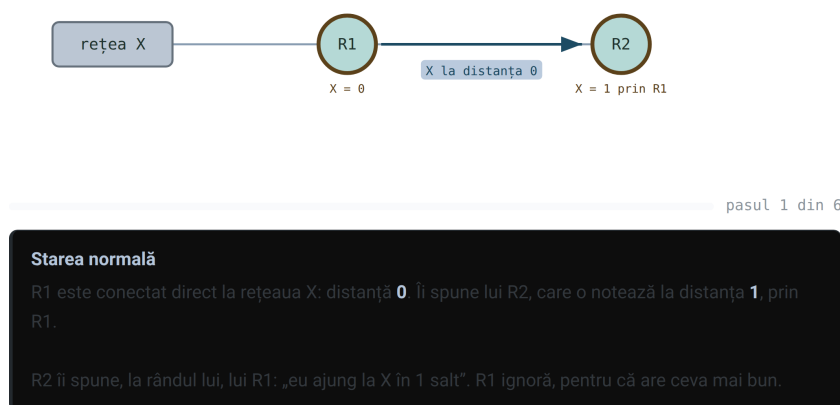


Figura 10.3 Numărătoarea spre infinit.

elemente fac asta posibil.

### 10.5.1 Adiacențele și protocolul Hello

Doi vecini direct conectați formează o **adiacență**. Relația se stabilește și se întreține prin mesaje Hello periodice, care au dublu rol: **keep-alive** - dacă mesajele încetează, vecinul e considerat căzut - și sincronizarea parametrilor. Vecinii cunoscuți se păstrează într-o **tabelă de vecini**.

Exemplu 10.1 tabela de vecini a lui R1

|   |                                                        |
|---|--------------------------------------------------------|
| 1 |                                                        |
| 2 | ID_R2 — Fa0/1 — 00:13 — FULL                           |
| 3 | ID_R3 — Fa0/2 — 00:00 — LOADING                        |
| 4 | ! interfata timp de la ultimul hello starea adiacentei |



**De ce Hello-ul rezolvă problema din secțiunea 2.** O rută statică nu are cum să afle că ruterul de la celălalt capăt a murit, atâta timp cât cablul e în priză și interfața e **up**. Un protocol dinamic nu se uită la cablu, ci la **conversație**. Dacă timp de 40 de secunde (temporizatorul *dead*) nu sosește niciun Hello, vecinul e declarat pierdut - indiferent ce spune starea interfeței. Este diferența dintre „firul e conectat” și „mai e cineva acolo”.

### 10.5.2 LSP: unitatea de informație

Un **LSP** (Link State Packet) descrie legăturile unui ruter: pentru fiecare interfață, rețeaua atașată, tipul legăturii, costul, starea ei și vecinul de la celălalt capăt. LSP-urile se trimit:

- la **inițializarea** procesului de rutare;
- imediat ce apare o **schimbare** în topologie - mecanismul de *triggered update*, sursa convergenței rapide;
- **periodic**, la interval mare (în OSPF, o dată la 30 de minute), ca reîmprospătare.

Imediat ce un ruter primește un LSP nou, îl retransmite tuturor celorlalți vecini. Procesul se numește **flooding** și asigură că, în câteva zecimi de secundă, toată rețeaua are aceeași informație.

**Interfețe care oscilează.** O interfață care se activează și se dezactivează repetat (*flapping*) ar declanșa recalculări continue ale topologiei, consumând tot procesorul tuturor rutelor din arie. De aceea orice protocol link-state așteaptă un scurt interval de stabilizare înainte de a recalcula - și de aceea o interfață instabilă e mai periculoasă decât una căzută de tot.

### 10.5.3 Cei patru pași

1. **Adiacențe și rețele conectate.** Ruterul își descoperă vecinii și rețelele direct conectate.
2. **Flooding de LSP-uri.** Fiecare ruter anunță ce știe și retransmite ce primește.
3. **Popularea tabelii de topologie.** Toate rețelele primite se păstrează, cu costurile lor - nu doar cele mai bune. Aceasta este harta.
4. **Dijkstra.** Se rulează algoritmul drumurilor minime, cu ruterul curent în vârf. Doar rezultatele ajung în tabela de rutare.

**Trei tabele, nu una.** Un ruter link-state ține **tabela de vecini** (cine e lângă mine și în ce stare), **tabela de topologie** (harta întregii arii, identică pe toate ruterele) și **tabela de rutare** (rezultatul lui Dijkstra, diferit pe fiecare ruter).

Dacă tabelele de topologie a două rutere din aceeași arie **diferă**, ceva e stricat - de regulă o adiacență care nu a ajuns niciodată la starea Full.

- 1 Interfața se activează și ruterul trimite Hello
- 2 Tabela de topologie conține acum harta completă
- 3 Rulează Dijkstra, cu ruterul propriu în vârf
- 4 Fiecare LSP primit este retransmis mai departe (flooding)
- 5 Cele mai bune căi se instalează în tabela de rutare
- 6 Se stabilește adiacența cu vecinul
- 7 Se schimbă LSP-uri, descriind legăturile proprii

Mută rândurile cu săgeți până ajung în ordinea corectă.

Figura 10.4 Puneți în ordine cei șapte pași.

## 10.6 Algoritmul lui Dijkstra

**Ideea, în trei propoziții.** Pornim din nodul propriu, cu distanța 0. Toate celelalte au, provizoriu, distanța infinit. La fiecare pas alegem nodul **nevizitat cu distanța cea mai mică** și îl declarăm definitiv - de acolo nu se mai poate ajunge mai ieftin. Apoi, prin el, încercăm să îmbunătățim distanțele vecinilor lui. Repetăm până se termină nodurile. Rezultatul este un **arbore** al drumurilor minime, cu noi în rădăcină.

Alegeți ruterul care calculează și observați că arborele obținut este **diferit pentru fiecare**: fiecare ruter se pune pe sine în vârf. Harta e aceeași; arborele nu.

Apoi modificați costul unei legături din arbore și urmăriți cum traficul se mută pe altă cale - exact ce se întâmplă într-o rețea reală când un administrator ajustează banda declarată a unei interfețe. Este principalul instrument de inginerie a traficului în OSPF: nu se schimbă cablurile, se schimbă costurile.

### Exemplu

Un ruter are două căi spre aceeași rețea: una prin trei legături de cost 1 fiecare, alta printr-o singură legătură de cost 4. Care câștigă și de ce nu numărul de salturi?

**Rezolvare.** Prima cale:  $1 + 1 + 1 = 3$ . A doua cale: **4**. Câștigă prima, deși are de trei ori mai multe salturi. Costul OSPF vine din **lățimea de bandă**. Cost 1 înseamnă 100 Mbps sau mai mult; cost 4 înseamnă 25 Mbps. Deci: trei legături rapide bat una lentă. Un protocol distance-vector clasic, care numără salturile, ar fi ales exact invers - și ar fi trimis tot traficul pe legătura de 25 Mbps. Este, în două rânduri, motivul pentru care metrica bazată pe bandă a înlocuit numărătoarea de salturi.

## 10.7 OSPF: caracteristici de bază

OSPF (Open Shortest Path First) a fost dezvoltat de IETF începând cu 1988, ca alternativă **deschisă** la protocoalele proprietare ale vremii. OSPFv2 (RFC 2328) este varianta pentru IPv4, OSPFv3 (RFC 5340) cea pentru IPv6.

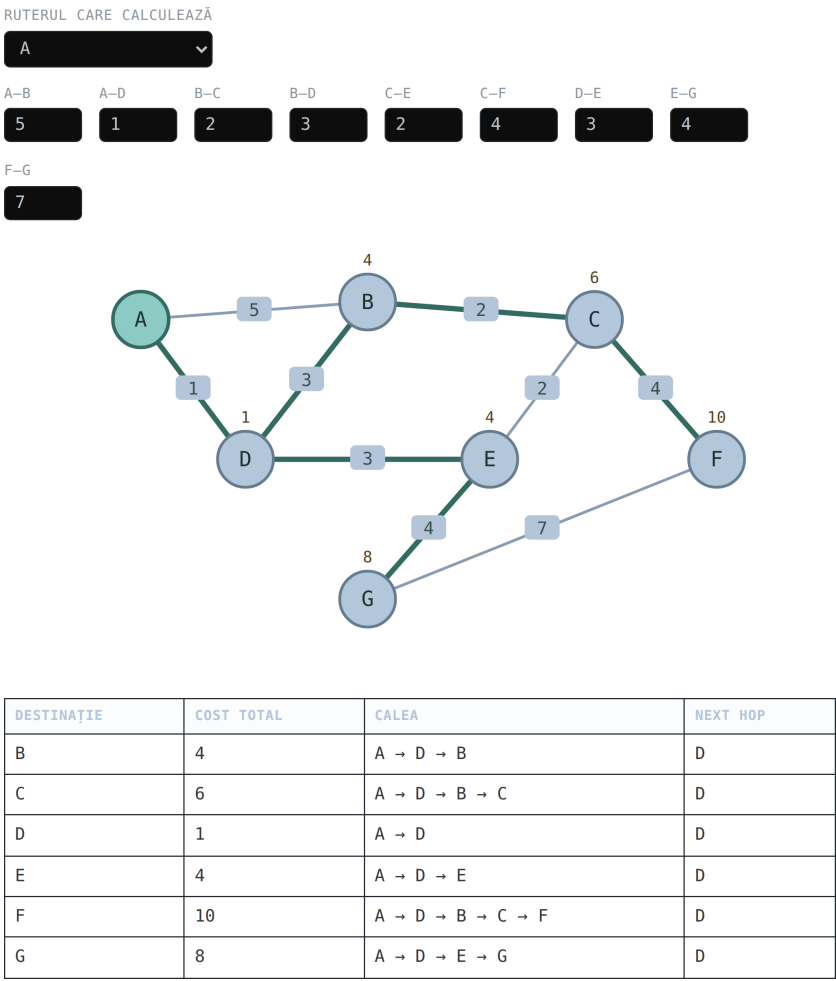


Figura 10.5 Algoritmul lui Dijkstra pe o topologie de șapte rutere.

Tabela 10.5 OSPF: caracteristici de bază.

| Caracteristică          | Valoare                                                                  |
|-------------------------|--------------------------------------------------------------------------|
| Transport               | direct peste IP, <b>protocol 89</b> - nu folosește TCP sau UDP           |
| Siguranța transmisiei   | mecanism propriu de confirmare (LSAck)                                   |
| Distanță administrativă | <b>110</b>                                                               |
| Adrese multicast        | 224.0.0.5 (toate ruterele OSPF), 224.0.0.6 (DR și BDR)                   |
| Metrică                 | cost = $10^8$ / lățimea de bandă în bps                                  |
| Timere hello / dead     | 10 s / 40 s pe rețele multiacces și punct-la-punct; 30 s / 120 s pe NBMA |
| Autentificare           | opțională, cu parolă simplă sau MD5 / SHA                                |

LĂȚIMEA DE BANDĂ A LEGĂTURII (Mbps) REFERENCE-BANDWIDTH (Mbps)

Fast Ethernet - 100 ▼ implicit - 100 ▼

cost brut (R / B) **1.00**

cost OSPF efectiv (minim 1) **1.00**

legături de acest tip pentru cost 10 **10.00 buc**

Figura 10.6 Costul OSPF al unei legături.

**Observație**

**Problema formulei implicite.** Cu referința implicită de  $10^8$  (adică 100 Mbps), orice legătură de 100 Mbps **sau mai rapidă** primește costul 1, pentru că minimul este 1. O legătură Fast Ethernet și una de 10 Gbps devin astfel **echivalente**, iar OSPF nu mai poate distinge între ele. Selectați în calculatorul de mai sus Gigabit cu referință implicită și veți vedea: cost 1. Selectați apoi referința de 10000 și diferența reapare. Într-o rețea modernă, referința se ajustează cu **auto-cost reference-bandwidth 10000** - **aceeași valoare pe toate ruterele din domeniu**, altfel costurile devin incoerente și rutarea imprevizibilă.

Tabela 10.6 OSPF: caracteristici de bază.

| Mediu                      | Cost cu referința implicită |
|----------------------------|-----------------------------|
| Serial 56 kbps             | 1785                        |
| T1 (1,544 Mbps)            | 64                          |
| Ethernet 10 Mbps           | 10                          |
| Fast Ethernet și mai rapid | 1                           |

**10.8 Mesajele și stările adiacenței**

Tabela 10.7 Mesajele și stările adiacenței.

| Mesaj                             | Rol                                                                          |
|-----------------------------------|------------------------------------------------------------------------------|
| <b>Hello</b>                      | descoperă vecinii și menține adiacențele                                     |
| <b>DBD</b> - Database Description | listă abreviată cu ce conține baza de date<br>link-state - un fel de cuprins |
| <b>LSR</b> - Link-State Request   | cere detalii despre o intrare văzută în DBD<br>și lipsă local                |
| <b>LSU</b> - Link-State Update    | răspunsul la LSR; conține unul sau mai<br>multe LSA-uri                      |
| <b>LSAck</b>                      | confirmă primirea unui LSU                                                   |

**De ce nu se trimite direct toată baza de date.** Ar fi risipă: cele două rutere au, de regulă, aproape aceleași informații. De aceea schimbul e în doi timpi: întâi **cuprinsul**

(DBD), apoi fiecare cere doar **capitolele care îi lipsesc** (LSR) și le primește (LSU). Este exact modelul unei sincronizări eficiente de fișiere.

Tabela 10.8 Mesajele și stările adiacenței.

| Stare          | Ce s-a întâmplat                                                                                                                 |
|----------------|----------------------------------------------------------------------------------------------------------------------------------|
| Down           | nu s-a primit niciun Hello                                                                                                       |
| Init           | ruterul a primit un Hello, dar nu se regăsește în el                                                                             |
| <b>Two-Way</b> | a primit un Hello în care <b>se regăsește propriul Router ID</b> - vecinul îl vede. Pe rețele multiacces, aici se aleg DR și BDR |
| Ex-Start       | se stabilește cine este master și cine slave pentru schimbul de baze de date                                                     |
| Exchange       | se schimbă DBD-urile                                                                                                             |
| Loading        | se cer și se primesc detaliile lipsă (LSR, LSU)                                                                                  |
| <b>Full</b>    | stare finală: bazele de date link-state sunt <b>identice</b>                                                                     |

1 Exchange - se schimbă cuprinsurile bazelor de date

2 Loading - se cer detaliile lipsă

3 Full - bazele de date sunt identice

4 Two-Way - ne vedem reciproc; se aleg DR și BDR

5 Down - tăcere completă

6 Init - am primit un Hello, dar nu mă vede

7 Ex-Start - se stabilește cine conduce schimbul

Mută rândurile cu săgeți până ajung în ordinea corectă.

Figura 10.7 Stările unei adiacențe, în ordine.

**Depanarea după starea în care s-a blocat.** Adiacența se formează doar dacă toți parametrii critici coincid: numărul **ariei**, intervalele **hello** și **dead**, **MTU**-ul interfeței, bitul de stub și autentificarea - iar Router ID-urile trebuie să fie **diferite**.

Blocată în **Init** → Hello-urile nu ajung în ambele sensuri: ACL, VLAN greșit, sau **passive-interface** pus din greșeală.

Blocată în **Two-Way** între două DROTHER-e → **normal**, nu e o problemă: pe un segment multiacces, două rutere obișnuite nu formează adiacență completă.

Blocată în **Exchange** sau **Ex-Start** → aproape sigur **MTU diferit** între capete.

Oscilează între Full și Down → legătură instabilă sau temporizatoare nepotrivite.

## 10.9 DR și BDR pe rețele multiacces

Pe o legătură punct-la-punct există o singură adiacență posibilă și nicio problemă. Pe un segment Ethernet cu  $n$  rutere, dacă fiecare ar forma adiacență cu fiecare, ar rezulta  $n(n-1)/2$  relații - 45 pentru zece rutere, fiecare cu propriile mesaje Hello, propriile sincronizări și propriile retransmiteri.

Soluția este centralizarea: se alege un **DR** (Designated Router), care primește și redistribuie actualizările, și un **BDR** (Backup DR), care preia dacă DR cade. Toate celelalte rutere (**DROTHER**) formează adiacență doar cu DR și BDR. Numărul de adiacențe scade la  $2(n-1)$ .

Tabela 10.9 DR și BDR pe rețele multiacces.

| Rutere pe segment | Fără DR | Cu DR și BDR | Economie |
|-------------------|---------|--------------|----------|
| 5                 | 10      | 8            | 20%      |
| 10                | 45      | 18           | 60%      |
| 20                | 190     | 38           | 80%      |
| 50                | 1225    | 98           | 92%      |

### 10.9.1 Criteriile de alegere

1. **Prioritatea cea mai mare** de pe interfață (implicit 1). Valoarea **0** exclude ruterul din alegere.
2. La egalitate, **Router ID-ul cel mai mare**.

Router ID-ul se determină, la rândul lui, în trei pași, în ordine strictă:

1. valoarea configurată manual cu `router-id`;
2. cea mai mare adresă IP de pe o interfață de **loopback**;
3. cea mai mare adresă IP de pe o interfață fizică **activă**.

#### Observație

**Alegerea nu este preemptivă.** Un ruter nou, chiar cu prioritate mai mare, **nu** preia rolul de DR. Dacă DR cade, BDR îi ia locul și se alege un nou BDR dintre DROTHER-e. Consecință practică: dacă vrei ca un anumit ruter să fie DR, porniți-l primul sau forțați o realegere cu `clear ip ospf process`.

Consecință și mai practică: folosiți **interfețe de loopback** pentru Router ID. Ele nu cad niciodată, deci identitatea ruterului nu se schimbă când pică un cablu.

```

2 ! ruterul care trebuie sa fie DR
3 R1(config)# interface gigabitEthernet 0/0
4 R1(config-if)# ip ospf priority 100
5
6 ! ruterul care nu trebuie sa fie niciodata DR sau BDR
7 R3(config)# interface gigabitEthernet 0/0
8 R3(config-if)# ip ospf priority 0
9
10 ! Router ID stabil , printr-o interfata care nu cade
11 R1(config)# interface loopback 0
12 R1(config-if)# ip address 1.1.1.1 255.255.255.255
13 R1(config)# router ospf 1
14 R1(config-router)# router-id 1.1.1.1
15
16 ! alegerea nu e preemptiva – trebuie fortata
17 R1# clear ip ospf process

```

### 10.10 OSPF pe mai multe arii

Dijkstra rulează cu atât mai lent cu cât graful e mai mare, iar orice schimbare într-un colț al rețelei obligă **toate** ruterele să recalculeze totul. Soluția de scalare este împărțirea domeniului OSPF în **arii**: fiecare arie rulează Dijkstra pe propriul graf, iar pentru destinațiile din afara ei doar adună distanța până la ruterul de frontieră.

**Regula de aur a ariilor.** Toate ariile trebuie să aibă conectivitate la **aria 0** (backbone). Traficul dintre două arii non-zero trece **obligatoriu** prin backbone.

Când o arie nu poate fi conectată fizic la aria 0, se configurează o legătură virtuală - sau, mai simplu și mai onest, un tunel GRE, ca în cursul trecut.

Tabela 10.10 OSPF pe mai multe arii.

| Tip de ruter                                  | Poziție                                                                                             |
|-----------------------------------------------|-----------------------------------------------------------------------------------------------------|
| <b>Internal router</b>                        | toate interfețele într-o singură arie                                                               |
| <b>Backbone router</b>                        | ruter cu cel puțin o interfață în aria 0                                                            |
| <b>ABR</b> - Area Border Router               | face legătura între două arii; are sincronizate bazele de date ale ambelor                          |
| <b>ASBR</b> - Autonomous System Border Router | introduce în OSPF rute externe, de exemplu prin redistribuire din alt protocol sau o rută implicită |

Tabela 10.11 Tipurile de LSA.

| Tip | Denumire         | Generat de             | Ce descrie                                        |
|-----|------------------|------------------------|---------------------------------------------------|
| 1   | Router LSA       | fiecare ruter          | starea propriilor legături, în interiorul ariei   |
| 2   | Network LSA      | DR                     | ruterele cu care DR are adiacență pe acel segment |
| 3   | Summary LSA      | ABR                    | rutele dintr-o arie, anunțate celorlalte arii     |
| 4   | ASBR Summary LSA | ABR                    | cum se ajunge la un ASBR                          |
| 5   | External LSA     | ASBR                   | rute externe domeniului OSPF                      |
| 7   | NSSA External    | ASBR dintr-o arie NSSA | rute externe într-o arie care nu acceptă tipul 5  |

Tabela 10.12 Codurile din tabela de rutare.

| Cod         | Provine din | Semnificație                                                                                 |
|-------------|-------------|----------------------------------------------------------------------------------------------|
| 0           | LSA 1 și 2  | rută din <b>aceeași</b> arie                                                                 |
| 0 IA        | LSA 3       | rută <b>inter-arie</b>                                                                       |
| 0 E1 / 0 E2 | LSA 5       | rute externe; <b>E1</b> acumulează costul intern, <b>E2</b> păstrează cost fix (implicit 20) |
| 0 N1 / 0 N2 | LSA 7       | rute externe într-o arie NSSA                                                                |

### 10.10.1 Tipurile de LSA

### 10.10.2 Codurile din tabela de rutare

Când există atât o rută E1, cât și una E2 spre aceeași destinație, se preferă **E1**: fiind cumulativă, reflectă mai fidel costul real al drumului. E2 este implicit tocmai pentru că e mai simplu - dar e și mai puțin exact.

## 10.11 Configurare și verificare

Exemplu 10.3 OSPFv2 într-o singură arie copiază

```

1
2 R1(config)# router ospf 1
3 R1(config-router)# router-id 1.1.1.1
4
5 ! ce interfete participa — masca este WILDCARD, ca la ACL-uri
```



```

6 R1(config-router)# network 192.168.10.0 0.0.0.255 area 0
7 R1(config-router)# network 10.0.0.0 0.0.0.3 area 0
8
9 ! interfata spre statii: anunta reseaua, dar nu trimite Hello
10 R1(config-router)# passive-interface gigabitEthernet 0/0
11
12 ! ajustarea referintei - ACEEASI valoare pe toate ruterele
13 R1(config-router)# auto-cost reference-bandwidth 10000
14
15 ! ruta implicita spre furnizor, anuntata in tot domeniul OSPF
16 R1(config)# ip route 0.0.0.0 0.0.0.0 203.0.113.1
17 R1(config-router)# default-information originate

```

**Numărul de proces este local.** router ospf 1 pe R1 și router ospf 55 pe R2 formează perfect adiacență. Numărul identifică procesul **pe acel ruter**, nu domeniul de rutare. Ce trebuie să coincidă este numărul **ariei**.

**passive-interface** se pune pe orice interfață pe care nu există alt ruter: rețeaua rămâne anunțată în OSPF, dar nu se mai trimit mesaje Hello inutile spre stații - ceea ce economisește bandă și închide o ușă de atac.

```

R1# ?
Comenzi disponibile aici:
  show ip ospf neighbor
  show ip ospf interface brief
  show ip route ospf
  show ip ospf database
  show ip protocols
  show ip ospf
  clear ip ospf process

```

R1#

Ordinea de depanare: întâi vecinii, apoi interfețele, apoi baza de date, apoi rutele.

Figura 10.8 Terminal: verificarea OSPF.

Priviți prima comandă: a treia linie arată **2WAY/DROTHER** cu prioritate 0. Nu este o eroare - este exact ce trebuie să se întâmple între două rutere obișnuite pe un segment multiacces. Doar adiacențele cu DR și BDR ajung la **FULL**.

Iar în tabela de rutare se citesc, dintr-o privire, cele trei origini: **0** din aceeași arie cu cost 2, **0 IA** din altă arie cu cost 12, și **0 E2** externă, cu costul fix 20 care nu se schimbă oricât ar fi drumul intern.

## 10.12 Greșeli frecvente

### Observație

#### Greșeli frecvente.

- „**Adiacența e blocată în Exchange**” Cele două capete au MTU diferit și refuză să sincronizeze bazele de date. `show ip ospf interface` pe ambele, comparați MTU-ul. Sau `ip ospf mtu-ignore`, ca soluție de avarie.
- „**Vecinii sunt în Two-Way și nu trec mai departe**” Dacă sunt două DROTHER-e pe un segment multiacces, este comportamentul **corect**. Verificați cine e DR. Adiacențele complete există doar cu DR și BDR.
- „**Am pus reference-bandwidth doar pe un ruter**” Costurile devin incoerente: același drum are valori diferite văzute din capete diferite. Aceeași valoare pe **toate** ruterele din domeniu. Fără excepție.
- „**Am folosit masca de rețea în comanda network**” OSPF cere **wildcard**, exact ca ACL-urile. `network 10.0.0.0 0.0.0.3 area 0`, nu `255.255.255.252`.
- „**Am uitat passive-interface pe segmentele cu stații**” Ruterul trimite Hello-uri spre utilizatori: bandă irosită și o ușă de atac deschisă - oricine poate porni un proces OSPF fals. `passive-interface default`, apoi `no passive-interface` doar unde chiar există vecini.
- „**Ruterul nou nu devine DR, deși are prioritate 200**” Alegerea nu este preemptivă. Nu e o defecțiune. `clear ip ospf process`, dacă chiar aveți nevoie de realegere.
- „**Am configurat OSPF, dar ruta implicită nu ajunge la ceilalți**” O rută statică nu intră singură în OSPF. `default-information originate` pe ruterul de margine.

## 10.13 Rezumat și glosar

**Ce trebuie să rămână.** Rutele statice nu **scalează** și nu **reacționează**. Ambele probleme le rezolvă un protocol dinamic. **Distance-vector** = zvonuri de la vecini; **link-state** = hartă proprie. Numărătoarea spre infinit e problema fundamentală a DV; split horizon și poison reverse o atenuează. Cei patru pași link-state: adiacențe → flooding → topologie → Dijkstra. Un ruter link-state ține **trei** tabele: vecini, topologie, rutare. OSPF: protocol 89, AD **110**, cost =  $10^8$  / bandă, multicast 224.0.0.5. Adiacența trece prin Init → Two-Way → Ex-Start → Exchange → Loading → **Full**. **DR** și **BDR** reduc adiacențele de la  $n(n-1)/2$  la  $2(n-1)$ ; alegerea **nu** e preemptivă. Toate ariile se conectează la **aria 0**. În comanda `network`, masca este **wildcard**.

## 10.14 Ce urmează

Cursul următor părăsește firul și trece în aer: rețele wireless, cu problemele lor specifice de acces la mediu - nu se poate asculta și transmite simultan, deci CSMA/CD nu funcționează - și cu o problemă de securitate pe care rețeaua cablată nu o are: mediul este public prin

definiție.

**Laboratorul 5** configurează OSPF single-area pe exact aceeași topologie pe care ați scris, mai devreme, rute statice. Compararea dintre cele două configurații, ca număr de linii și ca reacție la un cablu scos, este scopul lucrării.

- RFC 2328 - OSPF versiunea 2
- RFC 5340 - OSPF pentru IPv6
- RFC 3630 - extensii de inginerie a traficului pentru OSPF
- RFC 2453 - RIP versiunea 2, pentru comparație
- John Moy, *OSPF: Anatomy of an Internet Routing Protocol*

### Întrebări de verificare

1. Ce pastreaza un ruter link-state in tabela de topologie?
  - a) Doar cea mai buna ruta spre fiecare destinatie
  - b) Toate rutele primite prin LSP-uri, cu costurile lor
  - c) Doar rutele statice
  - d) Adresele MAC ale vecinilor
2. De ce limita RIP este de 15 salturi?
  - a) Pentru ca antetul are doar 4 biti pentru metrica
  - b) Pentru ca 16 inseamna infinit, iar limita opreste numaratoarea spre infinit intr-un timp rezonabil
  - c) Pentru ca ruterele nu pot stoca mai mult
  - d) Este o alegere arbitrara
3. Ce face split horizon?
  - a) Imparte reseaua in doua jumatati
  - b) Nu anunta o ruta inapoi pe interfata de la care a fost aflata
  - c) Crijteaza actualizarile de rutare
  - d) Reduce numarul de adiacente
4. De ce se alege un DR pe un segment Ethernet cu mai multe rutere OSPF?
  - a) Pentru ca doar el poate ruta
  - b) Ca sa reduca numarul de adiacente de la  $n(n-1)/2$  la  $2(n-1)$
  - c) Pentru ca celelalte rutere nu au Router ID
  - d) Ca sa cripteze mesajele
5. Un ruter nou, cu prioritate 200, este adaugat pe un segment unde exista deja un DR cu prioritate 1. Ce se intampla?

- a) Devine imediat DR
  - b) Nu se intampla nimic - alegerea DR nu este preemptiva
  - c) Devine BDR automat
  - d) Adiacentele se rup
6. Doua rutere OSPF raman blocate in starea Exchange. Cauza cea mai probabila?
- a) Router ID-uri identice
  - b) MTU diferit pe cele doua interfete
  - c) Prioritati diferite
  - d) Numere de proces diferite
7. Cu referinta implicita de  $10^8$ , ce cost OSPF au o legatura de 100 Mbps si una de 10 Gbps?
- a) 1 si respectiv 0,01
  - b) Ambele 1 - OSPF nu le poate distinge
  - c) 10 si respectiv 1
  - d) 64 si respectiv 1
8. Prin ce arie trece obligatoriu traficul dintre aria 10 si aria 20?
- a) Direct, fara arie intermediara
  - b) Prin aria 0, backbone-ul
  - c) Prin aria cu numarul cel mai mic
  - d) Depinde de protocolul de redistribuire
9. Ce masca se foloseste in comanda network a lui OSPF?
- a) Masca de retea obisnuita
  - b) Masca wildcard, ca la listele de acces
  - c) Nu se foloseste masca
  - d) Prefixul CIDR



# 11. Rețele wireless

---

---

- Din cursul de nivel 2
- Fizica din spatele deciziilor
- Benzile libere și cine le mai folosește
- Evoluția standardelor 802.11
- Spreading și acces multiplu
- Planificarea canalelor
- Organizarea unei rețele wireless
- Accesul la mediu
- Cadrul 802.11
- Două probleme care nu există pe cablu
- Securitatea rețelelor wireless
- Greșeli frecvente
- Rezumat și glosar
- Ce urmează
- Întrebări de verificare

---

---

Într-o rețea pe cablu, mediul este al tău și nimeni nu îl vede. Într-o rețea fără fir totul se schimbă: mediul este public, oricine din parcare poate asculta, două stații pot să nu se audă între ele deși vorbesc amândouă cu același punct de acces, iar coliziunile nu pot fi detectate - nici măcar în principiu. Fiecare dintre aceste probleme a cerut o soluție proprie, iar cursul acesta le ia pe rând, de la fizica undelor până la protocolul care hotărăște cine are voie să vorbească și la criptarea care ține parcare afară.

## 11.1 Din cursul de nivel 2

**Ce trebuie să aveți în minte.** Pe un mediu **partajat**, două stații care transmit simultan produc o **coliziune**. **CSMA/CD** rezolvă asta: ascultă, transmite, iar dacă detectezi coliziune, oprește-te și reîncearcă după un timp aleator. Un **domeniu de coliziune** este zona în care două transmisii se pot ciocni. Adresa MAC identifică o interfață; cadrul are sursă, destinație și CRC. **Modulația** transformă biții într-un semnal transmisibil pe mediu (cursul 1).

Astăzi vom vedea că CSMA/CD, mecanismul care a făcut Ethernet-ul posibil, **nu poate fi folosit** în aer. Nu pentru că cineva a decis altfel, ci pentru că fizica nu permite. Iar din această imposibilitate decurge aproape tot ce e specific la 802.11.

### 11.1.1 Ce veți putea face după acest curs

#### Ce veți putea face după acest capitol.

- Să explicați compromisul dintre frecvență, acoperire și debit
- Să spuneți ce a crescut debitul de la 2 Mbps la peste 10 Gbps, pe patru pârgii
- Să distingeți FHSS, DSSS, OFDM și OFDMA
- Să planificați canalele într-o clădire cu mai multe puncte de acces
- Să folosiți corect termenii BSS, IBSS, ESS, SSID și BSSID
- Să explicați de ce CSMA/CD nu funcționează în aer și cum îl înlocuiește CSMA/CA
- Să descrieți problema stației ascunse și modul în care RTS/CTS o rezolvă
- Să alegeți mecanismul de securitate potrivit și să spuneți de ce cele „simple” nu sunt securitate

### 11.2 Fizica din spatele deciziilor

Rețelele wireless folosesc **microunde**, o felie din spectrul electromagnetic aflată aproximativ între 1 și 10 GHz. Alegerea nu e arbitrară: rezultă dintr-un compromis cu cinci termeni, pe care merită să îl aveți în minte de fiecare dată când cineva propune „o bandă mai bună”.

Tabela 11.1 Fizica din spatele deciziilor.

| Proprietate                 | Frecvențe mici             | Frecvențe mari |
|-----------------------------|----------------------------|----------------|
| Distanța de propagare       | <b>mare</b>                | mică           |
| Penetrarea pereților        | <b>bună</b>                | slabă          |
| Interferențe                | multe - banda e aglomerată | <b>puține</b>  |
| Lățime de bandă disponibilă | mică                       | <b>mare</b>    |
| Costul echipamentelor       | <b>mic</b>                 | mare           |

**De ce frecvențele mari trec mai greu prin pereți.** Lungimea de undă este invers proporțională cu frecvența. La 2,4 GHz ea este de circa **12,5 cm**; la 5 GHz, de circa **6 cm**. O undă „ocolește” mai ușor obstacolele mai mici decât lungimea ei de undă și este absorbită sau reflectată de cele comparabile ori mai mari. Un perete de 15 cm este, pentru unda de 12,5 cm, un obstacol pe care îl mai poate negocia; pentru cea de 6 cm, e un zid în toată puterea cuvântului. Același motiv explică de ce radioul FM prinde în tunel, iar Wi-Fi-ul nu: la 100 MHz lungimea de undă e de trei metri.

Comparați 2,4 și 5 GHz la aceeași distanță: pierderea crește cu circa 6,4 dB doar din schimbarea frecvenței, fără niciun perete în plus. Aceasta este, în cifre, explicația pentru care banda de 5 GHz „nu ajunge la fel de departe” - deși puterea emisă e identică. Iar calculul de mai sus este pentru **spațiu liber**: într-o clădire reală se mai adaugă 3–5 dB per perete de gips-carton și 10–15 dB per perete de beton.



Figura 11.1 Lungime de undă și buget de semnal.

Tabela 11.2 Benzile libere și cine le mai folosește.

| Tehnologie            | Bandă               | Aplicație                                               |
|-----------------------|---------------------|---------------------------------------------------------|
| LoRaWAN               | 868 MHz (UE)        | IoT pe distanțe mari, consum foarte mic, debit minuscul |
| Zigbee                | 868 MHz și 2,4 GHz  | automatizări, senzori, becuri inteligente               |
| Bluetooth             | 2,4000 – 2,4835 GHz | periferice, distanțe scurte                             |
| Wi-Fi                 | 2,4 / 5 / 6 GHz     | rețele locale fără fir                                  |
| Cuptorul cu microunde | 2,45 GHz            | nu comunică nimic, dar emite cu sute de wați            |

### 11.3 Benzile libere și cine le mai folosește

Toate acestea funcționează în **benzi ISM** (Industrial, Scientific and Medical), utilizabile fără licență și fără taxă, cu respectarea unor limite de putere. În România, alocările sunt reglementate de **ANCOM**.

**Paradoxul benzilor libere.** Faptul că banda este liberă explică **simultan** succesul enorm al Wi-Fi și aglomerația din ea. Oricine poate emite acolo, deci echipamentele sunt ieftine și se vând peste tot - și, din exact același motiv, într-un bloc de locuințe banda de 2,4 GHz este astăzi practic inutilizabilă la debit maxim.

Nu există administrator al benzii. Nu se poate reclama nimănui. Singura soluție e să treceți la 5 sau 6 GHz, unde e mai puțină lume - deocamdată.

### 11.4 Evoluția standardelor 802.11

Creșterea de la 2 Mbps la 30 Gbps în mai puțin de trei decenii nu vine dintr-o singură invenție, ci din **patru pârgii** acționate simultan - toate cunoscute deja din cursul 1.

1. **Modulații tot mai dense.** De la BPSK și QPSK la 256-QAM (802.11ac) și 4096-QAM (802.11be). Fiecare pas crește numărul de biți per simbol - și cere, în schimb, un raport semnal-zgomot mai bun. De aceea debitul maxim se obține doar lângă punctul de acces.
2. **MIMO.** Mai multe antene care transmit simultan fluxuri spațiale independente. Patru



Tabela 11.3 Evoluția standardelor 802.11.

| Generație | Standard | Bandă           | Debit maxim | ma- | Modulație / acces       | An   |
|-----------|----------|-----------------|-------------|-----|-------------------------|------|
| Wi-Fi 0   | 802.11   | 2,4 GHz         | 2 Mbps      |     | FHSS / DSSS             | 1997 |
| Wi-Fi 1   | 802.11b  | 2,4 GHz         | 11 Mbps     |     | DSSS                    | 1999 |
| Wi-Fi 2   | 802.11a  | 5 GHz           | 54 Mbps     |     | OFDM                    | 1999 |
| Wi-Fi 3   | 802.11g  | 2,4 GHz         | 54 Mbps     |     | OFDM / DSSS             | 2003 |
| Wi-Fi 4   | 802.11n  | 2,4 / 5 GHz     | 600 Mbps    |     | OFDM + MIMO             | 2009 |
| Wi-Fi 5   | 802.11ac | 5 GHz           | 6,93 Gbps   |     | OFDM, până la 8 fluxuri | 2014 |
| Wi-Fi 6   | 802.11ax | 2,4 / 5 GHz     | 9,6 Gbps    |     | OFDMA                   | 2019 |
| Wi-Fi 7   | 802.11be | 2,4 / 5 / 6 GHz | ~30 Gbps    |     | OFDMA, 4096-QAM         | 2024 |

fluxuri înseamnă, în condiții bune, de patru ori debitul - folosind exact aceeași bandă.

3. **Canale mai late.** De la 20 MHz la 40 MHz (802.11n), 160 MHz (802.11ac) și 320 MHz (802.11be). Banda dublă înseamnă debit dublu - dacă găsiți atâta spectru curat.

4. **Metode mai bune de acces multiplu.** Trecerea de la DSSS la OFDM și apoi la OFDMA.

**De ce nu obțineți niciodată debitul de pe cutie.** Cifrele din tabel sunt maxime teoretice, obținute cu numărul maxim de antene, la distanță zero, fără alt trafic. În practică se pierd:

- jumătate din debit pe **antetele și confirmările** protocolului (mediul e semiduplex, fiecare cadru cere ACK);
- încă o parte din **partajarea** cu celelalte stații din același BSS;
- încă o parte din **scăderea modulației** odată cu distanța.

Un „Wi-Fi 6 de 1200 Mbps” livrează, realist, 400–600 Mbps la câțiva metri și mult mai puțin în camera alăturată.

### 11.5 Spreading și acces multiplu

**De ce împrăștiem semnalul pe o bandă mai largă decât e nevoie.** Pare o risipă: dacă informația încapă în 1 MHz, de ce să ocupăm 20? Pentru că într-o bandă liberă zgomotul e **punctual și imprevizibil**: cuptorul cu microunde strică o felie, telefonul Bluetooth alta. Un semnal concentrat pe 1 MHz care nimerește peste zgomot este pierdut complet. Un semnal **împrăștiat** pe 20 MHz pierde doar felia afectată, iar restul e suficient ca receptorul să reconstituie informația. În plus, energia fiind distribuită, semnalul e mai greu de detectat de cineva care nu știe ce caută - motiv pentru care tehnica vine, ca multe altele, din aplicații militare.

#### Analogie pentru OFDM și OFDMA

| TEHNOLOGIE        | CE ADUCE                                                 |
|-------------------|----------------------------------------------------------|
| MIMO              | mai mulți utilizatori servite simultan, planificat de AP |
| 256-QAM           | debit dublu, dacă există spectru liber                   |
| canale de 160 MHz | mai mulți biți per simbol, dacă semnalul e curat         |
| OFDMA             | fluxuri spațiale simultane, pe aceeași bandă             |
| beamforming       | concentrarea energiei către un client anume              |

Apasă un element din stânga, apoi perechea lui din dreapta.

Figura 11.2 Pârghia și efectul ei.

Tabela 11.4 Spreading și acces multiplu.

| Tehnică                          | Cum funcționează                                                                        | Observație                                                                              |
|----------------------------------|-----------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| <b>FHSS</b><br>Frequency Hopping | emițătorul „sare” între subbenzi după o secvență cunoscută ambelor capete               | simplic, dar folosește o singură subbandă la un moment dat; Bluetooth încă îl folosește |
| <b>DSSS</b><br>Direct Sequence   | fiecare bit este înlocuit cu o secvență fixă de <i>chipuri</i> , care ocupă tot canalul | semnalul se reface chiar dacă o parte din secvență e perturbată                         |
| <b>OFDM</b>                      | banda se împarte în multe subpurtătoare <b>ortogonale</b> , transmise simultan          | foarte rezistent la reflexii multiple; baza tuturor standardelor moderne                |
| <b>OFDMA</b>                     | subpurtătoarele sunt alocate <b>utilizatorilor diferiți</b> , simultan                  | Wi-Fi 6; transformă un mediu cu contenție într-unul planificat                          |

## OFDM

este un camion cu 64 de compartimente care pleacă spre o singură adresă. Chiar dacă trimiteți un plic mic, camionul pleacă tot. Dacă vreți să livrați la altcineva, așteptați următorul camion.

## OFDMA

încarcă în  
**același**

camion pachete pentru mai mulți destinatari și le lasă pe rând, pe același drum. Nimeni nu mai așteaptă un camion întreg pentru un plic.

De aceea Wi-Fi 6 nu e mult mai rapid pentru un singur client, dar e dramatic mai bun într-o sală cu o sută de laptopuri - exact scenariul unei săli de curs.

**Trei tipuri de „simultan”.** MIMO = mai multe fluxuri, pe aceeași frecvență, separate **spațial** (antene diferite).

**OFDM** = mai multe subpurtătoare, pentru **un singur** utilizator la un moment dat.

**OFDMA** = aceleași subpurtătoare, împărțite între **mai mulți** utilizatori.

Se combină: un AP Wi-Fi 6 face toate trei în același timp.

### 11.6 Planificarea canalelor

Banda de 2,4 GHz are aproximativ 85 MHz utilizabili (2400 – 2485 MHz), iar un canal 802.11g ocupă circa **22 MHz**. Canalele sunt însă definite din 5 în 5 MHz - deci se **suprapun** masiv. Rezultă loc pentru exact trei rețele care nu se suprapun deloc: canalele **1, 6 și 11**.

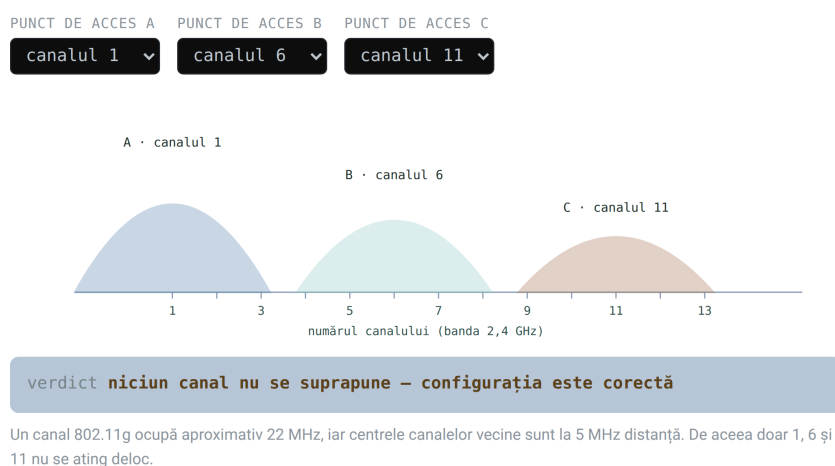


Figura 11.3 Suprapunerea canalelor în banda de 2,4 GHz.

Puneți toate trei punctele de acces pe canalele 1, 6 și 11 - nicio suprapunere. Apoi mutați unul pe canalul 3 și priviți ce se întâmplă: se suprapune parțial și cu 1, și cu 6. A stricat două rețele în loc să evite una.

#### Observație

**Greșeala „canalului 3”.** Un administrator care alege canalul 3 „pentru că e liber” provoacă interferență parțială atât pe 1, cât și pe 6.

Iar suprapunerea **parțială** este mai gravă decât folosirea **aceluiași** canal. Două rețele pe canalul 1 se aud reciproc și își împart politicos mediul prin CSMA/CA: fiecare așteaptă când cealaltă vorbește. O rețea pe canalul 3 nu este auzită ca semnal valid, deci nimeni nu îi cedează mediul - pentru celelalte, este pur și simplu **zgomot**.

**La 5 GHz problema aproape dispare.** Sunt disponibile peste 20 de canale de 20 MHz care nu se suprapun, deci planificarea devine banală. Prețul: acoperire mai mică, deci mai multe puncte de acces pentru aceeași clădire.

Atenție însă la canalele **DFS** (52–140): acolo Wi-Fi-ul este oaspete, iar radarele meteo sunt proprietarul. Dacă echipamentul detectează un radar, este obligat să părăsească imediat canalul - ceea ce înseamnă câteva secunde de întrerupere pentru toți clienții.

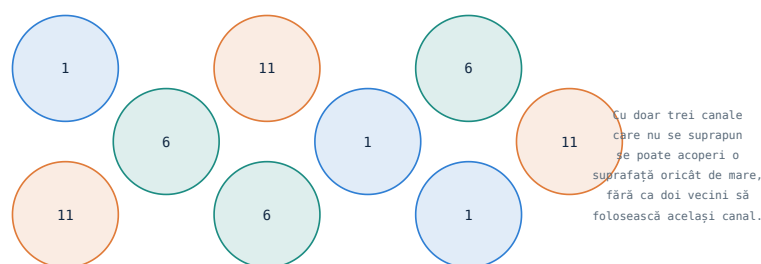
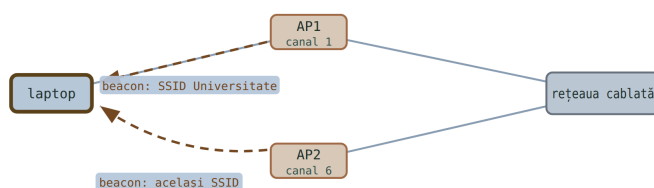


Figura 11.4 Reutilizarea canalelor într-o rețea cu mai multe puncte de acces. Este exact aceeași idee ca la celulele de telefonie mobilă, aplicată la scara unei clădiri.

Tabela 11.5 Organizarea unei rețele wireless.

| Termen                            | Ce este                                                                                                       |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------|
| <b>BSS</b> - Basic Service Set    | unitatea de bază: una sau mai multe stații plus cel mult un punct de acces                                    |
| <b>IBSS</b> - Independent BSS     | BSS fără punct de acces: rețea ad-hoc, stațiile vorbesc direct între ele                                      |
| <b>ESS</b> - Extended Service Set | mai multe BSS-uri legate printr-un sistem de distribuție (de regulă rețeaua cablată)                          |
| <b>BSSID</b>                      | identificator de 48 de biți al unui BSS; într-o rețea cu AP este chiar <b>adresa MAC</b> a punctului de acces |
| <b>SSID</b>                       | numele rețelei, configurabil de administrator; ceea ce vede utilizatorul în listă                             |



pasul 1 din 6

#### Scanarea

Laptopul ascultă pe fiecare canal cadrele **beacon**, emise de fiecare AP la circa 100 ms. Un beacon conține SSID-ul, BSSID-ul, ratele suportate și cerințele de securitate.

Alternativ, laptopul poate întreba activ, cu un cadru **probe request** - și atunci își strigă numele rețelelor căutate, ceea ce e relevant pentru secțiunea de securitate.

Figura 11.5 De la scanare la roaming.

## 11.7 Organizarea unei rețele wireless

Echipamentele tipice sunt: adaptoare wireless, puncte de acces, bridge-uri wireless (leagă două segmente cablate prin aer), repetoare de semnal, antene direcționale sau omnidirecționale și, în instalațiile mari, **controlere WLAN** care administrează central zeci de puncte de acces *lightweight* - acestea din urmă neavând practic configurație proprie.

## 11.8 Accesul la mediu

Mediul wireless este partajat, exact ca vechiul cablu coaxial. Dar CSMA/CD nu poate fi folosit, din două motive fizice imposibil de ocolit:

**De ce nu se poate detecta o coliziune în aer. Motivul 1.** O stație nu poate asculta în timp ce transmite. Propriul semnal, la propria antenă, este cu **zeci de ordine de mărime** mai puternic decât orice ar putea recepționa de la distanță. E ca și cum ai încerca să auzi o șoaptă din cealaltă cameră în timp ce strigi. **Motivul 2.** Chiar dacă ar putea, coliziunea se produce **la receptor**, nu la emițător. Două stații aflate la 100 m una de alta pot transmite fără să se deranjeze reciproc - dar semnalele lor se pot ciocni perfect la punctul de acces aflat între ele. Pe cablu, ambele probleme dispar: semnalul e același pe tot firul și o stație își poate compara ce transmite cu ce se aude pe mediu.

Soluția este **CSMA/CA** - *Collision Avoidance* în loc de *Detection*. Dacă nu putem detecta coliziunile, le prevenim.

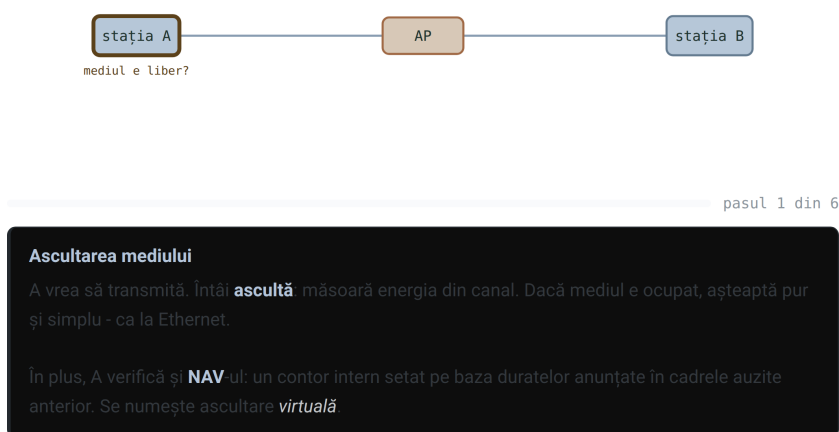


Figura 11.6 CSMA/CA, pas cu pas.

**PCF a rămas, practic, pe hârtie.** Aproape niciun echipament comercial nu l-a implementat: era complicat și nu aducea destul. Ideea de acces **planificat de AP** a revenit însă, în altă formă și cu mult mai mult succes, prin **OFDMA** în Wi-Fi 6 - unde punctul de acces chiar programează cine transmite și pe ce subpurtătoare.

Tabela 11.6 Accesul la mediu.

| Funcție                                        | Când se folosește                      | Cum decide cine transmite                                                                  |
|------------------------------------------------|----------------------------------------|--------------------------------------------------------------------------------------------|
| <b>DCF</b> - Distributed Coordination Function | întotdeauna, inclusiv în rețele ad-hoc | distribuit, prin CSMA-/CA: fiecare stație ascultă, așteaptă DIFS și un backoff aleator     |
| <b>PCF</b> - Point Coordination Function       | opțional, doar cu punct de acces       | centralizat: AP-ul interoghează pe rând stațiile (polling), într-o perioadă fără conținție |

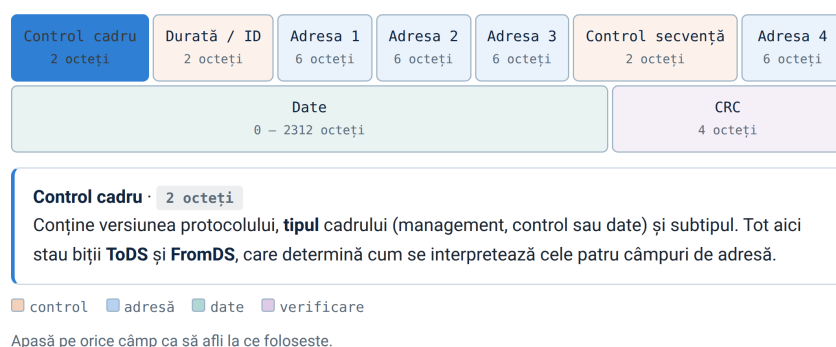


Figura 11.7 Cadrul 802.11 - apăsați pe câmpuri.

## 11.9 Cadrul 802.11

**De ce patru adrese și nu două.** Ethernet are nevoie de două: sursa și destinația, ambele pe același segment.

În aer, cadrul are **patru roluri** potențial diferite: cine îl emite acum, cine îl primește acum, de la cine provine originalul și pentru cine este destinat în final. Într-un BSS obișnuit trei sunt suficiente, pentru că AP-ul este mereu unul dintre capete. A patra apare doar când cadrul sare între două puncte de acces.

## 11.10 Două probleme care nu există pe cablu

### 11.10.1 Stația ascunsă

### 11.10.2 Stația expusă

Problema simetrică, și mai puțin cunoscută: o stație aude o transmisie și tace, **deși transmisia ei nu ar deranja pe nimeni** - pentru că destinatarii sunt în direcții opuse.

Rezultatul nu este o coliziune, ci **pierdere de capacitate**: mediul stă degeaba deși ar fi putut fi folosit. Soluțiile sunt folosirea de canale diferite, antene direcționale, sau coordonarea centralizată - adică exact ce face OFDMA în Wi-Fi 6.

**Cele două probleme, în oglindă. Stația ascunsă:** nu aud pe cineva pe care ar trebui să îl aud → transmit când nu trebuie → **coliziune**.

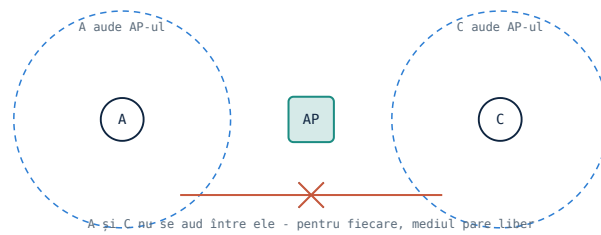
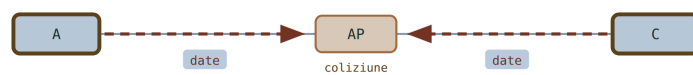


Figura 11.8 Problema stației ascunse. A și C sunt amândouă în raza AP-ului, dar nu în raza celeilalte. Ascultă mediul, îl găsesc liber și transmit simultan; coliziunea se produce la AP, unde niciuna dintre ele nu o poate observa.



pasul 1 din 6

#### Problema

A și C nu se aud reciproc. Amândouă ascultă mediul, îl găsesc liber și transmit.

Coliziunea se produce **la AP**. Niciuna dintre ele nu o poate observa - pentru fiecare, totul a mers perfect. Vor afla abia când nu vine ACK-ul.

Figura 11.9 RTS/CTS rezolvă stația ascunsă.

**Stația expusă:** aud pe cineva pe care nu ar trebui să îl iau în seamă → tac când aş putea vorbi → **capacitate irosită**.

Prima produce erori, a doua produce încetineală. Prima are o soluție standard (RTS/CTS), a doua nu.

## 11.11 Securitatea rețelilor wireless

Pe cablu, un atacator trebuie să intre în clădire. În aer, îi ajunge să fie în parcare. Diferența aceasta explică de ce securitatea wireless a fost, de la început, o problemă mai grea - și de ce primele soluții au eșuat spectaculos.

### 11.11.1 Mecanismele care nu sunt securitate

#### Observație

**Ascunderea SSID-ului face rău, nu bine.** Un dispozitiv configurat pentru o rețea ascunsă nu o poate găsi pasiv, deci trebuie să **întrebe activ**: emite continuu cadre *probe request* cu numele rețelei căutate.

Tabela 11.7 Mecanismele care nu sunt securitate.

| Măsură                      | Ce pretinde                        | De ce nu ajunge                                                                                        |
|-----------------------------|------------------------------------|--------------------------------------------------------------------------------------------------------|
| Reducerea puterii de emisie | semnalul nu iese din clădire       | o antenă direcțională îl captează de la sute de metri; iar recepția e mult mai sensibilă decât emisia  |
| Ascunderea SSID-ului        | rețeaua devine invizibilă          | SSID-ul apare <b>în clar</b> în cadrele de asociere ale clienților legitimi - e destul să aștepti unul |
| Filtrarea după adrese MAC   | doar dispozitivele cunoscute intră | adresele MAC circulă <b>neprotejate</b> în fiecare cadru și pot fi copiate în câteva secunde           |

Rezultatul: telefonul dumneavoastră strigă numele rețelei „ascunse” în fiecare cafenea, aeroport și stație de metrou prin care treceți. Ați ascuns rețeaua de la locul ei și ați anunțat-o peste tot altundeva.

Toate trei sunt utile ca măsuri de **igienă** - reduc zgomotul și tentativele oportuniste - dar niciuna nu oprește un atacator care știe ce face. Securitatea reală începe la criptare.

Tabela 11.8 Mecanismele care nu sunt securitate.

| Mecanism           | Criptare                                 | Autentificare            | Statut                                                                   |
|--------------------|------------------------------------------|--------------------------|--------------------------------------------------------------------------|
| Open               | niciuna                                  | niciuna                  | doar pentru rețele publice, cu portal spart complet; se sparge în minute |
| <b>WEP</b> (1999)  | RC4, cheie statică                       | cheie partajată          | soluție de tranziție, depășită                                           |
| <b>WPA</b> (2003)  | TKIP                                     | PSK sau RADIUS           | încă acceptabil                                                          |
| <b>WPA2</b> (2004) | AES-CCMP                                 | PSK sau 802.1X           | recomandat                                                               |
| <b>WPA3</b> (2018) | AES, cadre de management protejate (PMF) | <b>SAE</b> în loc de PSK |                                                                          |

**De ce a căzut WEP: vectorul de inițializare.** WEP folosea RC4 cu o cheie fixă, la care adăuga un **vector de inițializare** (IV) de doar **24 de biți**, transmis în clar, pentru ca fiecare cadru să fie criptat diferit. 24 de biți înseamnă 16,7 milioane de valori. Într-o rețea încărcată, ele se epuizează în câteva ore - iar de la primul IV repetat se pot deduce informații despre cheie. Mai rău: atacatorul nu trebuie să aștepte. Poate **injecta** trafic (retransmițând cereri ARP captate) ca să forțeze rețeaua să genereze IV-uri rapid. Din 2007, o rețea WEP se sparge în sub cinci minute, cu unelte publice. Lecția generală: nu criptarea a fost slabă, ci **modul în care a fost folosită**.

Două îmbunătățiri ale WPA3 merită înțelese, pentru că rezolvă slăbiciuni concrete ale



lui WPA2:

- **SAE** (Simultaneous Authentication of Equals) înlocuiește PSK-ul clasic. Cu WPA2-PSK, un atacator care captează strângerea de mână de patru pași poate încerca **offline**, la nesfârșit și fără să atingă rețeaua, dicționare întregi. Cu SAE, fiecare încercare de parolă cere o **interacțiune** cu rețeaua - deci atacul devine lent, detectabil și limitabil.
- **Perfect Forward Secrecy**. Cheia de sesiune se derivă din nou la fiecare conexiune, deci compromiterea parolei **nu** permite decriptarea traficului înregistrat anterior. Cu WPA2, cine avea parola și o captură veche putea citi tot.

#### Observație

**Personal și Enterprise.** În modul **personal**, toată lumea folosește aceeași parolă - dacă pleacă un angajat, trebuie schimbată pentru toți, pe toate dispozitivele. În modul **enterprise**, fiecare utilizator se autentifică individual, prin **802.1X** și un server **RADIUS**, cu credențiale proprii sau certificat.

Pentru orice rețea de organizație - inclusiv cea a unei facultăți - enterprise este singura variantă gestionabilă: se revocă un singur cont, nu se schimbă o parolă pentru trei mii de oameni.

**Rogue AP și evil twin.** Cea mai eficientă cale de atac nu este spargerea criptării, ci **păcălirea clientului**: se ridică un punct de acces cu același SSID și semnal mai puternic, iar dispozitivele se conectează singure.

Apărarea nu stă în parolă, ci în **autentificarea reciprocă** - clientul trebuie să verifice certificatul serverului RADIUS, exact ca un browser care verifică certificatul unui site HTTPS. Un dispozitiv configurat să nu valideze certificatul este vulnerabil oricât de puternică ar fi criptarea.

### 11.12 Greșeli frecvente

#### Observație

**Greșeli frecvente.**

- „**Am pus toate AP-urile pe canale libere, 1, 3, 5**” Canalele 3 și 5 se suprapun parțial cu 1 și 6, deci interferează cu tot. Doar **1, 6 și 11** în banda de 2,4 GHz. Fără excepții.
- „**Am ascuns SSID-ul, deci rețeaua e mai sigură**” Nu e. Iar clienții își strigă numele rețelei peste tot în lume. SSID vizibil, WPA3 sau WPA2 cu parolă lungă. Securitatea vine din criptare.
- „**Am pus canale de 40 MHz pe 2,4 GHz, ca să am debit dublu**” Un canal de 40 MHz ocupă aproape jumătate din bandă și strică ambii vecini. 20 MHz pe 2,4 GHz. Canalele late au sens doar pe 5 și 6 GHz.
- „**Debitul real e mult sub cel de pe cutie**” Normal: jumătate din timpul mediului se duce pe protocol, ACK-uri și partajare. Așteptați-vă la 40–50% din cifra teoretică, în cel mai bun caz.

- „**Am adăugat un repetor ca să acopăr toată casa**” Un repetor retransmite ce aude pe **același canal**, deci înjumătățește debitul disponibil și ocupă mediul de două ori pentru fiecare cadru. Un al doilea AP legat prin cablu, pe alt canal. Sau mesh cu radio dedicat.
- „**Filtrarea MAC ține intrușii afară**” Adresele circulă în clar în fiecare cadru și se pot clona în câteva secunde. Utilă ca inventar, inutilă ca securitate.
- „**Am configurat WPA2-Enterprise, dar clienții nu verifică certificatul**” Un evil twin poate captura credențialele oricui. Configurați explicit validarea certificatului serverului RADIUS pe toate dispozitivele.

### 11.13 Rezumat și glosar

**Ce trebuie să rămână.** Frecvență mai mare = debit mai mare, dar acoperire și penetrare mai mici. Nu există bandă „mai bună”. Debitul a crescut prin **modulație densă, MIMO, canale late și OFDMA**. În banda de 2,4 GHz există exact **trei** canale care nu se suprapun: 1, 6, 11. **SSID** = numele rețelei; **BSSID** = adresa MAC a unui AP. Roaming-ul păstrează primul și schimbă al doilea. CSMA/CD e imposibil în aer: nu poți asculta cât transmiți, iar coliziunea se produce la **receptor**. **CSMA/CA**: ascultă, DIFS, backoff aleator, transmite, așteaptă **ACK**. Câmpul **Durată** și **NAV**-ul sunt ascultarea „virtuală” a mediului. **RTS/CTS** rezolvă stația ascunsă, pentru că CTS-ul AP-ului e auzit de toți. Puterea redusă, SSID-ul ascuns și filtrarea MAC **nu sunt securitate**. **WPA3 cu SAE** este recomandarea; **Enterprise** este singura variantă gestionabilă într-o organizație.

### 11.14 Ce urmează

Ultimul curs adună la un loc atacurile care funcționează **în interiorul** rețelei locale - inclusiv cele care exploatează chiar mecanismele studiate în cursurile 3 și 4, de la tabela CAM la protocolul STP - și configurațiile care le opresc.

**Laboratorul 7** include construirea și securizarea unei rețele wireless în Packet Tracer, cu planificarea canalelor și WPA2-Enterprise.

- IEEE 802.11 - standardul complet, cu amendamentele a/b/g/n/ac/ax/be
- IEEE 802.11i - mecanismele de securitate care au devenit WPA2
- Wi-Fi Alliance - specificațiile WPA3 și cerințele de certificare
- ANCOM - reglementările pentru benzile ISM în România
- Matthew Gast, *802.11 Wireless Networks: The Definitive Guide*

### Întrebări de verificare

1. De ce Wi-Fi folosește CSMA/CA în loc de CSMA/CD?

- a) Pentru ca este mai rapid
- b) Pentru ca o stație nu poate asculta în timp ce transmite, iar coliziunea se produce la receptor, nu la emitor

- c) Pentru ca standardul IEEE interzice CD
  - d) Pentru ca nu exista coliziuni in aer
2. La ce serveste campul Durata din cadrul 802.11?
- a) Masoara cat a durat transmisia anterioara
  - b) Anunta cat timp va fi ocupat mediul, iar celelalte statii isi seteaza NAV-ul pe aceasta valoare
  - c) Indica vechimea cadrului
  - d) Stabileste intervalul dintre doua beacon-uri
3. Ce rezolva mecanismul RTS/CTS?
- a) Problema statiei expuse
  - b) Problema statiei ascunse: CTS-ul AP-ului este auzit de toate statiile din raza lui, inclusiv de cele care nu aud emitatorul
  - c) Cresterea debitului
  - d) Criptarea traficului
4. De ce backoff-ul din CSMA/CA este aleator?
- a) Ca sa economiseasca energie
  - b) Ca doua statii care au asteptat impreuna sfarsitul unei transmisii sa nu porneasca simultan
  - c) Pentru ca nu se poate masura timpul exact
  - d) Ca sa cripteze transmisia
5. Un administrator pune trei puncte de acces pe canalele 1, 3 si 6. Care este problema?
- a) Nicio problema, sunt canale diferite
  - b) Canalul 3 se suprapune partial atat cu 1 cat si cu 6, deci interfereaza cu ambele
  - c) Trebuiau puse toate pe acelasi canal
  - d) Canalul 3 nu exista
6. De ce ascunderea SSID-ului nu este o masura de securitate?
- a) Pentru ca reduce debitul
  - b) Pentru ca SSID-ul apare in clar in cadrele de asociere ale clientilor legitimi
  - c) Pentru ca nu functioneaza pe 5 GHz
  - d) Pentru ca il pot vedea doar administratorii
7. Doua puncte de acces din aceeași rețea, într-un ESS, au:

- a) Acelasi SSID si acelasi BSSID
  - b) Acelasi SSID, dar BSSID-uri diferite
  - c) SSID-uri diferite si acelasi BSSID
  - d) SSID-uri si BSSID-uri diferite
8. De ce a fost spart WEP?
- a) Pentru ca AES era prea slab
  - b) Pentru ca vectorul de initializare avea doar 24 de biti si se repeta rapid, iar atacatorul putea forta generarea lui
  - c) Pentru ca nu folosea deloc criptare
  - d) Pentru ca parolele erau prea scurte
9. Ce aduce nou SAE, in WPA3, fata de PSK-ul din WPA2?
- a) Crijteaza mai rapid
  - b) Impiedica atacul de dictionar offline pe strangerea de mana captata
  - c) Elimina nevoia de parola
  - d) Creste raza de acoperire



## 12. Securitatea LAN și atacuri de rețea

---

---

- Din cursurile de comutare
- Trei familii de atacuri
- Recunoașterea
- Atacuri de blocare a serviciului
- CAM overflow
- Man-in-the-middle prin ARP poisoning
- VLAN hopping
- Atacuri împotriva STP
- Parole, hash-uri și salt
- Factorul uman și codul ostil
- Lista scurtă de securizare a unui switch
- Greșeli frecvente
- Rezumat și glosar
- Încheiere
- Întrebări de verificare

---

---

Firewall-ul de la marginea rețelei, oricât de bine configurat, nu vede absolut nimic din ce se întâmplă între două calculatoare din aceeași încăpere. Traficul lor nu îl traversează niciodată. Cursul acesta se ocupă exact de acest spațiu orb: atacurile care funcționează în interiorul rețelei locale, aproape toate exploatând mecanisme pe care le-am construit noi înșine, în cursurile 3 și 4, și care fac exact ce au fost proiectate să facă. Nu vom repara protocoalele - nu se poate. Vom adăuga în jurul lor verificările care le lipsesc.

### 12.1 Din cursurile de comutare

**Ce trebuie să aveți în minte.** Un switch **învață** adresele MAC din câmpul **sursă** al cadrelor primite. Tabela **CAM** are dimensiune finită; ce nu se găsește în ea se **inundă** pe toate porturile. **ARP** asociază o adresă IP cu o adresă MAC, întrebând prin broadcast. **VLAN-ul nativ** circulă **neetichetat** pe un trunk. **STP** alege ca rădăcină switch-ul cu cel mai mic Bridge ID. **DHCP** funcționează prin broadcast; clientul acceptă prima ofertă sosită.

Fiecare dintre aceste șase propoziții este, astăzi, punctul de plecare al unui atac. Nu

pentru că ar fi implementate greșit, ci pentru că sunt implementate **exact așa cum au fost gândite** - într-o epocă în care rețeaua locală era, implicit, un spațiu de încredere.

### 12.1.1 Ce veți putea face după acest curs

**Ce veți putea face după acest capitol.**

- Să clasificați un atac după scop și să spuneți ce mecanism exploatează
- Să explicați mecanica atacurilor CAM overflow, ARP poisoning și VLAN hopping
- Să descrieți cum poate un singur cablu opri o rețea întreagă prin STP
- Să configurați port security, DHCP snooping și Dynamic ARP Inspection
- Să închideți VLAN hopping-ul prin trei setări
- Să explicați ce face și, mai ales, ce **nu** face un salt
- Să aplicați o listă completă de securizare pe un switch de acces

## 12.2 Trei familii de atacuri

**De ce firewall-ul nu ajută aici.** Un firewall stă **între** două rețele și vede doar traficul care îl traversează. Dacă stația A și stația B sunt în același VLAN, pe același switch, traficul dintre ele nu ajunge niciodată la un ruter, deci nici la firewall. Switch-ul îl comută direct, la nivelul 2. Toate atacurile din acest curs se poartă **sub** nivelul la care privește un firewall clasic. Apărarea trebuie să fie tot acolo - pe switch.

Tabela 12.1 Trei familii de atacuri.

| Familie              | Scop                              | Exemple în LAN                                                     |
|----------------------|-----------------------------------|--------------------------------------------------------------------|
| <b>Recunoaștere</b>  | afierea a ce există și ce rulează | ping sweep, port scan, sniffing, interogări DNS                    |
| <b>Blocare (DoS)</b> | indisponibilizarea unui serviciu  | SYN flood, smurf, CAM overflow, atac STP                           |
| <b>Acces</b>         | obținerea de drepturi nemeritate  | ARP poisoning, VLAN hopping, spargere de parole, inginerie socială |

**Observația care organizează tot cursul.** Aproape fiecare atac de aici exploatează un mecanism **care funcționează exact așa cum a fost proiectat**.

ARP nu autentifică nimic - prin proiect. Switch-ul crede orice MAC sursă - prin proiect. STP acceptă orice BPDU - prin proiect. DHCP-ul acceptă prima ofertă - prin proiect.

Toate au fost gândite într-o epocă în care a fi conectat fizic la rețea însemna a fi de încredere. Contramăsurile nu repară protocoalele; adaugă în jurul lor verificările care lipsesc.

### Analogie

E ca într-o clădire de birouri unde toate ușile interioare sunt deschise, pentru că oricine a trecut de recepție este considerat angajat.

Firewall-ul este recepția. Măsurile din acest curs sunt încuietorile de pe ușile interioare - și motivul pentru care se numesc, împreună,

**securitate în adâncime**

### 12.3 Recunoașterea

Înainte oricărui atac serios vine culegerea de informații: ce adrese IP sunt active, ce servicii rulează pe ele, în ce versiune, ce sisteme de operare și în cine au încredere utilizatorii rețelei.

Aceleași unelte sunt folosite zilnic și de administratori - diferența este **autorizarea**, nu instrumentul.

Tabela 12.2 Recunoașterea.

| Utilitar  | Ce face                                                                          | Exemplu                                        |
|-----------|----------------------------------------------------------------------------------|------------------------------------------------|
| nmap      | descoperă stații active, porturi deschise, servicii, versiuni, sistem de operare | <code>nmap -sV 192.168.1.0/24</code>           |
| tcpdump   | captură de trafic din linia de comandă, cu filtre                                | <code>tcpdump -i eth0 -c 10 dst port 80</code> |
| Wireshark | captură și analiză grafică; cere interfața în mod promiscuu                      | format libpcap, comun cu tcpdump               |
| whois     | informații administrative despre un domeniu                                      | <code>whois example.com</code>                 |
| host, dig | interogări DNS: servere de nume, de mail                                         | <code>host -t MX example.com</code>            |

#### Observație

**Regula de aur a acestui curs.** Toate uneltele de mai sus se folosesc **exclusiv** pe rețeaua de laborator, pe propriile dumneavoastră dispozitive.

Scanarea sau interceptarea traficului altcuiva - chiar și în aceeași rețea, chiar și „doar ca să văd dacă merge” - este ilegală și se pedepsește. Scopul acestui curs este să înțelegeți ce vede un atacator ca să puteți construi apărarea, nu să exersați pe sisteme străine.

```
user@lab:~$ ?
Comenzi disponibile aici:
nmap -sn 192.168.1.0/24
nmap -sV 192.168.1.10
arp -a
tcpdump -i eth0 -n arp
host -t MX example.com
ip neigh flush all
```

user@lab:~\$

Înșiriri tipice ale uneltelor de recunoaștere. Rulate, în acest curs, doar pe rețeaua de laborator.

Figura 12.1 Terminal: ce vede o recunoaștere.

Priviți a treia comandă. Două adrese IP diferite cu **aceeași** adresă MAC este anomalia



care trădează un atac de tip man-in-the-middle - și este verificarea cea mai ieftină pe care o puteți face pe o stație suspectă.

Ce poate face un firewall împotriva recunoașterii: să blocheze răspunsul la ICMP echo, să închidă porturile vulnerabile, să interzică inițierea conexiunilor din exterior, să limiteze rata cererilor. Ce **nu** poate face: să oprească un atacator aflat deja în rețeaua locală. De aici, restul cursului.

## 12.4 Atacuri de blocare a serviciului

### 12.4.1 Smurf attack

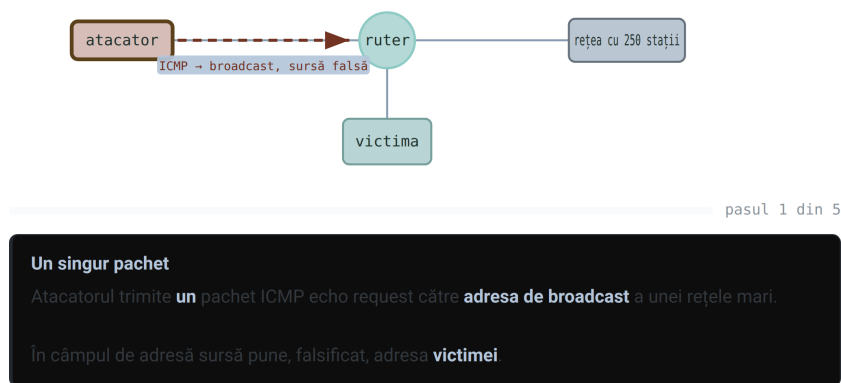


Figura 12.2 Smurf attack: amplificarea.

### 12.4.2 TCP SYN flood

L-am întâlnit în cursul 8: atacatorul trimite mii de segmente SYN cu adrese sursă falsificate, iar serverul alocă câte o intrare în coada de conexiuni **pe jumătate deschise** pentru fiecare. Când coada se umple, conexiunile legitime sunt refuzate.

Apărări: **SYN cookies**, care permit serverului să nu aloce nimic până la confirmare; limitarea numărului de conexiuni half-open per sursă; și, ca întotdeauna, monitorizarea traficului în condiții normale - o anomalie se observă doar dacă știți cum arată normalul.

## 12.5 CAM overflow

Acesta este atacul care exploatează direct mecanismul din cursul 3. Memoria unui switch este finită - tipic între 8 000 și 32 000 de intrări - deci tabela CAM se poate umple. Iar când se umple, switch-ul nu mai poate învăța adrese noi și **inundează** toate cadrele necunoscute pe toate porturile. Cu alte cuvinte: redevine un hub.

Reluați simulatorul de tabelă CAM din cursul 3 și urmăriți ce se întâmplă la fiecare cadru: switch-ul învață **doar** din adresa sursă și inundă **orice** destinație necunoscută. Atacul nu face decât să împingă aceste două reguli la limită.

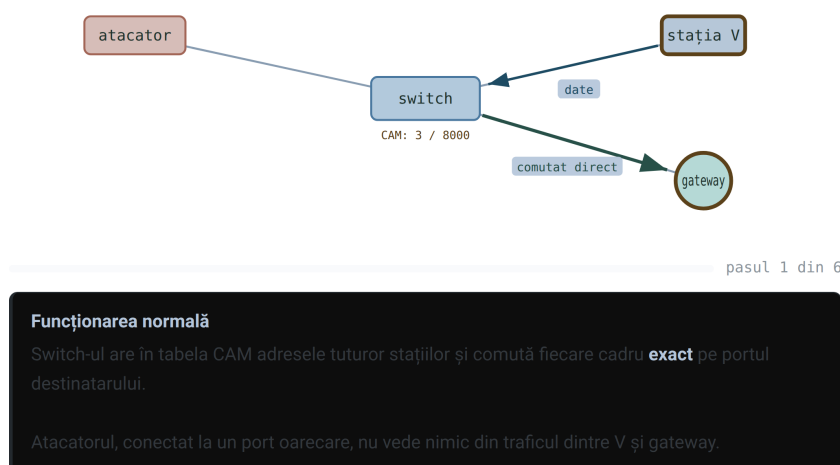


Figura 12.3 CAM overflow, pas cu pas.

SURSĂ DESTINAȚIE

A C

Tabela CAM a switch-ului

| ADRESĂ MAC            | PORT |
|-----------------------|------|
| tabela CAM este goală |      |

Ce face switch-ul

niciun cadru trimis încă

Figura 12.4 Simulator de tabelă CAM - învățare și inundare.

## Exemplu 12.1 contramăsura: port security copiază

```

1
2 SW(config)# interface fastEthernet 0/5
3 SW(config-if)# switchport mode access
4 SW(config-if)# switchport port-security
5 SW(config-if)# switchport port-security maximum 2
6 SW(config-if)# switchport port-security mac-address sticky
7 SW(config-if)# switchport port-security violation restrict
8 SW(config-if)# switchport port-security aging time 60
9
10 SW# show port-security interface fastEthernet 0/5
11 SW# show mac address-table count

```

**Ce înseamnă sticky.** Adresele învățate dinamic sunt scrise automat în configurația activă, ca și cum le-ați fi introdus manual. Avantajul: nu trebuie să cunoașteți dinainte adresa fiecărui calculator.

Atenție însă: dacă nu salvați configurația, ele se pierd la repornire - iar dacă le salvați, portul rămâne legat de acel calculator anume, ceea ce devine o problemă la prima înlocuire de echipament.

Tabela 12.3 CAM overflow.

| Acțiune la violare | Ce face                                                 | Când o folosiți                                                         |
|--------------------|---------------------------------------------------------|-------------------------------------------------------------------------|
| protect            | aruncă tăcut cadrele în exces                           | rar - nu aflați niciodată că s-a întâmplat ceva                         |
| restrict           | aruncă, incrementează un contor, trimite SNMP și syslog | <b>implicit recomandat</b>                                              |
| shutdown           | închide portul (err-disable)                            | zone sensibile; cere intervenție manuală sau <b>errdisable recovery</b> |

## 12.6 Man-in-the-middle prin ARP poisoning

ARP, așa cum l-am studiat în cursul 5, are o proprietate care astăzi pare de neconceput: **nu autentifică nimic**. Mai mult, majoritatea implementărilor acceptă un răspuns ARP chiar dacă nu au trimis nicio cerere - mecanism numit *gratuitous ARP*, util pentru anunțarea unei schimbări de adresă, dar exploatabil.

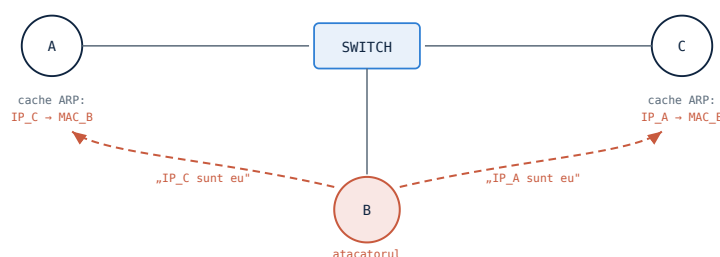


Figura 12.5 ARP poisoning. B trimite fiecăruia dintre cei doi un răspuns ARP nesolicitat. Ambele stații își actualizează cache-ul și încep să adreseze cadrele către MAC-ul lui B. Switch-ul le comută corect - către atacator.

### Exemplu 12.2 contramăsura: DHCP snooping + Dynamic ARP Inspection copiază

```

1
2 ! 1. switch-ul invata ce adresa IP a primit fiecare port, prin
   ↪ DHCP
3 SW(config)# ip dhcp snooping
4 SW(config)# ip dhcp snooping vlan 10,20
5 SW(config)# interface gigabitEthernet 0/1
6 SW(config-if)# ip dhcp snooping trust
7
8 ! 2. pe baza acestei evidente, se verifica fiecare mesaj ARP
9 SW(config)# ip arp inspection vlan 10,20
10 SW(config)# interface gigabitEthernet 0/1
11 SW(config-if)# ip arp inspection trust

```

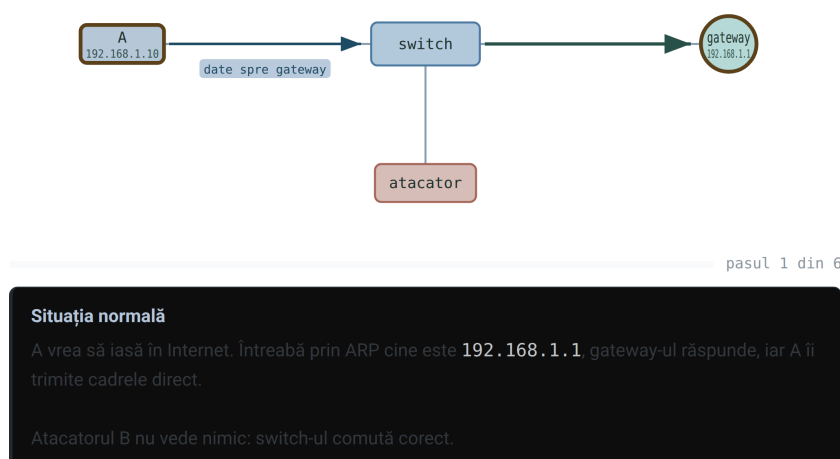


Figura 12.6 ARP poisoning și cum se încheie.

```

12
13 ! 3. pentru statiile cu adresa statica , evidenta se scrie manual
14 SW(config)# arp access-list SERVERE
15 SW(config-arp-nacl)# permit ip host 192.168.1.50 mac host 000c
    ↪ .2911.2233
16 SW(config)# ip arp inspection filter SERVERE vlan 10
17
18 SW# show ip dhcp snooping binding
19 SW# show ip arp inspection statistics

```

**Un beneficiu suplimentar: rogue DHCP.** DHCP snooping oprește, în același timp, un atac complet diferit: cineva își pornește propriul server DHCP și se anunță drept gateway al tuturor. Clienții acceptă prima ofertă sosită - iar un server local, în aceeași încăpere, răspunde mai repede decât cel legitim.

Porturile *nettrusted* nu au voie să trimită mesaje de server (Offer, Ack), deci atacul nici nu pornește. Aceeași comandă, două atacuri închise.

## 12.7 VLAN hopping

Am construit VLAN-uri tocmai ca să izolăm departamente. Există două tehnici prin care izolarea poate fi ocolită - ambele exploatând comportamente **implicite** ale switch-urilor.

### 12.7.1 Switch spoofing

Multe switch-uri Cisco au implicit activat **DTP** (Dynamic Trunking Protocol), care negociază automat dacă un port devine acces sau trunk. Un atacator poate să se prezinte drept switch și să negocieze un trunk. Odată obținut, poate trimite și primi cadre etichetate 802.1Q pentru **orice** VLAN.

## Exemplu 12.3 contramăsura: nicio negociere pe porturile de acces copiază

```

1
2 SW(config)# interface range fastEthernet 0/1 - 22
3 SW(config-if-range)# switchport mode access
4 SW(config-if-range)# switchport nonegotiate
5 SW(config-if-range)# switchport access vlan 10
6
7 ! porturile nefolosite: inchise si mutate intr-un VLAN mort
8 SW(config)# interface range fastEthernet 0/23 - 24
9 SW(config-if-range)# switchport access vlan 999
10 SW(config-if-range)# shutdown

```

### 12.7.2 Double tagging

Tehnica exploatează **VLAN-ul nativ**, despre care știm din cursul 4 că circulă **neetichetat** pe trunk. Nu necesită niciun protocol special pe atacator - doar posibilitatea de a construi un cadru cu două etichete.

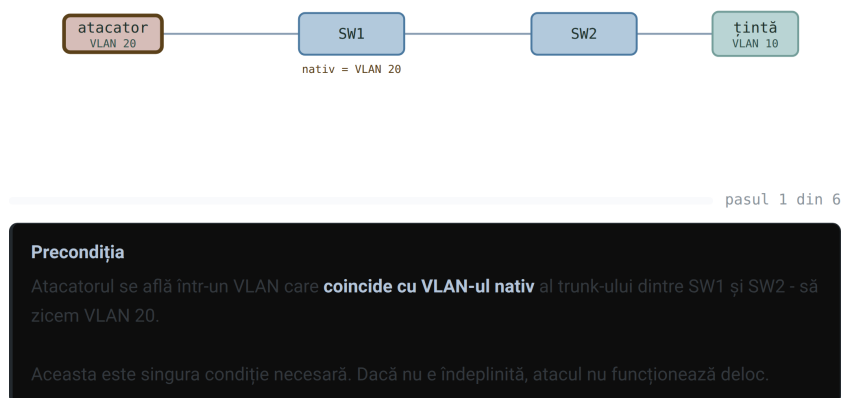


Figura 12.7 Double tagging, pas cu pas.

**Trei reguli care închid VLAN hopping-ul.** 1. VLAN-ul nativ al trunk-urilor să fie un VLAN nefolosit, în care nu se află nicio stație.

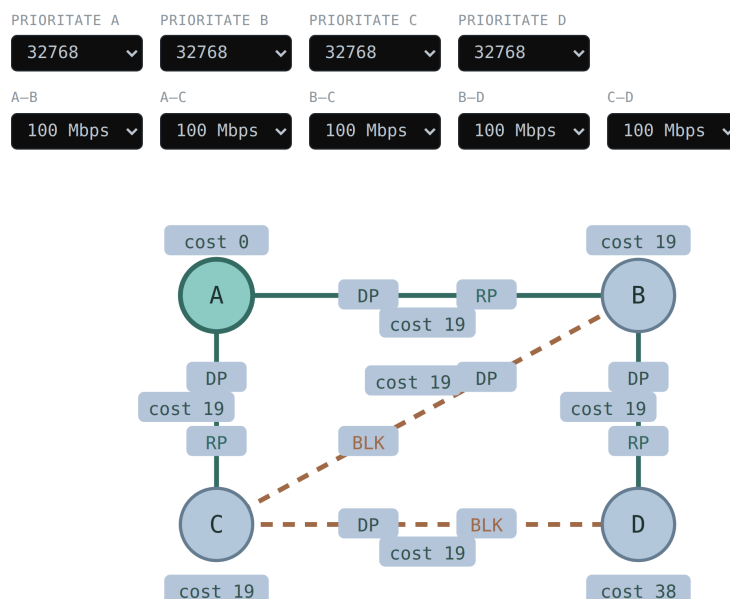
2. Porturile de acces să fie explicit `switchport mode access` cu `switchport nonegotiate`.

3. Pe trunk-uri, lista de VLAN-uri permise să fie declarată explicit (`switchport trunk allowed vlan 10,20,30`), nu lăsată implicit la „toate”.

### 12.8 Atacuri împotriva STP

STP nu folosește nicio formă de autentificare: **orice** dispozitiv care trimite BPDU-uri participă la alegerea rădăcinii. Un atacator conectat oriunde în rețea poate anunța BPDU-uri cu un Bridge ID foarte mic și deveni root bridge - moment în care o parte însemnată a

traficului începe să treacă prin el.



| SWITCH | BRIDGE ID              | ROL         | COST SPRE RĂDĂCINĂ | PRIN |
|--------|------------------------|-------------|--------------------|------|
| A      | 32768 . 000A.0000.0001 | root bridge | 0                  | —    |
| B      | 32768 . 000B.0000.0002 | non-root    | 19                 | A    |
| C      | 32768 . 000C.0000.0003 | non-root    | 19                 | A    |
| D      | 32768 . 000D.0000.0004 | non-root    | 38                 | B    |

Rădăcina este switch-ul cu cel mai mic Bridge ID (prioritate, apoi adresa MAC). Pe fiecare switch fără rol de rădăcină se alege un singur port rădăcină, iar pe fiecare legătură un singur port desemnat. Ce rămâne se blochează.

Figura 12.8 Alegerea rădăcinii și rolurile porturilor.

Coborâți prioritatea unui switch la 0 în simulatorul de mai sus și priviți cum se reorganizează **întreaga** topologie în jurul lui. Exact asta face atacatorul - doar că switch-ul cu prioritate 0 este laptopul lui, aflat sub birou.

1. Atacatorul se conectează la rețeaua de switch-uri, cu un singur cablu.
2. Anunță BPDU-uri cu prioritatea 0, deci cu cel mai mic BID posibil.
3. STP reconverge; el devine rădăcină, iar rolurile porturilor se recalculează în jurul lui.
4. Traficul dintre segmente trece acum prin mașina lui, unde poate fi capturat.

Varianta de blocare a serviciului este și mai simplă: atacatorul își **schimbă continuu BID-ul**, forțând recalculări permanente. Porturile nu ajung niciodată în starea forwarding, iar rețeaua este, practic, oprită - cu un singur cablu și niciun exploit.

```

2 ! porturile spre statii: trec direct in forwarding...
3 SW(config)# interface range fastEthernet 0/1 - 22
4 SW(config-if-range)# spanning-tree portfast
5
6 ! ...si se inchid daca de acolo apare un BPDU
7 SW(config-if-range)# spanning-tree bpduguard enable
8
9 ! radacina se stabileste explicit , pe switch-ul din centrul
  ➔ rețelei
10 SW_CORE(config)# spanning-tree vlan 1-100 root primary
11
12 ! porturile spre switch-uri care nu au voie sa devina radacina
13 SW(config)# interface gigabitEthernet 0/2
14 SW(config-if)# spanning-tree guard root
15
16 ! reactivarea automata a porturilor inchise , dupa 5 minute
17 SW(config)# errdisable recovery cause bpduguard
18 SW(config)# errdisable recovery interval 300

```

Tabela 12.4 Atacuri împotriva STP.

| Mecanism           | Ce face                                                | Unde se pune                                                 |
|--------------------|--------------------------------------------------------|--------------------------------------------------------------|
| <b>PortFast</b>    | sare peste listening și learning pe porturile de acces | porturi spre stații; <b>nicio-dată</b> spre switch-uri       |
| <b>BPDU Guard</b>  | închide portul dacă primește un BPDU                   | împreună cu PortFast, întotdeauna                            |
| <b>Root Guard</b>  | împiedică un port să devină port rădăcină              | pe porturile spre switch-uri care nu au voie să fie rădăcină |
| <b>BPDU Filter</b> | oprește complet trimiterea și primirea de BPDU-uri     | rare; de folosit cu prudență, poate <b>crea</b> bucle        |

**Observație**

**PortFast fără BPDU Guard este periculos.** PortFast singur nu protejează pe nimeni: doar grăbește trecerea în forwarding. Dacă cineva conectează un switch pe acel port, se poate forma o buclă înainte ca STP să apuce să reacționeze.

Cele două se pun întotdeauna împreună. Regula mnemotehnică: *PortFast promite că nu e niciun switch acolo; BPDU Guard verifică promisiunea.*

**12.9 Parole, hash-uri și salt**

De ce nu se stochează parolele, ci hash-urile lor. Un **hash** este o funcție într-un singur sens: din parolă se calculează instantaneu, dar din rezultat nu se poate reveni

Tabela 12.5 Parole, hash-uri și salt.

| Metodă            | Cum funcționează                                                         | Contramăsură                         |
|-------------------|--------------------------------------------------------------------------|--------------------------------------|
| Sniffing          | parolele trimise în clar (Telnet, FTP, HTTP) se citesc direct din trafic | SSH, HTTPS - orice cu criptare       |
| Brute force       | se încearcă toate combinațiile posibile                                  | parole lungi; limitarea încercărilor |
| Atac de dicționar | se încearcă cuvinte uzuale și variații simple ale lor                    | parole care nu sunt cuvinte          |
| Rainbow tables    | lanțuri de hash-uri precalculate, care schimbă spațiu pe timp            | <b>salt</b>                          |

la parolă. Serverul stochează doar hash-ul. La autentificare, calculează hash-ul parolei introduse și compară. Dacă baza de date este furată, atacatorul nu are parolele - are doar hash-urile. Problema: dacă doi utilizatori au aceeași parolă, au și **același hash**. Iar un atacator poate precalcula hash-urile celor mai frecvente câteva milioane de parole și le poate căuta direct. Aici intră în scenă rainbow tables.

O **rainbow table** nu stochează toate hash-urile posibile - ar fi imposibil - ci punctele de pornire ale unor lanțuri de hash-uri, din care valorile intermediare se recalculează la nevoie. Este un compromis clasic memorie-timp, iar tabele publice de mii de gigaocteți sunt disponibile pe Internet.

### 12.9.1 Salt-ul

Un **salt** este un segment aleator, generat pentru fiecare utilizator și concatenat la parolă înainte de hashing. Este stocat alături de hash, în clar - nu este secret și nici nu trebuie să fie.

Exemplu 12.5 o linie din /etc/shadow

```

1
2 trudy:$6$/tKy92iM$/.cIxbEX49qHpZt74D5L0W1vXO2fJuXjyXJnsT0.M...
3 #      ^      ^      ^
4 #      |      |      hash-ul propriu-zis
5 #      |      salt-ul, generat aleator
6 #      algoritmul: 6 = SHA-512

```

#### Observație

**Ce face și ce nu face salt-ul.** Salt-ul **nu previne** atacurile cu rainbow tables și nu face parola mai greu de ghicit.

Ce face: obligă atacatorul să construiască o tabelă nouă **pentru fiecare utilizator în parte** - ceea ce transformă un atac de câteva minute asupra întregii baze de date într-un



efort separat per cont. Și face ca doi utilizatori cu aceeași parolă să aibă hash-uri diferite. Este o măsură de **scalare**, nu de imposibilitate: o parolă slabă rămâne slabă și cu salt. Pentru rezistență reală se folosesc funcții deliberat **lente** - bcrypt, scrypt, Argon2 - care fac fiecare încercare costisitoare.

## 12.10 Factorul uman și codul ostil

### 12.10.1 Ingineria socială

Cea mai eficientă cale de acces nu este tehnică. Atacatorul convinge o persoană că este de încredere - un coleg de la IT, un furnizor, un superior grăbit - și obține direct informația. Ținta nu are, de regulă, profil tehnic, vrea să ajute și nu conștientizează valoarea a ceea ce deține.

Ingineria socială **ocolește** orice mecanism tehnic, oricât de bine configurat. Nu există port security pentru un telefon. Singura apărare reală este instruirea periodică a angajaților non-tehnici și existența unor **proceduri clare** pentru cererile neobișnuite - de exemplu, regula că o parolă nu se comunică niciodată telefonic, indiferent cine sună.

**Phishing**-ul este varianta industrializată a aceleiași idei: în loc de un apel, un milion de mesaje.

### 12.10.2 Cod executabil ostil

Tabela 12.6 Cod executabil ostil.

| Tip               | Cum se propagă                                                  | Ce îl caracterizează                                                   |
|-------------------|-----------------------------------------------------------------|------------------------------------------------------------------------|
| <b>Virus</b>      | atașat unui program; are nevoie ca utilizatorul să îl ruleze    | efect direct, adesea vizibil                                           |
| <b>Troian</b>     | ascuns într-o aplicație aparent utilă                           | efect subtil - de regulă deschide o porțiță de acces; greu de detectat |
| <b>Vierme</b>     | exploatează vulnerabilități; <b>nu</b> are nevoie de utilizator | răspândire foarte rapidă; adesea construiește botnet-uri               |
| <b>Ransomware</b> | oricare din căile de mai sus                                    | criptează datele și cere răscumpărare; se propagă lateral în LAN       |

**De ce segmentarea contează atât de mult azi.** Un ransomware modern nu se mulțumește cu stația infectată: se propagă **lateral**, exploatănd exact încrederea implicită din rețeaua locală.

VLAN-urile, ACL-urile între ele și măsurile din acest curs nu împiedică infecția inițială - dar limitează dramatic cât de departe ajunge. Este diferența dintre un incident și un dezastru.

## 12.11 Lista scurtă de securizare a unui switch

Tot ce s-a discutat mai sus se rezumă la câteva configurații care se aplică oricărui switch de acces, în orice rețea de organizație.

| MĂSURA                               | ATACUL PE CARE ÎL ÎNCHIDE              |
|--------------------------------------|----------------------------------------|
| port-security cu maximum și sticky   | double tagging                         |
| ip dhcp snooping                     | CAM overflow, dispozitive neautorizate |
| ip arp inspection                    | preluarea rădăcinii STP                |
| switchport mode access + nonegotiate | servere DHCP false                     |
| VLAN nativ nefolosit                 | switch spoofing prin DTP               |
| portfast + bpduguard                 | ARP poisoning, man-in-the-middle       |

Apasă un element din stânga, apoi perechea lui din dreapta.

Figura 12.9 Măsura și atacul corespunzător.

Tabela 12.7 Lista scurtă de securizare a unui switch.

| Măsură                                          | Împotriva cui                           |
|-------------------------------------------------|-----------------------------------------|
| port-security cu maximum și sticky              | CAM overflow, dispozitive neautorizate  |
| ip dhcp snooping                                | servere DHCP false                      |
| ip arp inspection                               | ARP poisoning, man-in-the-middle        |
| switchport mode access + nonegotiate            | switch spoofing                         |
| VLAN nativ nefolosit, VLAN-uri permise explicit | double tagging                          |
| portfast + bpduguard                            | atacuri STP                             |
| porturile nefolosite: shutdown, în VLAN mort    | conectări neautorizate                  |
| SSH în loc de Telnet, cu ACL pe liniile VTY     | interceptarea parolelor de administrare |

Exemplu 12.6 administrare prin SSH, nu prin Telnet copiază

```

1
2 SW(config)# hostname SW1
3 SW(config)# ip domain-name retele.local
4 SW(config)# crypto key generate rsa modulus 2048
5 SW(config)# ip ssh version 2
6 SW(config)# username admin privilege 15 secret ParolaPuternica
7 SW(config)# service password-encryption
8
9 SW(config)# access-list 20 permit 10.0.100.0 0.0.0.255

```

```

10 SW(config)# line vty 0 15
11 SW(config-line)# transport input ssh
12 SW(config-line)# login local
13 SW(config-line)# access-class 20 in
14 SW(config-line)# exec-timeout 5 0

```

1 Setează explicit modul access și dezactivează negocierea DTP

2 Pornește DHCP snooping și marchează porturile de încredere

3 Închide porturile nefolosite și mută-le într-un VLAN mort

4 Activează port security, cu un maxim rezonabil de adrese

5 Înlocuiește Telnet cu SSH și limitează accesul cu access-class

6 Pune portfast și bpduguard pe toate porturile de acces

7 Activează Dynamic ARP Inspection pe VLAN-urile de utilizatori

Mută rândurile cu săgeți până ajung în ordinea corectă.

Figura 12.10 Ordinea de securizare a unui switch nou.

## 12.12 Greșeli frecvente

### Observație

#### Greșeli frecvente.

- **„Am pus portfast pe toate porturile, inclusiv spre switch-uri"** Se poate forma o buclă înainte ca STP să reacționeze - și o buclă de nivel 2 oprește rețeaua în câteva secunde. PortFast **doar** spre stații, întotdeauna cu BPDU Guard alături.
- **„Am activat DAI, dar serverele cu adrese statice nu mai comunică"** DAI se bazează pe evidența construită de DHCP snooping. O stație care nu a folosit DHCP nu apare acolo. Adăugați intrări manuale cu `arp access-list` pentru serverele cu adresă fixă.
- **„Am pus port-security maximum 1 peste tot"** Un telefon IP cu PC în spate are două adrese; o mașină virtuală în bridge adaugă altele. Alegeți maximum după ce chiar se conectează acolo. 2 sau 3 e mai realist.
- **„Am lăsat VLAN 1 ca VLAN nativ"** Este valoarea implicită și, adesea, chiar VLAN-ul în care se află stațiile - preconditiona exactă a atacului double tagging. VLAN nativ dedicat, gol, diferit de 1 și de orice VLAN de utilizatori.
- **„Am configurat totul, dar nu am salvat"** Adresele sticky și întreaga configurație dispar la prima repornire. `copy running-config startup-config`. Aceeași lecție ca în cursul 6.
- **„Port security cu violation shutdown, și acum jumătate din porturi sunt**

**închise"** Un utilizator care își schimbă laptopul închide portul, iar reactivarea cere intervenție manuală. **restrict** pentru zonele obișnuite; **errdisable recovery** acolo unde folosiți **shutdown**.

- „**Avem firewall, deci rețeaua internă e sigură**" Firewall-ul nu vede traficul dintre două stații din același VLAN. Securitate în adâncime: segmentare, măsuri pe switch, criptare între capete.

### 12.13 Rezumat și glosar

Ce trebuie să rămână. Firewall-ul de margine **nu vede** traficul din interiorul unui VLAN. Aproape toate atacurile din LAN exploatează protocoale care funcționează **corect**. **CAM overflow** falsifică adrese MAC **sursă** → se închide cu **port-security**. **ARP poisoning** exploatează lipsa autentificării în ARP → se închide cu **DHCP snooping** + **DAI**. **Switch spoofing** exploatează DTP → **nonegotiate**. **Double tagging** exploatează VLAN-ul nativ → VLAN nativ nefolosit. **Atacul STP** exploatează lipsa autentificării BPDU → **PortFast** + **BPDU Guard**. **Salt-ul** nu întărește parola: obligă atacatorul la o tabelă per cont. **Ingineria socială** ocolește orice măsură tehnică. Se apără cu proceduri și instruire. Securitatea este **în adâncime**: nicio măsură singură nu ajunge.

### 12.14 Încheiere

Cu acest curs se închide parcursul. Am pornit, în cursul 1, de la un semnal electric pe un fir și de la întrebarea cum se transformă un bit într-o tensiune. Am ajuns la o rețea locală segmentată în VLAN-uri, rutată dinamic, filtrată cu liste de acces, translatată la ieșire, extinsă fără fir și apărută împotriva atacurilor din interiorul ei.

Ce merită să rămână, dincolo de comenzi: aproape fiecare mecanism pe care l-am studiat a fost inventat ca **soluție** la o problemă concretă, iar aproape fiecare a devenit, ulterior, **problema** altcuiva. Switch-ul a rezolvat coliziunile și a creat CAM overflow. VLAN-ul a rezolvat difuzarea și a creat VLAN hopping. NAT-ul a salvat IPv4 și a rupt peer-to-peer. Nu există decizie de proiectare fără cost - există doar decizii luate în cunoștință de cauză și decizii luate din obișnuință.

**Laboratorul 7** execută, controlat, trei dintre atacurile de aici și apoi le închide, pe rând, ușile.

- Cisco - *Catalyst Switch Security Configuration Guide*: port security, DHCP snooping, DAI
- RFC 2827 / BCP 38 - filtrarea la intrarea în rețea împotriva falsificării adreselor
- RFC 3704 - filtrarea la intrare pentru rețele cu legături multiple
- OWASP - resurse despre parole, hashing și autentificare
- Documentația **nmap** și **Wireshark** - pentru uz strict autorizat

### Întrebări de verificare

1. Ce adrese MAC trebuie falsificate într-un atac de tip CAM overflow?

- a) Adresele destinație
  - b) Adresele sursă, pentru că acestea sunt cele învățate de switch
  - c) Adresa de broadcast
  - d) Adresa punctului de acces
2. Care este simptomul cel mai vizibil al unui CAM overflow reușit?
- a) Porturile se închid
  - b) Switch-ul începe să inunde traficul pe toate porturile, deci se comportă ca un hub
  - c) Adresele IP se schimbă
  - d) Se pierde legătura cu gateway-ul
3. De ce funcționează ARP poisoning?
- a) Pentru că switch-urile nu au tabelul CAM
  - b) Pentru că ARP nu autentifică nimic, iar stațiile acceptă răspunsuri pe care nu le-au cerut
  - c) Pentru că IP-ul nu are checksum
  - d) Pentru că VLAN-urile nu sunt configurate
4. Pe ce se bazează Dynamic ARP Inspection ca să decidă dacă un mesaj ARP este fals?
- a) Pe o listă de adrese MAC configurată manual
  - b) Pe evidența perechilor port-MAC-IP construită de DHCP snooping
  - c) Pe tabelul de rutare
  - d) Pe numărul VLAN-ului
5. Ce condiție trebuie îndeplinită pentru un atac de tip double tagging?
- a) Atacatorul trebuie să aibă acces fizic la trunk
  - b) VLAN-ul atacatorului trebuie să coincidă cu VLAN-ul nativ al trunk-ului
  - c) Switch-ul trebuie să ruleze RSTP
  - d) Toate VLAN-urile trebuie să aibă același ID
6. De ce atacul de tip double tagging este unidirecțional?
- a) Pentru că switch-urile filtrează răspunsurile
  - b) Pentru că răspunsul ar pleca din VLAN-ul tinta și nu are cum să ajungă înapoi la atacator
  - c) Pentru că eticheta interioară se pierde

- d) Pentru ca STP blocheaza calea de intoarcere
7. Ce combinatie de comenzi protejeaza cel mai bine un port de acces impotriva unui atac STP?
- a) spanning-tree portfast, singur
  - b) spanning-tree portfast impreuna cu spanning-tree bpduguard enable
  - c) spanning-tree bpdufilter
  - d) switchport nonegotiate
8. Ce realizeaza folosirea unui salt la stocarea parolelor?
- a) Face imposibila spargerea parolei
  - b) Obliga atacatorul sa construiasca o rainbow table separata pentru fiecare utilizator
  - c) Cripoteaza parola
  - d) Elimina nevoia de parole complexe
9. Care afirmatie despre firewall-ul de margine este corecta?
- a) Protejeaza si traficul dintre doua statii din acelasi VLAN
  - b) Nu vede traficul dintre doua statii din acelasi VLAN, pentru ca acesta nu il traverseaza
  - c) Inlocuieste nevoia de port security
  - d) Detecteaza automat ARP poisoning



# 13. Laborator 1: Primii pași: topologie, adresare și primul ping

---

---

Dacă nu ați mai făcut asta niciodată  
Echipamente necesare  
Cinci cuvinte, o dată pentru totdeauna  
Prima rețea: două calculatoare  
Topologia completă  
Linia de comandă, în trei minute  
Pași de lucru  
Ce poartă un pachet, segment cu segment  
Trei defecte de diagnosticat  
Sarcini de lucru  
Provocare  
Ce se predă  
Întrebări de verificare

---

---

Această lucrare nu presupune nimic știut dinainte. Construim o rețea de la zero: două calculatoare, un switch, un ruter și un plan de adresare. Fiecare comandă este scrisă întreagă, alături de ce trebuie să apară pe ecran după ea. La final încetinim rețeaua până la viteza gândului și urmărim un singur ping prin fiecare echipament.

## 13.1 Dacă nu ați mai făcut asta niciodată

O rețea de calculatoare este formată din trei feluri de lucruri. Atât. Tot restul cursului adaugă detalii peste această schemă.

**Ce facem astăzi, în două fraze.** Construim două rețele mici, le dăm adrese și le legăm printr-un ruter. Apoi verificăm, cu o singură comandă, că un calculator din prima rețea poate vorbi cu un server din a doua - și ne uităm la ce se întâmplă de fapt când o face.

### 13.1.1 Două moduri de a lucra

Aveți de ales, iar ambele variante sunt acceptate la predare:

**Ce veți putea face după acest capitol.**

- **Bancul de lucru din această pagină.** Nu se instalează nimic. Echipamentele sunt simulate aici, în browser, cu aceleași comenzi ca pe un echipament adevărat. Începeți



Tabela 13.1 Dacă nu ați mai făcut asta niciodată.

| Ce                                             | Rolul, într-o propoziție                                                                    | Exemplu din viața de zi cu zi                    |
|------------------------------------------------|---------------------------------------------------------------------------------------------|--------------------------------------------------|
| <b>Stații</b><br>(PC, laptop, server, telefon) | Emit și primesc date. Ele sunt motivul pentru care rețeaua există.                          | Calculatorul de pe care citiți pagina.           |
| <b>Switch</b>                                  | Leagă între ele stațiile din aceeași rețea și trimite fiecare mesaj numai către destinatar. | Priza multiplă inteligentă din perete, la birou. |
| <b>Ruter</b>                                   | Leagă rețele diferite între ele. Fără ruter, o rețea nu poate vorbi cu alta.                | Cutia de la furnizorul de internet, de acasă.    |

direct de la secțiunea 4.

- **Cisco Packet Tracer**, instalat pe calculatorul vostru. Este mai aproape de realitate, are modul de simulare vizuală și veți avea nevoie de el la laboratoarele următoare. Instalarea cere un cont gratuit de *Cisco Networking Academy*.

**Recomandarea mea.** Faceți pașii întâi în bancul de lucru din pagină, unde greșelile nu costă nimic și obiectivele se bifează singure. Apoi refaceți aceeași topologie în Packet Tracer, pentru fișierul .pkt care se predă. A doua oară durează un sfert din timp.

### 13.2 Echipamente necesare

Lista completă pentru această lucrare. În Packet Tracer, echipamentele se iau din bara de jos: alegeți întâi **categoria** din stânga, apoi **modelul** din dreapta, apoi dați clic în spațiul de lucru.

Tabela 13.2 Echipamente necesare.

| Buc. | Echipament     | Model                      | Categoria din Packet Tracer   | La ce folosește aici                 |
|------|----------------|----------------------------|-------------------------------|--------------------------------------|
| 1    | Ruter          | 4331                       | Network Devices<br>→ Routers  | leagă cele două rețele               |
| 2    | Switch         | 2960                       | Network Devices<br>→ Switches | leagă stațiile din fiecare rețea     |
| 3    | Calculator     | PC-PT                      | End Devices → End Devices     | stațiile PC0, PC1, PC2               |
| 1    | Server         | Server-PT                  | End Devices → End Devices     | destinația din rețeaua a doua        |
| 6    | Cablu de cupru | Copper<br>Straight-Through | Connections → Connections     | toate legăturile din această lucrare |

În plus, pentru voi: un caiet sau un fișier în care notați planul de adresare și răspunsurile,

și aproximativ două ore.

### 13.2.1 Ce cablu, între ce echipamente

Regula are o singură idee: două echipamente **de același fel** au nevoie de un cablu care inversează firele; două echipamente **de feluri diferite**, nu.

Tabela 13.3 Ce cablu, între ce echipamente.

| Între              | Cablu                   | De ce                            |
|--------------------|-------------------------|----------------------------------|
| PC ↔ switch        | Copper Straight-Through | echipamente de niveluri diferite |
| switch ↔ ruter     | Copper Straight-Through | tot niveluri diferite            |
| switch ↔ switch    | Copper Cross-Over       | același nivel                    |
| PC ↔ ruter, direct | Copper Cross-Over       | același nivel                    |

**Dacă nu vrei să vă gândiți.** Pictograma cu fulger („Automatically Choose Connection Type”) alege singură cablul potrivit. Pentru această lucrare alegeți-l manual - este singurul mod în care regula se fixează. De la laboratorul 3 puteți folosi fulgerul liniștit.

#### Observație

**Cum se leagă efectiv un cablu în Packet Tracer.** Clic pe tipul de cablu → clic pe primul echipament → din meniul care apare alegeți *portul* (de exemplu FastEthernet0) → clic pe al doilea echipament → alegeți și acolo portul. Dacă nu apare niciun meniu, ați ratat echipamentul; apăsați Esc și reluați.

### 13.3 Cinci cuvinte, o dată pentru totdeauna

Restul lucrării folosește aceste cinci noțiuni la fiecare pas. Nu treceți mai departe până nu vă sună familiare.

#### Observație

**adresă IP.** Numărul unic al unei stații în rețea, scris ca patru grupuri de la 0 la 255: 192.168.1.10. Este ca adresa poștală: cine nu are una, nu primește nimic.

#### Observație

**mască de subrețea.** Spune care parte din adresă înseamnă „strada” și care „numărul casei”. Cu masca 255.255.255.0, primele trei grupuri sunt strada: 192.168.1.10 și 192.168.1.11 sunt vecini, iar 192.168.2.10 este pe altă stradă.

#### Observație

**gateway implicit.** Adresa ruterului din propria rețea. Când destinația este pe altă stradă, stația trimite totul acolo și lasă ruterul să se descurce. Fără gateway, o stație vorbește doar cu vecinii ei.

**Observație**

**adresă MAC.** Numărul din fabrică al plăcii de rețea, de forma 00D0.B0AB.0001. Nu se schimbă și nu are legătură cu locul în care se află echipamentul. Switch-urile lucrează numai cu ea.

**Observație**

**ping.** Comanda care întreabă „ești acolo?” și așteaptă răspuns. Dacă primiți **Reply from...**, comunicația funcționează în ambele sensuri. Este primul lucru pe care îl încercă orice inginer de rețea, la orice problemă.

**13.4 Prima rețea: două calculatoare**

Înainte de topologia completă, o rețea cât se poate de mică: două calculatoare și un switch. Aici învățați mecanica - mai departe totul este repetiție.

În bancul de lucru de mai jos, dați clic pe **PC0**, completați în panoul din dreapta adresa 192.168.10.1 și masca 255.255.255.0, apoi apăsați *aplică*. Faceți la fel pentru **PC1**, cu adresa 192.168.10.2. Reveniți pe PC0 și scrieți în terminal:

Exemplu 13.1 pe PC0 copiază

```
1
2 ping 192.168.10.2
```

**Ce tocmai s-a întâmplat.** Nu ați configurat absolut nimic pe switch și totuși comunicația funcționează. Un switch nu are nevoie de configurare ca să comute: învață singur, din traficul care trece prin el, ce adresă MAC se află pe ce port. Verificați: alegeți SW1, scrieți **enable**, apoi **show mac address-table**.

**Exemplu**

Schimbați adresa lui PC1 în 192.168.20.2, păstrând masca. Mai merge ping-ul? De ce? **Rezolvare.** Nu mai merge. Cu masca 255.255.255.0, primele trei grupuri definesc rețeaua: 192.168.10.x și 192.168.20.x sunt două rețele diferite. PC0 constată că destinația nu este la el pe stradă, caută un gateway - și nu are niciunul, deci renunță. Este exact motivul pentru care în secțiunea următoare apare un ruter.

**13.5 Topologia completă****13.5.1 Planul de adresare**

Acesta este tabelul pe care îl completați înainte de a atinge orice echipament. În practică, un plan de adresare scris dinainte previne mai multe probleme decât orice comandă de depanare.

**De ce switch-urile nu au adresă.** Un switch nu are nevoie de adresă IP ca să comute cadre - el lucrează la nivelul 2. O adresă i se dă doar pentru administrare de la distanță, iar aceea aparține interfeței virtuale VLAN 1, nu funcției de comutare. O vom configura în laboratorul 2.

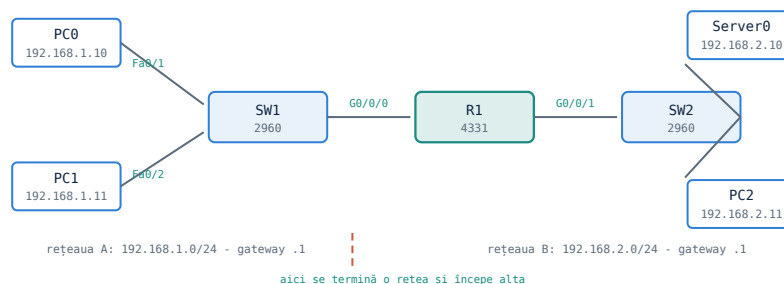


Figura 13.1 Două rețele separate de un ruter. Toate stațiile din stânga se aud direct între ele; pentru a ajunge în dreapta, trebuie să treacă prin R1.

Tabela 13.4 Planul de adresare.

| Dispozitiv | Interfață | Adresă IP    | Mască         | Gateway     |
|------------|-----------|--------------|---------------|-------------|
| R1         | G0/0/0    | 192.168.1.1  | 255.255.255.0 | -           |
| R1         | G0/0/1    | 192.168.2.1  | 255.255.255.0 | -           |
| PC0        | Fa0       | 192.168.1.10 | 255.255.255.0 | 192.168.1.1 |
| PC1        | Fa0       | 192.168.1.11 | 255.255.255.0 | 192.168.1.1 |
| Server0    | Fa0       | 192.168.2.10 | 255.255.255.0 | 192.168.2.1 |
| PC2        | Fa0       | 192.168.2.11 | 255.255.255.0 | 192.168.2.1 |
| SW1, SW2   | -         | -            | -             | -           |

### Observație

**Convenția care vă scutește de confuzii.** În toate laboratoarele, ruterul primește prima adresă utilizabilă din rețea (.1), iar stațiile primesc adrese de la .10 în sus. Prima adresă a blocului (.0) este adresa rețelei, iar ultima (.255) este adresa de broadcast - niciuna nu se dă unei stații.

## 13.6 Linia de comandă, în trei minute

Ruterele și switch-urile nu au ecran și nici mouse. Se configurează scriind comenzi într-o consolă, în sistemul de operare **Cisco IOS**. Consola arată intimidant două minute, apoi devine cea mai rapidă unealtă pe care o veți folosi.

### 13.6.1 Cum se citește promptul

Textul dinaintea cursorului vă spune întotdeauna **pe ce echipament** sunteți și **în ce mod**. Nu se scrie niciodată de mână: îl afișează echipamentul.

Se coboară cu **exit** (un nivel) sau **end** (direct în modul privilegiat).

**Trei obiceiuri care vă scutesc de jumătate din timpul pierdut.** *Tab* completează comanda începută. *?* arată variantele posibile în punctul curent - inclusiv în mijlocul unei comenzi. Iar comenzile pot fi prescurtate cât timp rămân neambigue: **conf t** în loc de **configure terminal**, **int g0/0/0** în loc de **interface gigabitEthernet 0/0/0**, **no shut** în loc de **no shutdown**. Toate trei funcționează și în bancul de lucru din pagină.

Tabela 13.5 Cum se citește promptul.

| Prompt             | Mod                      | Cum se intră              | Ce se poate face                                |
|--------------------|--------------------------|---------------------------|-------------------------------------------------|
| Router>            | utilizator               | la conectare              | doar câteva comenzi de afișare                  |
| Router#            | privilegiat              | <b>enable</b>             | toate comenzile <b>show</b> , salvare, depănare |
| Router(config)#    | configurare globală      | <b>configure terminal</b> | modifică echipamentul în ansamblu               |
| Router(config-if)# | configurare de interfață | <b>interface G0/0/0</b>   | modifică o singură interfață                    |

### Observație

Trei mesaje de eroare pe care le veți vedea sigur. % Invalid input detected at '^' marker. - cuvântul de sub semnul ^ este greșit sau nu există în modul curent. % Incomplete command. - comanda este corectă, dar neterminată; apăsați ? ca să vedeți ce mai lipsește. % Ambiguous command - prescurtarea se potrivește cu mai multe comenzi; mai scrieți o literă.

## 13.7 Pași de lucru

Bancul de lucru de mai jos are topologia completă din figura 1, deja cablată. Urmăți pașii în ordine; lista de obiective din dreapta se bifează singură pe măsură ce avansați.

1. **Construiți topologia** doar în Packet Tracer Plasați cele două switch-uri, ruterul, cele trei stații și serverul. Legați-le exact ca în figura 1, cu cablu **Copper Straight-Through**. Salvați fișierul ca **lab01\_nume.pkt** și obișnuți-vă să salvați des. Priviți culoarea capetelor de cablu: **verde** înseamnă legătură activă, **roșu** înseamnă că legătura nu s-a stabilit, iar **portocaliu**, între două switch-uri, înseamnă că portul trece prin stările STP și va deveni verde în aproximativ 30 de secunde. În bancul de lucru din pagină topologia este deja cablată; treceți la pasul 2.
2. **Configurați cele patru stații** În Packet Tracer: clic pe PC0 → fila *Desktop* → *IP Configuration*. În bancul din pagină: clic pe PC0 → panoul *Desktop* → *IP Configuration* din dreapta. În ambele cazuri completați cele trei câmpuri din planul de adresare și repetați pentru PC1, PC2 și Server0. Verificați, din *Desktop* → *Command Prompt* (sau direct în terminalul bancului): pe PC0 copiază ipconfig ping 192.168.1.11 Al doilea ping **trebuie** să funcționeze deja: PC0 și PC1 sunt în aceeași rețea, iar switch-ul îi leagă direct - ruterul nu are niciun rol aici. Rezultatul arată așa: ce trebuie să apară Pinging 192.168.1.11 with 32 bytes of data: Request timed out. Reply from 192.168.1.11: bytes=32 time<1ms TTL=128 Reply from 192.168.1.11: bytes=32 time<1ms TTL=128 Reply from 192.168.1.11: bytes=32 time<1ms TTL=128 Ping

statistics for 192.168.1.11: Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),  
Primul pachet se pierde aproape întotdeauna. Nu este o defecțiune - este ARP, iar la  
pasul 8 vedeți exact de ce.

3. **Configurați ruterul** În Packet Tracer: clic pe R1 → fila *CLI* → apăsați **Enter**. Dacă  
vă întreabă „*Would you like to enter the initial configuration dialog?*”, răspundeți **no**.  
În bancul din pagină: clic pe R1 și scrieți direct în terminal. Scrieți comenzile de  
mai jos **rând cu rând**, apăsând Enter după fiecare. Promptul se schimbă pe parcurs -  
comparați-l cu ce vedeți la stânga fiecărei linii: configurația de bază a lui R1copiază Ro-  
uter> enable Router# configure terminal Router(config)# hostname R1 R1(config)#  
no ip domain-lookup R1(config)# interface gigabitEthernet 0/0/0 R1(config-if)# des-  
cription Legatura spre rețeaua A R1(config-if)# ip address 192.168.1.1 255.255.255.0  
R1(config-if)# no shutdown R1(config-if)# exit R1(config)# interface gigabitEther-  
net 0/0/1 R1(config-if)# description Legatura spre rețeaua B R1(config-if)# ip ad-  
dress 192.168.2.1 255.255.255.0 R1(config-if)# no shutdown R1(config-if)# end R1#  
copy running-config startup-config Ce face fiecare grup de comenzi: ComandaEfectul  
**enable** + **configure terminal** deschide dreptul de a modifica echipamentul **hostname**  
R1redenumeste ruterul; de acum promptul arată R1# **interface** ...intră în configura-  
rea unei singure interfețe fizice **ip address** ...dă interfeței adresa de gateway a rețelei  
ei **no shutdown**pornește interfața **copy running-config startup-config**salvează, ca  
munca să reziste la repornire **Cele două comenzi pe care le uită toată lumea no**  
**shutdown**: interfețele unui ruter sunt *închise implicit*, spre deosebire de cele ale unui  
switch. Fără această comandă, interfața are adresă, dar este moartă.  
**copy running-config startup-config**: fără ea, toată munca dispare la repornirea  
echipamentului. **running-config** este configurația din memoria de lucru; **startup-config**  
este cea citită la pornire.  
**no ip domain-lookup** nu este obligatorie, dar vă scutește de așteptarea de 30 de se-  
cunde care apare ori de câte ori scrieți greșit o comandă și IOS încearcă să o rezolve  
prin DNS.
4. **Verificați starea interfețelor** pe R1copiază R1# show ip interface brief Trebuie să  
vedeți exact asta - ambele interfețe configurate cu **up** pe ambele coloane: ce trebuie  
să apară Interface IP-Address OK? Method Status Protocol GigabitEthernet0/0/0  
192.168.1.1 YES manual up up GigabitEthernet0/0/1 192.168.2.1 YES manual up  
up GigabitEthernet0/0/2 unassigned YES unset administratively down down Prima  
coloană de stare se referă la nivelul fizic (există semnal pe fir?), a doua la protocolul  
de linie (legătura chiar funcționează?). Dacă nu vedeți asta, tabelul de mai jos vă  
spune ce lipsește: Ce vedețiCe înseamnăCe faceți **up** / **up**totul în regulămergeți mai  
departe **administratively down**lipsește **no shutdown**reintrăți pe interfață și dați **no**  
**shutdown down** / **down**fără semnalcablu lipsă, capăt greșit sau echipamentul din față  
e oprit **up** / **down**semnal există, protocolul nude obicei nepotrivire de încapsulare; rar  
în această lucrare **unassigned**interfața nu are adresăați configurat altă interfață decât

cea legată

5. **Priviți ce știe ruterul despre lume** pe R1copiază R1# show ip route Trebuie să apară exact **două** rute marcate C (rețele conectate direct) și adresele proprii marcate L (local). Ruterul cunoaște doar ce este legat de el - nimic mai mult, și nimeni nu i-a spus nimic altceva. Din laboratorul 5 începem să-i spunem.
6. **Testați conectivitatea, în ordine crescătoare** pe PC0copiază ping 192.168.1.1 ping 192.168.2.1 ping 192.168.2.10 tracert 192.168.2.10 Cele patru comenzi urcă în dificultate: gateway-ul propriu, interfața îndepărtată a ruterului, serverul din cealaltă rețea, apoi traseul complet. **Dacă una eșuează, problema este exact între ea și cea anterioară** - este cea mai rapidă metodă de a localiza un defect și o veți folosi tot semestrul. **tracert** trebuie să arate două salturi: întâi 192.168.1.1 (ruterul), apoi 192.168.2.10 (serverul).
7. **Priviți tabela ARP a stației** pe PC0copiază arp -a Veți vedea intrări pentru 192.168.1.11 (PC1, din aceeași rețea) și pentru 192.168.1.1 (gateway-ul). Nu veți vedea nimic pentru 192.168.2.10, deși tocmai i-ați dat ping cu succes. **De ce nu apare serverul în tabela ARP** Pentru că PC0 nu a vorbit niciodată direct cu el. A trimis cadrele către adresa MAC a gateway-ului, iar de acolo mai departe s-a ocupat ruterul. Adresa IP destinație a rămas cea a serverului; adresa MAC destinație a fost, tot timpul, a lui R1. Aceasta este regula celor două adrese, pe care o verificați experimental în secțiunea următoare.
8. **Priviți tabela MAC a switch-ului** pe SW1copiază Switch> enable Switch# show mac address-table Switch# clear mac address-table dynamic Switch# show mac address-table Prima afișare arată o tabelă populată, deși nimeni nu a configurat-o: s-a umplut singură, din traficul care a trecut. După **clear** este goală. Dați un ping de pe PC0 și afișați-o din nou - se umple la loc în mai puțin de o secundă.

### 13.8 Ce poartă un pachet, segment cu segment

Aceasta este partea pentru care merită făcut laboratorul. Un ping de la PC0 la Server0 traversează patru legături. Pe fiecare dintre ele, pachetul poartă o pereche de adrese IP și o pereche de adrese MAC. Întrebarea este: care dintre ele se schimbă pe drum?

#### 13.8.1 În bancul de lucru din pagină

Dați de pe PC0 comanda ping 192.168.2.10 și priviți tabelul **Traseul ultimului pachet**, care apare sub terminal. Are un rând pentru fiecare legătură traversată.

#### 13.8.2 În Packet Tracer

1. Comutați din *Realtime* în *Simulation*, din colțul din dreapta jos.
2. Apăsați *Edit Filters*, deselectați tot și lăsați bifate doar **ICMP** și **ARP**.

3. De pe PC0 dați `ping 192.168.2.10`, apoi apăsați *Capture / Forward* pas cu pas.
4. Faceți clic pe plic și apoi pe fila *Inbound PDU Details* ca să citiți adresele.

Ordinea evenimentelor este întotdeauna aceeași:

1. Primul pachet este un **ARP request**, în broadcast: PC0 caută adresa MAC a gateway-ului. Observați că switch-ul îl trimite pe toate porturile.
2. R1 răspunde cu un **ARP reply**, de data asta unicast, doar către PC0.
3. Abia acum pleacă **ICMP echo request** - adevăratul ping.
4. Urmăriți același pachet după ce iese din R1 pe G0/0/1. **Adresele IP sunt aceleași. Adresele MAC sunt complet diferite.**
5. Urmăriți răspunsul până înapoi la PC0.

Completați tabelul de mai jos pentru pachetul ICMP echo request, pe fiecare segment. Este unul dintre livrabilele lucrării:

Tabela 13.6 În Packet Tracer.

| Segment       | MAC sursă | MAC destinație | IP sursă | IP destinație |
|---------------|-----------|----------------|----------|---------------|
| PC0 → SW1     |           |                |          |               |
| SW1 → R1      |           |                |          |               |
| R1 → SW2      |           |                |          |               |
| SW2 → Server0 |           |                |          |               |

**Regula celor două adrese.** Adresa IP spune *unde vrea să ajungă* pachetul și nu se schimbă pe tot drumul. Adresa MAC spune *cui i se dă pachetul acum* și se rescrie la fiecare ruter traversat. Un switch nu schimbă niciuna: el doar citește MAC-ul destinație și alege portul.

### 13.9 Trei defecte de diagnosticat

Introduceți intenționat, pe rând, fiecare dintre defectele de mai jos - în bancul de lucru sau în Packet Tracer. Înainte de a repara, observați **exact ce simptom produce**. La laboratoarele următoare veți lucra invers: veți primi simptomul și veți căuta cauza.

În bancul de lucru, defectul al doilea se produce așa:

Exemplu 13.2 pe R1 copiază

```

1
2 R1# configure terminal
3 R1(config)# interface g0/0/1
4 R1(config-if)# shutdown
5 R1(config-if)# end

```



Tabela 13.7 Trei defecte de diagnosticat.

| Defect            | Cum îl produceți                | Simptom așteptat                                                                             | Comanda care îl scoate la iveală               |
|-------------------|---------------------------------|----------------------------------------------------------------------------------------------|------------------------------------------------|
| gateway lipsă     | ștergeți gateway-ul de pe PC0   | ping în rețeaua proprie merge, ping în cealaltă rețea dă <b>Destination host unreachable</b> | <b>ipconfig</b> pe stație                      |
| interfață închisă | <b>shutdown</b> pe G0/0/1 la R1 | PC0 ajunge la 192.168.1.1, dar nu mai departe                                                | <b>show ip interface brief</b>                 |
| mască greșită     | puneți masca 255.255.0.0 pe PC2 | ping-ul dinspre PC2 pleacă, dar răspunsul nu se mai întoarce                                 | <b>ipconfig</b> plus calculul adresei de rețea |

... iar reparația este aceeași comandă cu **no** în față. Observați că obiectivul „PC0 ajunge la Server0” se stinge și se reaprinde singur.

### Exemplu

La al treilea defect, de ce anume se rupe comunicația? PC2 are adresa 192.168.2.11 cu masca 255.255.0.0.

**Rezolvare.** Cu masca /16, PC2 consideră că rețeaua lui este 192.168.0.0/16 - care include și 192.168.1.0/24. Când vrea să răspundă lui PC0, crede că acesta este **în aceeași rețea** și încearcă un ARP direct pentru el, în loc să trimită cadrul spre gateway. Nimeni nu răspunde, pentru că PC0 este dincolo de ruter. Este exemplul perfect pentru de ce masca trebuie să fie identică pe toate stațiile unui segment: o mască diferită nu produce o eroare vizibilă la configurare, ci un comportament asimetric și derutant mult mai târziu.

## 13.10 Sarcini de lucru

**Ce veți putea face după acest capitol.**

- Construiți topologia din figura 1 și configurați-o complet, conform planului de adresare
- Obțineți ping reușit de la fiecare stație spre fiecare altă stație (12 teste)
- Completați tabelul cu adrese MAC și IP pe cele patru segmente
- Reproduceți cele trei defecte și notați simptomul fiecăruia, în cuvintele voastre
- Adăugați o a treia rețea, 192.168.3.0/24, pe interfața G0/0/2 a lui R1, cu un switch și două stații; verificați conectivitatea completă
- Salvați configurațiile și verificați cu **show startup-config**

### 13.11 Provocare

#### Exemplu

Eliminați ruterul din topologie și legați direct cele două switch-uri printr-un cablu crossover. Stațiile își păstrează adresele: 192.168.1.x în stânga, 192.168.2.x în dreapta. În bancul de lucru puteți face asta din panoul **Cablurile**: scoateți cele două cabluri ale lui R1 și legați SW1 direct de SW2.

Întrebări:

Mai funcționează ping-ul între PC0 și Server0? De ce? Ce se întâmplă dacă schimbați masca tuturor stațiilor în 255.255.0.0? Explicați rezultatul. Ce se întâmplă cu un broadcast emis de PC0 acum, comparativ cu topologia inițială? Este soluția de la punctul 2 acceptabilă într-o rețea reală? Argumentați folosind noțiunea de domeniu de broadcast.

### 13.12 Ce se predă

Tabela 13.8 Ce se predă.

| Livrabil                                                    | Format   | Punctaj |
|-------------------------------------------------------------|----------|---------|
| Fișierul Packet Tracer cu topologia completă și funcțională | .pkt     | 40 %    |
| Tabelul cu adrese MAC și IP pe cele patru segmente          | document | 20 %    |
| Descrierea celor trei defecte și a simptomelor observate    | document | 20 %    |
| Răspunsurile la provocare, cu argumentare                   | document | 20 %    |

Fișierul .pkt trebuie să conțină și a treia rețea cerută la sarcini, iar configurațiile să fie salvate în `startup-config`.

#### Întrebări de verificare

1. Ați configurat adresa pe o interfața de ruter, dar aceasta apare administrativ down. Ce lipsește?
  - a) Comanda `copy running-config startup-config`
  - b) Comanda `no shutdown`
  - c) Un cablu crossover
  - d) Configurarea gateway-ului
2. De ce primul ping dintre două stații eșuează aproape întotdeauna?
  - a) Pentru că rețeaua este supraincercată

- b) Pentru ca statia trebuie intai sa afle adresa MAC prin ARP, iar primul pachet ICMP expira in acest timp
  - c) Pentru ca switch-ul nu a pornit
  - d) Pentru ca masca este gresita
3. PC0 a dat cu succes ping serverului din alta retea. De ce nu apare serverul in arp -a?
- a) Pentru ca tabela ARP se sterge imediat
  - b) Pentru ca PC0 nu a vorbit niciodata direct cu el: cadrele au fost adresate MAC-ului gateway-ului
  - c) Pentru ca serverul are ARP dezactivat
  - d) Pentru ca ping-ul nu foloseste ARP
4. Ce se schimba si ce ramane la fel cand un pachet trece printr-un ruter?
- a) Se schimba adresele IP, raman adresele MAC
  - b) Se schimba adresele MAC, raman adresele IP
  - c) Se schimba amandoua
  - d) Nu se schimba nimic
5. Ce arata prima si a doua coloana de stare din show ip interface brief?
- a) Adresa IP si masca
  - b) Starea nivelului fizic si starea protocolului de linie
  - c) Numarul de pachete trimise si primite
  - d) VLAN-ul si viteza
6. O statie are adresa 192.168.1.10 cu masca 255.255.255.0 si niciun gateway. Cu cine poate vorbi?
- a) Cu nimeni
  - b) Doar cu statiile 192.168.1.x din propria retea
  - c) Cu toata lumea
  - d) Doar cu ruterul

## 14. Laborator 2: Comutarea în LAN: tabela MAC, ARP, domenii

---

---

- Ce demontăm astăzi
- Echipamente necesare
- Topologia de construit
- Bancul de lucru
- Tabela MAC, observată cum se umple
- Switch contra hub
- ARP, cu cache-ul golit
- Administrarea switch-ului
- Port security
- Sarcini de lucru
- Provocare
- Ce se predă
- Întrebări de verificare

---

---

Un switch primește un dispozitiv nou și, în câteva milisecunde, știe unde se află - fără ca nimeni să îi fi spus nimic. Lucrarea aceasta demontează mecanismul, pas cu pas: goliți tabela MAC, o umpleți privind-o, comparați switch-ul cu un hub pe aceeași topologie și vedeți la propriu ce înseamnă un domeniu de coliziune.

### 14.1 Ce demontăm astăzi

În laboratorul 1 ați dat un ping și a mers. Nimeni nu v-a explicat de ce switch-ul a știut unde să trimită cadrul. Astăzi aflați, iar răspunsul are doar trei reguli.

Există și a patra situație, mai rară: dacă destinația este pe chiar portul de intrare, switch-ul aruncă cadrul (*drop*) - cele două stații se aud oricum direct.

#### Observație

**domeniu de coliziune.** Porțiunea de rețea în care două stații nu pot transmite simultan fără să se încurce. Fiecare port de switch creează unul separat. Un hub le contopește pe toate într-unul singur - de aici vine toată diferența de performanță dintre cele două.

Tabela 14.1 Ce demontăm astăzi.

| Regula        | Ce face switch-ul                                                                                                                                     |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>învață</b> | La fiecare cadru care intră, notează <i>adresa MAC sursă</i> și portul pe care a intrat. Nimeni nu îi spune nimic: deduce singur.                     |
| <b>caută</b>  | Se uită la <i>adresa MAC destinație</i> . Dacă o găsește în tabelă, trimite cadrul <b>doar</b> pe portul acela ( <i>unicast</i> ).                    |
| <b>inundă</b> | Dacă nu o găsește - sau dacă destinația este broadcast - trimite cadrul pe <b>toate</b> porturile, în afară de cel pe care a intrat ( <i>flood</i> ). |

**Observație**

**domeniu de broadcast.** Porțiunea de rețea până la care ajunge un mesaj trimis „către toți”. Se oprește la un ruter (sau la granița unui VLAN, din laboratorul 3). Nici switch-ul, nici hub-ul nu îl opresc.

**Observație**

**ARP.** Protocolul care traduce o adresă IP într-o adresă MAC. O stație știe cu cine vrea să vorbească (192.168.5.11), dar are nevoie de numărul plăcii de rețea ca să compună cadrul. Întreabă în broadcast, primește răspuns unicast și ține minte răspunsul câteva minute.

**14.2 Echipamente necesare**

Tabela 14.2 Echipamente necesare.

| Buc. | Echipament     | Model                      | Categoria din Packet Tracer   | La ce folosește aici              |
|------|----------------|----------------------------|-------------------------------|-----------------------------------|
| 1    | Switch         | 2960                       | Network Devices<br>→ Switches | partea „inteligentă” a topologiei |
| 1    | Hub            | Hub-PT                     | Network Devices<br>→ Hubs     | termenul de comparație            |
| 6    | Calculator     | PC-PT                      | End Devices → End Devices     | PC-A ... PC-F                     |
| 7    | Cablu de cupru | Copper<br>Straight-Through | Connections → Connections     | toate legăturile                  |

Opțional, pentru ultima sarcină: încă un switch 2960 și un cablu Copper Cross-Over.

**Ce se poate face în bancul din pagină și ce cere Packet Tracer.** Bancul de lucru din această pagină acoperă: tabela MAC și golirea ei, comportamentul hub-ului, ARP,

adresa de management pe VLAN 1, port security cu violare provocată prin mutarea unui cablu.

Cer Packet Tracer: modul *Simulation* cu plicuri urmărite pas cu pas, măsurarea timpilor sub trafic simultan, sesiunea SSH propriu-zisă și îmbătrânirea reală a intrărilor în timp.

### 14.3 Topologia de construit

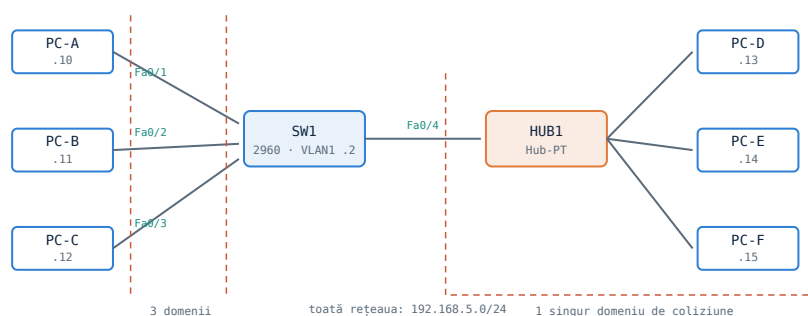


Figura 14.1 Aceeași rețea, două filozofii de interconectare. În stânga, fiecare stație are domeniul ei de coliziune. În dreapta, toate trei îl împart.

Tabela 14.3 Topologia de construit.

| Dispozitiv       | Adresă IP            | Mască         | Port pe SW1 / HUB1 |
|------------------|----------------------|---------------|--------------------|
| PC-A             | 192.168.5.10         | 255.255.255.0 | SW1 Fa0/1          |
| PC-B             | 192.168.5.11         | 255.255.255.0 | SW1 Fa0/2          |
| PC-C             | 192.168.5.12         | 255.255.255.0 | SW1 Fa0/3          |
| PC-D, PC-E, PC-F | 192.168.5.13 – .15   | 255.255.255.0 | HUB1               |
| SW1              | 192.168.5.2 (VLAN 1) | 255.255.255.0 | -                  |

Nu există ruter și nu este nevoie de gateway: toate stațiile sunt în aceeași rețea. Legătura SW1 Fa0/4 – HUB1 se face cu cablu straight-through.

### 14.4 Bancul de lucru

Topologia din figura 1, cablată și adresată. Toate secțiunile următoare se pot face aici; deschideți-l într-o filă alături de text și lucrați pe măsură ce citiți.

### 14.5 Tabela MAC, observată cum se umple

1. **Notați adresele MAC ale stațiilor** Pentru fiecare stație, deschideți terminalul (în Packet Tracer: *Desktop* → *Command Prompt*) și rulați: pe fiecare stație copiați ipconfig /all Notați câmpul *Physical Address* - este adresa MAC. Veți avea nevoie de aceste șase valori la fiecare pas care urmează; fără ele, tabela switch-ului este un șir de cifre fără sens.

2. **Goliți tabela și verificați că este goală** pe SW1copiază Switch> enable Switch# clear mac address-table dynamic Switch# show mac address-table Trebuie să nu apară nicio intrare. Dacă mai apar, înseamnă că a trecut deja trafic între timp - repetați cele două comenzi imediat una după alta.
3. **Un singur cadru, și priviți ce s-a întâmplat** De pe PC-A, dați un singur ping către PC-B, apoi reveniți imediat pe SW1: pe PC-Acopiază ping -n 1 192.168.5.11 pe SW1copiază SW1# show mac address-table Întrebare Câte intrări au apărut și de unde a aflat switch-ul fiecare dintre ele? Vezi răspunsul Două: MAC-ul lui PC-A pe Fa0/1 și MAC-ul lui PC-B pe Fa0/2. Prima a fost învățată din **cererea ARP** a lui PC-A - un broadcast, deci inundat pe toate porturile. A doua, din **răspunsul ARP** al lui PC-B. Ping-ul propriu-zis nu a mai avut nimic de învățat: până să plece, switch-ul știa deja totul. Concluzia care contează: switch-ul nu învață din ping. Învață din *orice* cadru care trece prin el, indiferent ce conține.
4. **Cele trei acțiuni, una câte una** Goliți din nou tabela și executați scenariile în ordinea de mai jos. În Packet Tracer treceți în *Simulation* și filtrați pe ICMP și ARP, ca să vedeți plicurile. În bancul din pagină, urmăriți tabela MAC după fiecare pas. ScenariuCe face switch-ulCum verificați PC-A → PC-B, cu tabela goală**flood**: cadrul iese pe toate porturile în afară de Fa0/1tabela era goală, deci destinația era necunoscută PC-B → PC-A, imediat după**unicast**: doar pe Fa0/1**show mac address-table** conține deja PC-A PC-A → broadcast**flood**, întotdeaunaîn Packet Tracer: ping 192.168.5.255
5. **Îmbătrânirea intrărilor** pe SW1copiază SW1# show mac address-table aging-time SW1# configure terminal SW1(config)# mac address-table aging-time 30 SW1(config)# end Valoarea implicită este de 300 de secunde. Reduceți-o la 30, generați trafic de la PC-C, apoi lăsați rețeaua liniștită și urmăriți tabela din 15 în 15 secunde: intrarea dispare. (Trecerea timpului se observă doar în Packet Tracer; în bancul din pagină comanda este acceptată, dar tabela nu îmbătrânește singură.) **De ce e nevoie de îmbătrânire** Dacă mutați PC-C de pe Fa0/3 pe Fa0/5, switch-ul ar continua să trimită traficul pentru el pe portul vechi - la nesfârșit. În practică, primul cadru emis de PC-C de pe noul port corectează imediat intrarea; îmbătrânirea acoperă cazul în care stația a plecat și nu mai vorbește deloc. Verificați asta chiar acum, în bancul de lucru: alegeți PC-C, în panoul **Cablurile** apăsați *scoate*, apoi *leagă* spre SW1 : Fa0/5. Dați un ping de pe PC-C și priviți din nou **show mac address-table**: intrarea s-a mutat singură pe noul port.

## 14.6 Switch contra hub

Partea din dreapta a topologiei există dintr-un singur motiv: să vedeți diferența, nu să v-o imaginați.

1. **Trafic prin hub** De pe PC-D dați ping către PC-E. În Packet Tracer, în *Simulation*, urmăriți plicul. Hub-ul îl trimite **pe toate porturile** - inclusiv la PC-F, care nu are

nicio treabă cu această conversație, și înapoi spre switch. PC-F primește cadrul, se uită la adresa MAC destinație, constată că nu îi aparține și îl aruncă. Timp și bandă consumate degeaba, la fiecare cadru. În bancul din pagină, verificați consecința pe switch: după un ping între două stații legate la hub, **show mac address-table** pe SW1 arată ambele adrese **pe același port**, Fa0/4. Switch-ul nu are cum să le separe: din punctul lui de vedere, tot ce e dincolo de hub stă pe un singur fir.

2. **Coliziunea, provocată** doar în Packet Tracer Reveniți în *Realtime*. Deschideți simultan terminalele pe PC-D și PC-F și lansați de pe ambele, cât mai apropiat în timp: pe PC-D și PC-F, aproape simultancopiază `ping -n 200 -l 1400 192.168.5.10` Comparați timpii de răspuns cu cei obținuți când rulați singur. Repetați experimentul înlocuind hub-ul cu un al doilea switch și observați diferența.
3. **Numărați domeniile** Exercițiu Pentru topologia din figura 1, câte domenii de coliziune și câte domenii de broadcast există? Vezi răspunsul **Domenii de coliziune: 5**. SW1 creează câte unul pe fiecare port: Fa0/1, Fa0/2, Fa0/3 și Fa0/4 - adică 4. Hub-ul, împreună cu PC-D, PC-E, PC-F și legătura spre switch, formează un singur domeniu - al cincilea. Atenție: legătura SW1-HUB1 aparține domeniului hub-ului, nu unuia separat. **Domenii de broadcast: 1**. Nu există niciun ruter și niciun VLAN, deci toate cele șase stații se aud în broadcast. Un `ping 192.168.5.255` de pe PC-A ajunge la toată lumea.

## 14.7 ARP, cu cache-ul golit

1. **Ștergeți cache-ul și priviți cererea** pe PC-Acopiază `arp -d arp -a ping -n 1 192.168.5.12` `arp -a` Prima afișare trebuie să spună **No ARP Entries Found**. După ping, apare adresa lui PC-C împreună cu MAC-ul ei. Acesta este întreg mecanismul: o întrebare, un răspuns, o notiță ținută minte câteva minute. În Packet Tracer, urmăriți cererea ARP în *Simulation* și citiți din *Outbound PDU Details*: adresa MAC destinație a cadrului: **FFFF.FFFF.FFFF** - broadcast; câmpul **Target MAC** din pachetul ARP: zerouri - exact ce se cere; câmpul **Target IP**: adresa lui PC-C. Toate stațiile primesc cererea. Doar PC-C răspunde; celelalte o ignoră tăcut.
2. **Efectul cache-ului** Dați `ping 192.168.5.12` imediat după `arp -d` și apoi încă o dată. Prima oară primul pachet se pierde; a doua oară, niciunul. Diferența este exact costul rezoluției ARP. **Rețineți simptomul** Într-o rețea cu probleme, ping-uri care „merg, dar prima dată eșuează” indică aproape întotdeauna o chestiune de ARP, nu de rutare. Este primul lucru pe care îl eliminați din listă.

## 14.8 Administrarea switch-ului

Până acum am configurat switch-ul stând la consola lui. Ca să îl putem administra de la distanță, are nevoie de o adresă IP și de un mod de acces protejat prin parolă.

Comenzile de mai jos se scriu rând cu rând. Cele care încep cu **!** sunt comentarii - nu se scriu.



---

 Exemplu 14.1 configurația de management a lui SW1 copiază
 

---

```

1
2 Switch> enable
3 Switch# configure terminal
4 Switch(config)# hostname SW1
5 SW1(config)# no ip domain-lookup
6
7 ! adresa de management sta pe interfata virtuala VLAN 1
8 SW1(config)# interface vlan 1
9 SW1(config-if)# ip address 192.168.5.2 255.255.255.0
10 SW1(config-if)# no shutdown
11 SW1(config-if)# exit
12
13 ! parola pentru modul privilegiat
14 SW1(config)# enable secret Cisco123
15 SW1(config)# service password-encryption
16
17 SW1(config)# end
18 SW1# copy running-config startup-config
  
```

---

Verificați din două locuri:

---

 Exemplu 14.2 pe SW1 copiază
 

---

```

1
2 SW1# show ip interface brief
  
```

---

Vlan1 trebuie să apară cu 192.168.5.2 și up / up. Dacă apare administratively down, ați uitat no shutdown pe interfața VLAN.

---

 Exemplu 14.3 pe PC-A copiază
 

---

```

1
2 ping 192.168.5.2
  
```

---

### 14.8.1 Accesul de la distanță

Pentru o sesiune SSH adevărată, mai sunt necesare un nume de domeniu, o cheie și un utilizator. Aceste comenzi se demonstrează în Packet Tracer:

---

 Exemplu 14.4 pe SW1, în Packet Tracer copiază
 

---

```

1
2 SW1(config)# ip domain-name retele.local
3 SW1(config)# crypto key generate rsa
4 ! la intrebarea despre lungimea cheii , raspundeti 1024
  
```

```

5 SW1(config)# username admin privilege 15 secret Admin123
6 SW1(config)# line vty 0 4
7 SW1(config-line)# transport input ssh
8 SW1(config-line)# login local
9 SW1(config-line)# exit
10 SW1(config)# banner motd #Acces autorizat numai#

```

---

#### Exemplu 14.5 pe PC-A copiază

---

```

1
2 ssh -l admin 192.168.5.2

```

---

**Adresa de pe VLAN 1 nu face switch-ul un ruter.** Este doar un punct de contact pentru administrare. Switch-ul continuă să comute pe baza adreselor MAC și nu va ruta nimic între rețele. Dacă trebuie administrat dintr-o altă rețea, are nevoie în plus de `ip default-gateway`.

### 14.9 Port security

Un port de switch acceptă, implicit, orice adresă MAC i se prezintă. Port security limitează numărul de adrese care pot fi învățate pe un port și decide ce se întâmplă când limita e depășită - prima apărare împotriva atacului de tip CAM overflow din cursul 12.

#### Exemplu 14.6 pe portul lui PC-A copiază

---

```

1
2 SW1# configure terminal
3 SW1(config)# interface fastEthernet 0/1
4 SW1(config-if)# switchport mode access
5 SW1(config-if)# switchport port-security
6 SW1(config-if)# switchport port-security maximum 1
7 SW1(config-if)# switchport port-security mac-address sticky
8 SW1(config-if)# switchport port-security violation shutdown
9 SW1(config-if)# end
10
11 SW1# show port-security
12 SW1# show port-security interface fastEthernet 0/1
13 SW1# show running-config

```

---

Dați un ping de pe PC-A, apoi priviți din nou `show running-config`: adresa MAC a lui PC-A a fost „lipită” automat în configurație. Asta face `sticky`.

#### 14.9.1 Provocați violarea

În bancul din pagină, mutarea cablului se face din panoul **Cablurile**:

1. Alegeți PC-A și apăsați *scoate* pe portul Fa0.

2. Alegeți PC-C, apăsați *scoate*, apoi *leagă* și alegeți SW1 : Fa0/1.
3. Dați un ping de pe PC-C către PC-B.
4. Pe SW1: `show interfaces status` - portul apare **err-disabled**.

#### Exemplu 14.7 reactivarea portului copiază

```

1
2 SW1# configure terminal
3 SW1(config)# interface fastEthernet 0/1
4 SW1(config-if)# shutdown
5 SW1(config-if)# no shutdown
6 SW1(config-if)# end

```

#### Observație

**Portul se închide din nou imediat.** Cât timp PC-C stă pe acel port, violarea se repetă la primul cadru: adresa lipită în configurație este tot a lui PC-A. Portul rămâne utilizabil doar dacă puneți PC-A la loc sau dacă ștergeți adresa lipită. Este exact comportamentul din rețelele reale - și motivul pentru care un administrator obosit alege **restrict**.

Tabela 14.4 Provocați violarea.

| Acțiune         | Cadrela în ex-ces | Contor | Notificare    | Portul              |
|-----------------|-------------------|--------|---------------|---------------------|
| <b>protect</b>  | aruncate          | nu     | nu            | rămâne activ        |
| <b>restrict</b> | aruncate          | da     | SNMP + syslog | rămâne activ        |
| <b>shutdown</b> | -                 | da     | SNMP + syslog | <b>err-disabled</b> |

#### Observație

**Alegeți cu cap acțiunea.** **shutdown** pare cea mai sigură, dar înseamnă că orice utilizator care își schimbă laptopul rămâne fără rețea până când vine cineva să reactiveze portul. În majoritatea rețelelor de birou, **restrict** este compromisul corect: blochează traficul suspect, dar anunță în loc să pedepsească.

### 14.10 Sarcini de lucru

**Ce veți putea face după acest capitol.**

- Construiți topologia și verificați conectivitatea completă între toate cele șase stații
- Documentați, cu capturi de ecran, cele trei acțiuni ale switch-ului: unicast, flood, drop
- Justificați în scris numărul de domenii de coliziune și de broadcast din topologie
- Comparați timpii de răspuns prin hub și prin switch, în condiții de trafic simultan
- Configurați complet managementul lui SW1 și demonstrați o conexiune SSH reușită

- Configurați port security pe toate cele trei porturi de acces, cu **restrict**, și provocați o violare pe unul dintre ele
- Adăugați un al doilea switch legat de SW1 și explicați ce se întâmplă cu domeniile

### 14.11 Provocare

#### Exemplu

Considerați topologia din figura 1, cu tabela MAC a lui SW1 **complet goală**. Se execută, în ordine, următoarele trei operații:

PC-C trimite un cadru unicast către PC-A  
PC-D trimite un cadru unicast către PC-A  
PC-B trimite un cadru unicast către PC-D

Pentru fiecare operație, precizați: ce intrări se adaugă în tabela MAC a lui SW1, ce acțiune ia switch-ul, și care dintre cele șase stații primesc efectiv cadrul.

**Rezolvare. 1. PC-C → PC-A.** SW1 învață MAC-ul lui PC-C pe Fa0/3. Destinația e necunoscută, deci **flood** pe Fa0/1, Fa0/2, Fa0/4. Primesc cadrul: PC-A, PC-B și, prin hub, PC-D, PC-E, PC-F. Doar PC-A îl procesează; restul îl aruncă. **2. PC-D → PC-A.** Cadrul intră pe Fa0/4, deci SW1 învață MAC-ul lui PC-D pe **Fa0/4** - portul dinspre hub. Destinația PC-A este încă necunoscută (PC-A nu a transmis nimic până acum), deci din nou **flood**, pe Fa0/1, Fa0/2, Fa0/3. Primesc: PC-A, PC-B, PC-C. În plus, PC-E și PC-F primesc cadrul direct de la hub, care îl repetă pe toate porturile. **3. PC-B → PC-D.** SW1 învață MAC-ul lui PC-B pe Fa0/2. Destinația PC-D este cunoscută, pe Fa0/4, deci de această dată **unicast**: cadrul pleacă doar pe Fa0/4. Îl primesc însă PC-D, PC-E și PC-F - pentru că hub-ul nu știe să facă altceva decât să repete. **Concluzia care contează:** switch-ul a devenit selectiv după primul cadru al fiecărei stații. Hub-ul nu devine niciodată. Iar un atacator conectat la hub vede tot traficul, fără să facă absolut nimic.

### 14.12 Ce se predă

Tabela 14.5 Ce se predă.

| Livrabil                                                           | Format   | Punctaj |
|--------------------------------------------------------------------|----------|---------|
| Fișierul Packet Tracer, cu management și port security configurate | .pkt     | 35 %    |
| Documentarea celor trei acțiuni ale switch-ului, cu capturi        | document | 25 %    |
| Analiza domeniilor de coliziune și broadcast, justificată          | document | 20 %    |
| Rezolvarea provocării, cu raționamentul pas cu pas                 | document | 20 %    |

**Întrebări de verificare**

1. Dintr-un singur ping între două stații, câte intrări apar în tabela MAC a switch-ului?
  - a) Una, a sursei
  - b) Două: sursa și destinația, învățate din cererea și răspunsul ARP
  - c) Trei
  - d) Niciuna, până la expirarea aging-time
2. Câte domenii de coliziune are un switch cu 24 de porturi, toate ocupate?
  - a) 1
  - b) 12
  - c) 24
  - d) Depinde de VLAN-uri
3. Un switch primește un cadru a cărui adresă MAC destinație este în tabela, pe chiar portul de intrare. Ce face?
  - a) Îl inunda
  - b) Îl comută unicast pe acel port
  - c) Îl aruncă - sursa și destinația sunt de aceeași parte
  - d) Îl trimite la router
4. Două stații sunt legate la un hub, iar hub-ul la un switch. Cum apar ele în tabela MAC a switch-ului?
  - a) Fiecare pe portul ei
  - b) Amândouă pe același port, cel dinspre hub
  - c) Nu apar deloc
  - d) Doar prima care transmite
5. Ce face opțiunea mac-address sticky din port security?
  - a) Fixează portul în starea up
  - b) Învăță automat adresa MAC conectată și o scrie în configurație
  - c) Blochează toate adresele MAC
  - d) Criptează adresa MAC
6. De ce un hub face imposibilă comunicarea full-duplex?
  - a) Pentru că nu are suficiente porturi

- b) Pentru ca toate statiile impart acelasi mediu si trebuie sa asculte inainte de a transmite
- c) Pentru ca nu suporta cabluri Cat 5e
- d) Pentru ca nu are tabela MAC



# 15. Laborator 3: VLAN-uri, trunk-uri și rutare inter-VLAN

---

---

Problema pe care o rezolvă VLAN-urile

Echipamente necesare

Topologia și planul

Bancul de lucru

Pasul 1: crearea VLAN-urilor

Pasul 2: legăturile trunk

Pasul 3: router-on-a-stick

Varianta modernă: switch de nivel 3

Spanning Tree, observat

Patru defecte de diagnosticat

Sarcini de lucru

Provocare

Ce se predă

Întrebări de verificare

---

---

Un singur switch fizic devine trei rețele care nu se aud între ele, apoi două switch-uri își trimit tot traficul printr-un singur cablu, iar la final un ruter le leagă înapoi - controlat, prin subinterfețe. Lucrarea aceasta este momentul în care o rețea încetează să fie o grămadă de cabluri și devine un proiect.

## 15.1 Problema pe care o rezolvă VLAN-urile

O firmă are trei departamente răspândite pe două etaje. Fiecare etaj are câte un switch. Contabilitatea nu are voie să vadă traficul producției, iar vizitatorii nu au voie să vadă nimic.

Fără VLAN-uri, aveți exact două soluții și amândouă sunt proaste: fie cumpărați câte un switch separat pentru fiecare departament, la fiecare etaj - șase switch-uri în loc de două -, fie trageți cabluri dintr-un capăt în altul al clădirii. **VLAN-ul este al treilea răspuns:** același switch fizic se comportă ca mai multe switch-uri independente, iar apartenența se decide din configurație, nu din cablare.



**Observație**

**VLAN.** O rețea locală virtuală. Fiecare port al switch-ului este atribuit unui VLAN, iar switch-ul refuză să comute cadre între VLAN-uri diferite. Două calculatoare în porturi vecine, dar în VLAN-uri diferite, sunt la fel de izolate ca și cum ar fi în clădiri diferite.

**Observație**

**port de acces.** Un port care aparține unui singur VLAN și pe care circulă cadre *netichetate* - obișnuite. Aici se leagă calculatoarele, care nu știu și nu trebuie să știe nimic despre VLAN-uri.

**Observație**

**port trunk.** Un port care transportă cadre din mai multe VLAN-uri deodată, pe același cablu. Ca să nu se amestece, fiecare cadru primește o *etichetă* de 4 octeți cu numărul VLAN-ului - standardul 802.1Q. Aici se leagă switch-urile între ele și switch-ul de ruter.

**Observație**

**VLAN nativ.** Singurul VLAN care circulă *netichetat* pe un trunk. Există din motive istorice, este o sursă clasică de atacuri, și de aceea se alege deliberat un VLAN gol, în care nu se află nicio stație.

**Ce facem astăzi, în trei fraze.** Împărțim un switch în trei VLAN-uri și dovedim că izolarea funcționează. Legăm switch-urile prin trunk-uri și dovedim că același VLAN se aude peste două switch-uri, dar VLAN-urile diferite tot nu. La final punem un ruter deasupra, ca departamentele să poată vorbi între ele - dar numai prin el, unde putem controla ce trece.

## 15.2 Echipamente necesare

Pentru secțiunea opțională cu switch de nivel 3, mai aveți nevoie de un switch 3560 (tot din *Network Devices* → *Switches*), care îl înlocuiește pe SW-CORE. Pentru secțiunea de Spanning Tree, un al treilea cablu inversor, între SW-A și SW-B.

**Observație**

**Regula cablurilor, pe scurt.** Switch ↔ switch înseamnă *cross-over*. Switch ↔ ruter și stație ↔ switch înseamnă *straight-through*. În bancul de lucru din pagină cablurile sunt deja puse corect; în Packet Tracer le alegeți voi.

## 15.3 Topologia și planul

**Observați o simetrie utilă.** SW-A și SW-B au exact aceeași configurație. Le veți scrie identic, ceea ce înseamnă că o puteți copia - și, în practică, exact așa se lucrează: se scrie o dată, se verifică, apoi se aplică pe restul.

Tabela 15.1 Echipamente necesare.

| Buc. | Echipament     | Model                   | Categoria din Packet Tracer   | La ce folosește aici                          |
|------|----------------|-------------------------|-------------------------------|-----------------------------------------------|
| 3    | Switch         | 2960                    | Network Devices<br>→ Switches | SW-A (etajul 1), SW-B (etajul 2), SW-CORE     |
| 1    | Ruter          | 4331                    | Network Devices<br>→ Routers  | rutarea între VLAN-uri                        |
| 6    | Calculator     | PC-PT                   | End Devices → End Devices     | PC1 ... PC6, câte două de fiecare departament |
| 6    | Cablu drept    | Copper Straight-Through | Connections                   | stații → switch-uri, switch → ruter           |
| 2    | Cablu inversor | Copper Cross-Over       | Connections                   | SW-A ↔ SW-CORE, SW-B ↔ SW-CORE                |

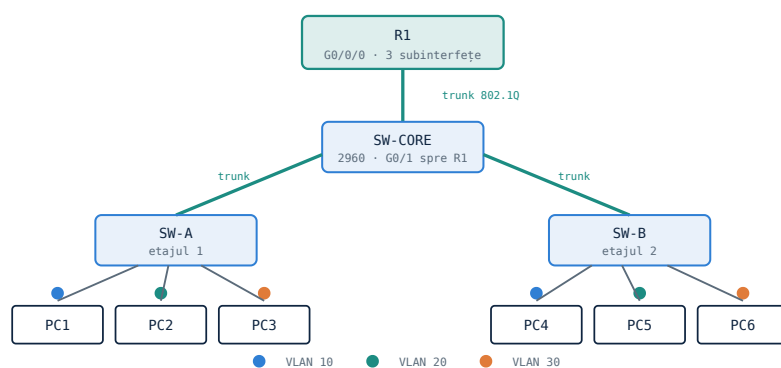


Figura 15.1 Trei departamente răspândite pe două etaje. Fiecare etaj are un switch, dar departamentele nu coincid cu etajele - exact situația pentru care au fost inventate VLAN-urile.

Tabela 15.2 Topologia și planul.

| VLAN | Nume      | Rețea           | Gateway      | Stații                              |
|------|-----------|-----------------|--------------|-------------------------------------|
| 10   | ADMIN     | 192.168.10.0/24 | 192.168.10.1 | PC1 (.11), PC4 (.14)                |
| 20   | PRODUCTIE | 192.168.20.0/24 | 192.168.20.1 | PC2 (.12), PC5 (.15)                |
| 30   | OASPETI   | 192.168.30.0/24 | 192.168.30.1 | PC3 (.13), PC6 (.16)                |
| 99   | NATIV     | -               | -            | niciuna - VLAN nativ, deliberat gol |
| 999  | NEFOLOSIT | -               | -            | porturile inactive                  |

Tabela 15.3 Topologia și planul.

| Switch  | Porturi de acces                                  | Porturi trunk                                |
|---------|---------------------------------------------------|----------------------------------------------|
| SW-A    | Fa0/1 → VLAN 10, Fa0/2 → VLAN 20, Fa0/3 → VLAN 30 | G0/1 spre SW-CORE                            |
| SW-B    | Fa0/1 → VLAN 10, Fa0/2 → VLAN 20, Fa0/3 → VLAN 30 | G0/1 spre SW-CORE                            |
| SW-CORE | niciunul                                          | G0/1 spre R1, G0/2 spre SW-A, G0/3 spre SW-B |

## 15.4 Bancul de lucru

Topologia din figura 1, cablată, cu adresele stațiilor deja puse. Restul îl configurați voi, urmând pașii din secțiunile 5–7. Obiectivele din dreapta se bifează singure.

## 15.5 Pasul 1: crearea VLAN-urilor

1. **Verificați punctul de plecare** Înainte de orice configurare, dați de pe PC1 un ping către PC4: pe PC1 copiază ping 192.168.10.14 **Funcționează**. Toate cele șase stații sunt, deocamdată, într-o singură rețea de nivel 2 - VLAN-ul 1, cel implicit. Rețineți rezultatul: peste cinci minute nu va mai merge, și acesta este scopul.
2. **Creați VLAN-urile - pe toate cele trei switch-uri** identic pe SW-A, SW-B și SW-CORE copiază Switch> enable Switch# configure terminal Switch(config)# vlan 10 Switch(config-vlan)# name ADMIN Switch(config-vlan)# vlan 20 Switch(config-vlan)# name PRODUCTIE Switch(config-vlan)# vlan 30 Switch(config-vlan)# name OASPETI Switch(config-vlan)# vlan 99 Switch(config-vlan)# name NATIV Switch(config-vlan)# vlan 999 Switch(config-vlan)# name NEFOLOSIT Switch(config-vlan)# exit Nu este nevoie să ieșiți din modul `config-vlan` între ele: comanda `vlan 20` funcționează direct din interiorul VLAN-ului 10. **Inclusiv SW-CORE, care nu are nicio stație** Un switch nu poate comuta traficul unui VLAN pe care nu îl are configurat local, chiar dacă doar îl tranzitează. *Un VLAN lipsă pe un switch intermediar este cauza numărul unu a problemelor din această lucrare* - și una dintre cele mai frecvente în rețelele reale.
3. **Atribuiți porturile de acces** pe SW-A - identic pe SW-B copiază SW-A(config)# interface fastEthernet 0/1 SW-A(config-if)# switchport mode access SW-A(config-if)# switchport access vlan 10 SW-A(config-if)# switchport nonegotiate SW-A(config-if)# spanning-tree portfast SW-A(config-if)# exit SW-A(config)# interface fastEthernet 0/2 SW-A(config-if)# switchport mode access SW-A(config-if)# switchport access vlan 20 SW-A(config-if)# switchport nonegotiate SW-A(config-if)# exit SW-A(config)# interface fastEthernet 0/3 SW-A(config-if)# switchport mode access SW-A(config-if)# switchport access vlan 30 SW-A(config-if)# switchport nonegotiate SW-A(config-if)# end Ce face fiecare linie: Comanda `Efectul switchport mode access` declară

portul ca port de utilizator, nu de legătură între switch-uri `switchport access vlan 10` îl mută în VLAN-ul 10 `switchport nonegotiate` oprește DTP, deci portul nu poate fi păcălit să devină trunk (atacul de *switch spoofing* din cursul 12) `spanning-tree portfast` portul intră imediat în funcțiune, fără cele 30 de secunde de așteptare STP

4. **Verificați izolarea - aici se vede rezultatul** pe SW-A copiază SW-A# `show vlan brief` Trebuie să vedeți exact asta: ce trebuie să apară VLAN Name Status Ports ----  
-----  
1 default active G0/1, G0/2  
10 ADMIN active Fa0/1 20 PRODUCTIE active Fa0/2 30 OASPETI active Fa0/3 99 NATIV active 999 NEFOLOSIT active Acum testați, de pe PC1: pe PC1 copiază `ping 192.168.20.12 ping 192.168.10.14` **Amândouă trebuie să eșueze.** Primul, pentru că PC2 este în alt VLAN. Al doilea, pentru că PC4 este pe alt switch și trunk-ul nu există încă. Eșecul este dovada că ați configurat corect. Întrebare PC1 și PC2 sunt conectate la același switch fizic, la două porturi vecine. De ce nu se aud? Vezi răspunsul Pentru că switch-ul refuză să comute între porturi din VLAN-uri diferite. Chiar dacă ar avea adresa MAC a lui PC2 în tabelă, nu ar folosi-o: tabela MAC este, de fapt, indexată și după VLAN. Cele două stații se află în domenii de broadcast distincte, iar din perspectiva nivelului 2 e ca și cum ar fi conectate la switch-uri complet separate. În plus, au și adrese din rețele IP diferite, deci nici nu ar încerca ARP direct - ar trimite cadrul spre gateway, care încă nu există.

## 15.6 Pasul 2: legăturile trunk

1. **Configurați trunk-urile** Trei switch-uri, patru porturi de legătură. Toate primesc aceeași configurație: pe SW-A, portul spre SW-CORE copiază SW-A(config)# `interface gigabitEthernet 0/1 SW-A(config-if)# switchport mode trunk SW-A(config-if)# switchport trunk native vlan 99 SW-A(config-if)# switchport trunk allowed vlan 10,20,30,99 SW-A(config-if)# switchport nonegotiate SW-A(config-if)# end` Repetați identic pe: SW-B, portul G0/1 SW-CORE, portul G0/2 (spre SW-A) SW-CORE, portul G0/3 (spre SW-B) SW-CORE, portul G0/1 (spre R1) - da, și acesta: routerul primește cadre etichetate **Cele două setări de disciplină `switchport trunk allowed vlan`**: lăsată implicit, o legătură trunk transportă *toate* cele 4094 de VLAN-uri posibile. Declarând explicit lista, reduceți traficul inutil și limitați ce poate injecta un atacator.  
`switchport trunk native vlan 99`: VLAN-ul nativ trebuie să fie unul gol. Dacă ar rămâne VLAN 1, o stație din VLAN 1 ar putea executa atacul de *double tagging* din cursul 12.
2. **Verificați** pe SW-A copiază SW-A# `show interfaces trunk` ce trebuie să apară Port Mode Encapsulation Status Native vlan G0/1 on 802.1q trunking 99 Port Vlans allowed on trunk G0/1 10,20,30,99 Dacă *Status* arată **not-trunking**, portul nu are legătură activă: verificați cablul sau capătul celălalt. Dacă *Native vlan* diferă între cele două

capete ale aceleiași legături, IOS afișează un avertisment în consolă - nu îl ignorați, este exact vulnerabilitatea pe care încercați să o închideți.

3. **Testați din nou izolarea** pe PC1copiază ping 192.168.10.14 ping 192.168.20.15 TestRezultat așteptatDe ce PC1 → PC4mergeaceleși VLAN 10, deși pe switch-uri diferite: cadrul a traversat trunk-ul, etichetat PC1 → PC5nu mergetrunk-ul transportă ambele VLAN-uri, dar nu le amestecă Aceasta este ideea centrală a lucrării, într-o singură propoziție: **un trunk transportă mai multe VLAN-uri fără să le unească.**
4. **Priviți eticheta 802.1Q** În bancul din pagină: dați ping 192.168.10.14 de pe PC1 și priviți tabelul *Traseul ultimului pachet*. Coloana lipsă de pe portul de acces și prezentă pe trunk este chiar eticheta - segmentele dintre switch-uri poartă VLAN 10, cele dinspre stații nu poartă nimic. În Packet Tracer: în *Simulation*, filtrat pe ICMP și ARP, faceți clic pe plicul de pe legătura trunk și deschideți *Outbound PDU Details*. Veți vedea, în structura cadrului Ethernet, un câmp suplimentar de 4 octeți, cu TPID 0x8100 și VLAN ID 10. Comparați cu același cadru pe un port de acces: acolo eticheta nu există. Aceasta este, la propriu, întreaga diferență dintre un port de acces și un port trunk.

### 15.7 Pasul 3: router-on-a-stick

Cele trei VLAN-uri sunt acum trei rețele complet separate. Departamentul ADMIN trebuie totuși să ajungă la resursele din PRODUCTIE. Ne trebuie un dispozitiv de nivel 3 - dar ruterul are o singură interfață liberă, iar noi avem trei rețele.

Soluția: **subinterfețele**. Aceeași interfață fizică se împarte logic în trei, fiecare cu adresa ei și cu eticheta ei de VLAN.

1. **Configurați subinterfețele pe R1** pe R1copiază Router> enable Router# configure terminal Router(config)# hostname R1 R1(config)# interface gigabitEthernet 0/0/0 R1(config-if)# no ip address R1(config-if)# no shutdown R1(config-if)# exit R1(config)# interface gigabitEthernet 0/0/0.10 R1(config-subif)# description Gateway VLAN 10 ADMIN R1(config-subif)# encapsulation dot1Q 10 R1(config-subif)# ip address 192.168.10.1 255.255.255.0 R1(config-subif)# exit R1(config)# interface gigabitEthernet 0/0/0.20 R1(config-subif)# encapsulation dot1Q 20 R1(config-subif)# ip address 192.168.20.1 255.255.255.0 R1(config-subif)# exit R1(config)# interface gigabitEthernet 0/0/0.30 R1(config-subif)# encapsulation dot1Q 30 R1(config-subif)# ip address 192.168.30.1 255.255.255.0 R1(config-subif)# exit R1(config)# interface gigabitEthernet 0/0/0.99 R1(config-subif)# encapsulation dot1Q 99 native R1(config-subif)# end R1# copy running-config startup-config **Trei greșeli care se fac de fiecare dată** 1. **no shutdown** se dă pe interfața *fizică*. Dacă ea rămâne închisă, toate subinterfețele sunt moarte, oricât de corect ar fi configurate. Încercați: dați **shutdown** pe G0/0/0 și priviți cum se sting toate obiectivele deodată.
2. Interfața fizică nu primește adresă IP - adresele stau pe subinterfețe.

3. Numărul subinterfeței (.10) nu are nicio semnificație pentru echipament; ceea ce contează este **encapsulation dot1q 10**. Folosiți totuși același număr - vă va salva o oră de depanare peste șase luni.
2. **Verificați rutarea completă** pe R1 copiază R1# show ip interface brief R1# show ip route În tabela de rutare trebuie să apară **exact trei rute conectate**, marcate **C**, câte una pentru fiecare VLAN. Dacă apar doar două, una dintre subinterfețe nu are adresă sau are eticheta greșită. pe PC1 copiază ping 192.168.10.1 ping 192.168.20.12 ping 192.168.30.13 tracert 192.168.30.16 **tracert** va arăta exact un salt intermediar: gateway-ul. Traficul dintre două VLAN-uri urcă până la ruter și coboară înapoi pe același fir - de aici și denumirea de „router on a stick”.
3. **Măsurați gâtuirea** Tot traficul dintre toate VLAN-urile trece de două ori prin același cablu G0/0/0: o dată în sus, o dată în jos. Într-o rețea reală, aceasta este limita arhitecturii - și motivul pentru care secțiunea următoare există. În Packet Tracer, rulați simultan de pe PC1 și PC4 ping-uri masive (**ping -n 200 -l 1400**) către stații din alt VLAN și comparați timpii cu cei obținuți când rulați singuri.

### 15.8 Varianta modernă: switch de nivel 3

Înlocuiți SW-CORE cu un switch **3560** și eliminați complet ruterul. Rutarea se face acum în hardware, pe backplane-ul switch-ului, fără niciun cablu implicat. În bancul din pagină, aceeași demonstrație se face activând rutarea chiar pe SW-CORE.

---

#### Exemplu 15.1 pe SW-CORE copiază

---

```

1
2 SW-CORE(config)# ip routing
3
4 SW-CORE(config)# interface vlan 10
5 SW-CORE(config-if)# ip address 192.168.10.1 255.255.255.0
6 SW-CORE(config-if)# no shutdown
7 SW-CORE(config-if)# exit
8
9 SW-CORE(config)# interface vlan 20
10 SW-CORE(config-if)# ip address 192.168.20.1 255.255.255.0
11 SW-CORE(config-if)# no shutdown
12 SW-CORE(config-if)# exit
13
14 SW-CORE(config)# interface vlan 30
15 SW-CORE(config-if)# ip address 192.168.30.1 255.255.255.0
16 SW-CORE(config-if)# no shutdown
17 SW-CORE(config-if)# end
18

```

---

```
19 SW-CORE# show ip route
```

---

### Observație

**Nu le faceți pe amândouă deodată.** Dacă lăsați și ruterul, și SVI-urile configurate, cele trei gateway-uri vor exista în două locuri - adrese IP duplicate. În bancul din pagină, scoateți întâi cablul lui R1 din panoul *Cablurile*, sau dați **shutdown** pe interfața lui fizică.

**Router-on-a-stick.** Funcționează cu orice ruter, chiar și vechi Ruterul poate face și NAT, VPN, ACL-uri complexe Configurare simplă, ușor de explicat

### Switch de nivel 3 cu SVI

- Rutare în hardware, fără gătuire
- Nu consumă niciun cablu și niciun port
- Echipament mai scump; funcții de securitate mai limitate decât la un ruter

Testați aceleași ping-uri ca înainte. Rezultatul funcțional este identic, dar traficul nu mai iese niciodată din switch - verificați cu **tracert**, unde saltul intermediar rămâne unul singur, dar este chiar switch-ul.

## 15.9 Spanning Tree, observat

Adăugați o legătură directă între SW-A și SW-B, configurată tot ca trunk. În bancul din pagină: alegeți SW-A, din panoul *Cablurile* apăsați *leagă* pe portul G0/2 și alegeți SW-B : G0/2. Ați creat o buclă fizică între cele trei switch-uri - exact situația din cursul 4.

### Exemplu 15.2 observarea arborelui copiază

---

```
1
2 SW-A# show spanning-tree
3 SW-CORE# show spanning-tree
```

---

1. Identificați care switch a devenit **root bridge** - apare cu mențiunea **This bridge is the root**. Fără nicio configurare, decide adresa MAC cea mai mică, deci de regulă cel mai vechi echipament din rețea. Rareori cel pe care l-ați fi ales voi.
2. Găsiți portul aflat în starea **BLK**, cu rolul **Altn**. Acela este cablul pe care STP l-a scos din uz ca să rupă bucla. Legătura există fizic, dar nu transportă nimic.
3. Verificați că ping-ul dintre PC1 și PC4 funcționează în continuare: bucla a fost neutralizată fără să se piardă conectivitatea.
4. Forțați rădăcina pe switch-ul din centru, unde vă convine să fie:

### Exemplu 15.3 rădăcina, stabilită explicit copiază

---

```
1
2 SW-CORE(config)# spanning-tree vlan 10,20,30,99 root primary
3 SW-A(config)# spanning-tree vlan 10,20,30,99 root secondary
```

---

Rulați din nou `show spanning-tree` pe SW-CORE: acum el este rădăcina, iar portul blocat s-a mutat. Comanda `root primary` nu face nimic magic - pune prioritatea la 24576, o valoare mai mică decât cea implicită (32768), iar prioritatea mai mică câștigă alegerea.

### Exemplu

Deconectați cablul dintre SW-CORE și SW-A. Cronometrați cât durează până când ping-ul dintre PC1 și PC4 începe să funcționeze din nou, prin legătura de rezervă.

**Rezolvare.** Aproximativ 30–50 de secunde cu STP clasic: max age (20 s) plus tranzițiile prin listening și learning ( $2 \times 15$  s). Activați apoi modul rapid și repetați măsurătoarea: pe toate switch-urile `Switch(config)# spanning-tree mode rapid-pvst`. Convergența scade la câteva secunde. Este exact motivul pentru care nicio rețea modernă nu mai rulează 802.1D clasic.

## 15.10 Patru defecte de diagnosticat

Provocați-le pe rând, în bancul de lucru. De fiecare dată notați **ce comandă v-a arătat problema** - asta rămâne util după laborator.

Tabela 15.4 Patru defecte de diagnosticat.

| Defect                             | Cum îl produceți                                              | Simptom                                         | Comanda care îl scoate la iveală              |
|------------------------------------|---------------------------------------------------------------|-------------------------------------------------|-----------------------------------------------|
| VLAN lipsă pe switch-ul de tranzit | pe SW-CORE: no<br>vlan 20                                     | VLAN 10 și 30 merg peste trunk, VLAN 20 nu      | <code>show vlan brief</code><br>pe SW-CORE    |
| VLAN scos din lista permisă        | pe SW-A G0/1:<br>switchport trunk<br>allowed vlan<br>10,30,99 | PC2 nu mai ajunge nicăieri în afara switch-ului | <code>show interfaces<br/>trunk</code>        |
| Port lăsat în VLAN 1               | pe SW-B: no<br>switchport access<br>vlan 10 pe Fa0/1          | PC4 dispăre din rețeaua lui, dar cablul e verde | <code>show vlan brief</code><br>pe SW-B       |
| Etichetă greșită pe subinterfață   | pe R1:<br>encapsulation<br>dot1Q 25 în loc de<br>20           | VLAN 20 nu are gateway; celelalte funcționează  | <code>show<br/>running-config</code><br>pe R1 |

**Tiparul comun.** Toate cele patru defecte produc același simptom general - „un VLAN nu merge, celelalte da” - și niciunul nu se vede în starea cablurilor. Când auziți acest simptom, ordinea de verificare este întotdeauna aceeași: VLAN-ul există pe *toate* switch-urile de pe drum? Portul de acces e în VLAN-ul corect? VLAN-ul e permis pe *fiecare* trunk? Subinterfața ruterului are eticheta corectă?

## 15.11 Sarcini de lucru

**Ce veți putea face după acest capitol.**

- Construiți topologia și configurați cele cinci VLAN-uri pe toate switch-urile



- Demonstrați izolarea între VLAN-uri înainte de configurarea ruterului
- Configurați trunk-urile cu VLAN nativ 99 și listă explicită de VLAN-uri permise
- Capturați și documentați o etichetă 802.1Q din modul Simulation
- Configurați router-on-a-stick și demonstrați conectivitate completă între toate cele trei VLAN-uri
- Realizați aceeași funcție cu un switch 3560 și SVI-uri; comparați traseul pachetului în ambele variante
- Adăugați bucla, observați STP și forțați rădăcina pe SW-CORE
- Provocați cele patru defecte din secțiunea 10 și notați, pentru fiecare, comanda de diagnostic

### 15.12 Provocare

#### Exemplu

VLAN-ul 30 este pentru vizitatori. Politica firmei cere: **vizitatorii au acces la Internet, dar nu la nicio resursă internă**. În laboratorul acesta încă nu aveți Internet, așa că simulați-l cu un server plasat într-un VLAN 40, 192.168.40.0/24.

Cerințe:

PC3 și PC6 (VLAN 30) trebuie să ajungă la serverul din VLAN 40 PC3 și PC6 **nu** trebuie să ajungă la nicio stație din VLAN 10 sau 20 Stațiile din VLAN 10 și 20 trebuie să ajungă peste tot, inclusiv la stațiile din VLAN 30

Punctul 3 este partea interesantă: traficul de la ADMIN spre OASPETI trebuie să treacă, dar cel invers nu. Gândiți-vă ce fel de listă de acces vă trebuie și pe ce direcție se aplică. Formulați în cuvinte regula exactă și motivați plasarea ei - configurarea propriu-zisă o veți face în laboratorul 6.

**Rezolvare.** O listă **standard** nu ajunge: ea filtrează doar după sursă, iar sursa este aceeași (VLAN 30) și în traficul permis (spre VLAN 40) și în cel interzis (spre VLAN 10 și 20). Vă trebuie o listă **extinsă**, care să vadă și destinația. Regula ar suna astfel: „respinge traficul IP cu sursa în 192.168.30.0/24 și destinația în 192.168.10.0/24 sau 192.168.20.0/24; permite restul”. Plasarea corectă este pe subinterfața G0/0/0.30, direcția **in** - adică cât mai aproape de sursă, ca traficul interzis să nici nu ajungă la ruter. Punctul 3 funcționează automat cu această plasare: traficul ADMIN → OASPETI intră pe G0/0/0.10, unde lista nu este aplicată, deci nu este filtrat.

### 15.13 Ce se predă

#### Întrebări de verificare

1. Doua statii sunt in porturi vecine ale aceluiasi switch, dar in VLAN-uri diferite. Se aud?
  - a) Da, sunt pe acelasi switch
  - b) Nu: switch-ul nu comuta intre VLAN-uri, tabela MAC este indexata si dupa VLAN

Tabela 15.5 Ce se predă.

| Livrabil                                                                          | Format   | Punctaj |
|-----------------------------------------------------------------------------------|----------|---------|
| Fișierul Packet Tracer cu VLAN-uri, trunk-uri și rutare inter-VLAN funcționale    | .pkt     | 35 %    |
| Captura etichetei 802.1Q, comentată                                               | document | 15 %    |
| Cele patru defecte, cu simptom și comandă de diagnostic                           | document | 20 %    |
| Comparația router-on-a-stick / switch de nivel 3, cu traseul pachetului în ambele | document | 15 %    |
| Provocarea: regula formulată în cuvinte, cu plasarea justificată                  | document | 15 %    |

- c) Da, daca au aceeași mască
  - d) Doar prin broadcast
2. Ce transportă un port trunk?
- a) Un singur VLAN, mai repede
  - b) Cadre din mai multe VLAN-uri, fiecare cu o etichetă 802.1Q de 4 octeți
  - c) Doar trafic de management
  - d) Doar cadre netichetate
3. De ce se alege un VLAN nativ gol, de exemplu 99?
- a) Ca să fie mai rapid
  - b) Pentru ca traficul VLAN-ului nativ circula netichetat; dacă ar fi VLAN-ul stațiilor, ar fi posibil atacul de double tagging
  - c) Pentru ca IOS o cere
  - d) Ca să încapă mai multe VLAN-uri
4. Ați configurat corect trei subinterfețe pe router, dar nimic nu funcționează. Ce verificați întâi?
- a) Adresele IP ale stațiilor
  - b) Dacă interfața fizică are no shutdown
  - c) Mască de rețea
  - d) Cablul de consolă

5. Un VLAN merge între stațiile de pe același switch, dar nu și spre celălalt switch. Unde cautați?
- a) La adresele IP
  - b) La lista switchport trunk allowed vlan și la existența VLAN-ului pe switch-ul de tranzit
  - c) La masca de subrețea
  - d) La viteza portului
6. Ce face comanda spanning-tree vlan 10 root primary?
- a) Blochează un port
  - b) Pune prioritatea switch-ului la 24576, deci îl face să castige alegerea rădăcinii
  - c) Dezactivează STP
  - d) Face portul să treacă direct în forwarding

## 16. Laborator 4: Proiectarea adreselor: VLSM și DHCP

---

---

Ce înseamnă „a proiecta adresele”

Echipamente necesare

Tema de proiectare

Cum se calculează, pas cu pas

Topologia de implementat

Bancul de lucru

Implementarea planului

DHCP pe ruter

DHCP relay

Patru probleme de diagnosticat

Sarcini de lucru

Provocare

Ce se predă

Întrebări de verificare

---

---

Până acum adresele v-au fost date. În lucrarea aceasta le proiectați singuri: primiți un singur bloc de adrese și o listă de cerințe reale, iar la final trebuie să iasă o schemă în care nimic nu se suprapune, nimic nu se risipește și mai rămâne loc pentru creștere. Apoi lăsați DHCP-ul să o distribuie.

### 16.1 Ce înseamnă „a proiecta adresele”

Un furnizor vă dă un bloc de adrese - un singur bloc. Din el trebuie să iasă toate rețelele firmei: producția, birourile, serverele, legăturile dintre rutere. Fiecare bucată trebuie să fie destul de mare cât să încapă stațiile, dar nu mai mare decât e nevoie, pentru că restul rămâne pentru mai târziu.

#### Observație

**VLSM.** *Variable Length Subnet Mask* - măști de lungime variabilă. Ideea: nu toate subrețelele trebuie să aibă aceeași dimensiune. O legătură între două rutere are nevoie de exact două adrese; un segment de producție are nevoie de cinci sute. Cu VLSM le puteți tăia pe fiecare pe măsură.

**Observație**

**prefix.** Numărul de biți care descriu rețeaua, scris după bară: /24 înseamnă 24 de biți de rețea și 8 de host. Cu 8 biți de host aveți  $2^8 = 256$  de adrese, din care 254 utilizabile - se scad adresa de rețea și cea de broadcast.

**Observație**

**DHCP.** Protocolul prin care o stație primește automat adresă, mască, gateway și DNS. Fără el, cineva ar trebui să configureze manual fiecare calculator din firmă - și să țină minte ce a dat fiecăruia.

**Regula de aur a acestei lucrări.** Planul se face *întâi pe hârtie*, complet, și abia apoi se atinge un echipament. Un plan greșit descoperit după ce ați configurat trei rutere înseamnă că reconfigurați trei rutere. Un plan greșit descoperit pe hârtie înseamnă că ștergeți un rând.

## 16.2 Echipamente necesare

Tabela 16.1 Echipamente necesare.

| Buc.        | Echipament   | Model                      | Categoria din Packet Tracer   | La ce folosește aici                     |
|-------------|--------------|----------------------------|-------------------------------|------------------------------------------|
| 3           | Ruter        | 4331                       | Network Devices<br>→ Routers  | R1 (sediu), R2 (producție), R3 (depozit) |
| 6           | Switch       | 2960                       | Network Devices<br>→ Switches | câte unul pe segment de utilizatori      |
| 2 / segment | Calculator   | PC-PT                      | End Devices                   | una statică, una prin DHCP               |
| 1           | Server       | Server-PT                  | End Devices                   | în segmentul Servere                     |
| 3           | Cablu serial | Serial DCE                 | Connections                   | legăturile punct-la-punct dintre rutere  |
| după caz    | Cablu drept  | Copper<br>Straight-Through | Connections                   | stații → switch-uri, switch-uri → rutere |

**Observație**

**Cablul serial are un capăt „ceas”.** Pe o legătură serială, unul dintre capete este **DCE** și trebuie să furnizeze ceasul. În Packet Tracer, capătul DCE are o mică pictogramă de ceas pe cablu, iar acolo se dă comanda `clock rate 128000`. Dacă interfața rămâne `up / down`, aproape sigur ați uitat această comandă sau ați pus-o pe capătul greșit. Pe legăturile Ethernet comanda nu există și nu e necesară.

**Simplificare pentru laborator.** Nu veți conecta 483 de calculatoare. Puneți câte

două stații per segment: una configurată static, una prin DHCP. Planul de adresare trebuie proiectat pentru cifrele reale, dar verificat cu două stații.

### 16.3 Tema de proiectare

#### Observație

**Situația.** Firma pentru care lucrați a primit de la furnizorul de Internet blocul **172.20.16.0/21**. Trebuie să acopere structura actuală și să lase loc de creștere pentru cel puțin trei ani.

Tabela 16.2 Tema de proiectare.

| Segment                      | Stații acum | Creștere estimată | Observații                |
|------------------------------|-------------|-------------------|---------------------------|
| Producție                    | 420         | +15 %             | cel mai mare segment      |
| Birouri                      | 180         | +25 %             | creștere rapidă           |
| Depozit                      | 90          | +10 %             | terminale mobile          |
| Oaspeți                      | 50          | -                 | izolat de rest            |
| Servere                      | 25          | +50 %             | adrese statice            |
| Management                   | 12          | +100 %            | doar echipamente de rețea |
| Legături R1–R2, R2–R3, R1–R3 | 2 fiecare   | -                 | punct-la-punct            |

### 16.4 Cum se calculează, pas cu pas

Metoda are patru pași și nu se sare peste niciunul.

1. **Adăugați creșterea** la fiecare cerință și rotunjiți în sus. Producție:  $420 \times 1,15 = 483$ .
2. **Calculați biții de host** necesari, din inegalitatea  $2^n - 2 \geq$  cerința. Pentru 483 de stații,  $n = 9$  ( $2^9 - 2 = 510$  utilizabile), deci prefixul este  $32 - 9 = /23$ .
3. **Ordonăți descrescător** după dimensiune și alocați în această ordine, începând de la adresa de start a blocului.
4. **Verificați** că totalul încapă și notați ce a rămas liber.

**De ce ordinea descrescătoare nu este opțională.** Un bloc de dimensiune 512 trebuie să înceapă la o adresă multiplu de 512 - asta înseamnă „alinieră”. Dacă alocați întâi câteva blocuri mici, următoarea adresă liberă nu mai este aliniată și trebuie să săriți până la următorul multiplu, pierzând tot ce e între. Alocând de la mare la mic, alinierea se păstrează singură.

Tabela 16.3 Un exemplu complet, pentru primele două.

| Pas                      | Producție (483)                            | Birouri (225)                              |
|--------------------------|--------------------------------------------|--------------------------------------------|
| biți de host             | $2^9 - 2 = 510 \geq 483 \rightarrow n = 9$ | $2^8 - 2 = 254 \geq 225 \rightarrow n = 8$ |
| prefix                   | $32 - 9 = /23$                             | $32 - 8 = /24$                             |
| mască                    | 255.255.254.0                              | 255.255.255.0                              |
| subrețea                 | 172.20.16.0/23                             | 172.20.18.0/24                             |
| interval utilizabil      | 172.20.16.1 – 172.20.17.254                | 172.20.18.1 – 172.20.18.254                |
| broadcast                | 172.20.17.255                              | 172.20.18.255                              |
| următoarea adresă liberă | 172.20.18.0                                | 172.20.19.0                                |

### 16.4.1 Un exemplu complet, pentru primele două

Continuați singuri cu Depozit, Oaspeți, Servere, Management și cele trei legături punct-la-punct. Apoi verificați:

| BLOC DE ADRESE |                  | STAȚII NECESARE (SEPARATE PRIN VIRGULĂ) |               |        |
|----------------|------------------|-----------------------------------------|---------------|--------|
| 172.20.16.0/21 |                  | 483,225,99,50,38,24,2,2,2               |               |        |
| CERINȚĂ        | SUBREȚEA         | INTERVAL UTILIZABIL                     | BROADCAST     | RISIPĂ |
| 483 stații     | 172.20.16.0/23   | 172.20.16.1 – 172.20.17.254             | 172.20.17.255 | 27     |
| 225 stații     | 172.20.18.0/24   | 172.20.18.1 – 172.20.18.254             | 172.20.18.255 | 29     |
| 99 stații      | 172.20.19.0/25   | 172.20.19.1 – 172.20.19.126             | 172.20.19.127 | 27     |
| 50 stații      | 172.20.19.128/26 | 172.20.19.129 – 172.20.19.190           | 172.20.19.191 | 12     |
| 38 stații      | 172.20.19.192/26 | 172.20.19.193 – 172.20.19.254           | 172.20.19.255 | 24     |
| 24 stații      | 172.20.20.0/27   | 172.20.20.1 – 172.20.20.30              | 172.20.20.31  | 6      |
| 2 stații       | 172.20.20.32/30  | 172.20.20.33 – 172.20.20.34             | 172.20.20.35  | 0      |
| 2 stații       | 172.20.20.36/30  | 172.20.20.37 – 172.20.20.38             | 172.20.20.39  | 0      |
| 2 stații       | 172.20.20.40/30  | 172.20.20.41 – 172.20.20.42             | 172.20.20.43  | 0      |

spațiu consumat **1068** / 2048 adrese  
 adrese de stație risipite **125**  
 rămâne liber de la **172.20.20.44**

Alocarea se face de la cea mai mare cerință spre cea mai mică – singura ordine care nu lasă găuri în spațiul de adrese.

Figura 16.1 Verificați-vă planul aici.

Faceți întâi calculul pe hârtie, apoi comparați cu widgetul. Dacă rezultatele diferă, găsiți unde ați greșit - de obicei la rotunjire sau la ordinea alocării. Widgetul este pentru verificare, nu pentru a înlocui exercițiul: la examen nu îl aveți.

#### Exemplu

De ce un segment cu 30 de stații are nevoie de /26 și nu de /27?

**Rezolvare.** Un /27 oferă  $2^5 = 32$  de adrese, din care se scad adresa de rețea și cea de

broadcast: rămân **30 utilizabile**. Deci /27 ar fi exact suficient - dar numai dacă nu vă trebuie **niciun** loc în plus, nici măcar pentru gateway. Atenție: gateway-ul consumă și el o adresă utilizabilă. Deci pentru 30 de **stații** plus gateway aveți nevoie de 31 de adrese utilizabile - și /27 nu mai ajunge. Este greșeala clasică de la examen.

### 16.4.2 Verificarea fiecărei subrețele

ADRESĂ IP      PREFIX /

172.20.16.1      23

10101100.00010100.00010000.00000001

maskă 255.255.254.0 (/23)

adresă de rețea 172.20.16.0

adresă de broadcast 172.20.17.255

interval utilizabil 172.20.16.1 – 172.20.17.254

adrese totale / utilizabile 512 / 510

wildcard 0.0.1.255

■ biți de rețea ■ biți de host

Figura 16.2 Calculator de verificare.

Introduceți pe rând fiecare subrețea din planul vostru și confirmați intervalul utilizabil și adresa de broadcast. Verificați în special că **broadcast-ul unei subrețele este cu exact 1 mai mic decât adresa de rețea a următoarei** - dacă nu, ori ați lăsat o gaură, ori subrețelele se suprapun.

## 16.5 Topologia de implementat

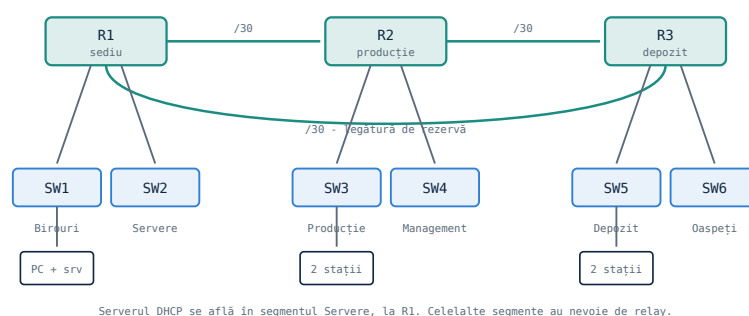


Figura 16.3 Trei rutere, șase segmente de utilizatori și trei legături punct-la-punct. Este topologia pe care veți lucra și în laboratorul 5.

### 16.5.1 Tabelul de adresare - se completează înainte de orice comandă

Primele două rânduri sunt completate ca model. Convenția: **gateway-ul primește prima adresă utilizabilă** din fiecare subrețea. Este o convenție, nu o regulă - dar dacă o respectați consecvent, oricine vă poate citi rețeaua fără să întrebe.



Tabela 16.4 Tabelul de adresare - se completează înainte de orice comandă.

| Segment    | Subrețea       | Mască         | Gateway     | Interval utilizabil | Broadcast     |
|------------|----------------|---------------|-------------|---------------------|---------------|
| Producție  | 172.20.16.0/23 | 255.255.254.0 | 172.20.16.1 | .16.1 – .17.254     | 172.20.17.255 |
| Birouri    | 172.20.18.0/24 | 255.255.255.0 | 172.20.18.1 | .18.1 – .18.254     | 172.20.18.255 |
| Depozit    |                |               |             |                     |               |
| Oaspeți    |                |               |             |                     |               |
| Servere    |                |               |             |                     |               |
| Management |                |               |             |                     |               |
| R1-R2      |                | /30           | -           |                     |               |
| R2-R3      |                | /30           | -           |                     |               |
| R1-R3      |                | /30           | -           |                     |               |

## 16.6 Bancul de lucru

O parte din topologie - R1 cu Birouri și Servere, R2 cu Producție, legate printr-o serială. Este suficientă ca să implementați planul, să configurați DHCP-ul și relay-ul, și să provocați toate cele patru probleme din secțiunea 10. Restul segmentelor se adaugă în Packet Tracer, după același tipar.

## 16.7 Implementarea planului

1. **Configurați ruterele** Adresele din exemplul de mai jos sunt cele din planul din secțiunea 5. Dacă planul vostru diferă, folosiți valorile voastre - dar atunci diferă și obiectivele din bancul de lucru. pe R1copiază Router> enable Router# configure terminal Router(config)# hostname R1 R1(config)# no ip domain-lookup R1(config)# interface gigabitEthernet 0/0/0 R1(config-if)# description LAN Birouri R1(config-if)# ip address 172.20.18.1 255.255.255.0 R1(config-if)# no shutdown R1(config-if)# exit R1(config)# interface gigabitEthernet 0/0/1 R1(config-if)# description LAN Servere R1(config-if)# ip address 172.20.19.193 255.255.255.192 R1(config-if)# no shutdown R1(config-if)# exit R1(config)# interface serial 0/1/0 R1(config-if)# description Legatura spre R2 R1(config-if)# ip address 172.20.20.33 255.255.255.252 R1(config-if)# clock rate 128000 R1(config-if)# no shutdown R1(config-if)# end R1# show ip interface brief pe R2copiază Router> enable Router# configure terminal Router(config)# hostname R2 R2(config)# interface gigabitEthernet 0/0/0 R2(config-if)# description LAN Productie R2(config-if)# ip address 172.20.16.1 255.255.254.0 R2(config-if)# no shutdown R2(config-if)# exit R2(config)# interface serial 0/1/0 R2(config-if)# ip address 172.20.20.34 255.255.255.252 R2(config-if)# no shutdown R2(config-if)# end **Masca /23 nu este 255.255.255.0** Producția are 512 adrese, deci masca este 255.255.254.0. Este cea mai frecventă greșeală din această lucrare, și are un simptom perfid: *jumătate* din stații răspund, cealaltă jumătate nu. Verificați cu **show ip**

`interface` `brief` și cu calculatorul de subrețele de mai sus.

2. **Adăugați rutare statică provizorie** Cele două rutere nu se cunosc încă între ele. Rutarea propriu-zisă este subiectul laboratorului 5; deocamdată scriem rutele de mână, cât să putem testa planul. pe R1copiază `R1(config)# ip route 172.20.16.0 255.255.254.0 172.20.20.34` pe R2copiază `R2(config)# ip route 172.20.18.0 255.255.255.0 172.20.20.33` `R2(config)# ip route 172.20.19.192 255.255.255.192 172.20.20.33` Verificați cu `show ip route` pe fiecare ruter, apoi dați ping de pe serverul din segmentul Servere către interfața lui R2. Dacă merge, planul de adresare este consistent.

## 16.8 DHCP pe ruter

Un ruter Cisco poate fi el însuși server DHCP. Este soluția uzuală în rețele mici și medii: nu mai aveți nevoie de un server dedicat.

1. **Configurați rezervoarele** pe R1copiază `R1(config)# ip dhcp excluded-address 172.20.18.1 172.20.18.20` `R1(config)# ip dhcp excluded-address 172.20.16.1 172.20.16.20` `R1(config)# ip dhcp pool BIROURI` `R1(dhcp-config)# network 172.20.18.0 255.255.255.0` `R1(dhcp-config)# default-router 172.20.18.1` `R1(dhcp-config)# dns-server 172.20.19.194` `R1(dhcp-config)# lease 0 8 0` `R1(dhcp-config)# exit` `R1(config)# ip dhcp pool PRODUCTIE` `R1(dhcp-config)# network 172.20.16.0 255.255.254.0` `R1(dhcp-config)# default-router 172.20.16.1` `R1(dhcp-config)# dns-server 172.20.19.194` `R1(dhcp-config)# end` `R1# show ip dhcp pool` **Excluderile se scriu întotdeauna primele** Dacă definiți rezervorul înainte de excluderi, ruterul poate alocă deja adresa gateway-ului unei stații. Rezultatul este un conflict de adrese greu de diagnosticat, pentru că apare doar când se conectează suficient de multe stații.

`lease 0 8 0` înseamnă 0 zile, 8 ore, 0 minute. Într-o rețea de birou, 8 ore acoperă o zi de lucru; într-o rețea de oaspeți, o oră este mai potrivit, ca adresele să se elibereze rapid.

2. **Testați pe o stație** În bancul din pagină: alegeți PC-BIR și apăsați butonul *cere prin DHCP*, sau scrieți în terminal `ipconfig /renew`. În Packet Tracer: comutați configurația IP a stației pe *DHCP*. pe PC-BIRcopiază `ipconfig /release` `ipconfig /renew` `ipconfig /all` Trebuie să apară: ce trebuie să apară DHCP: adresă primită de la R1 IP address.....: 172.20.18.21 Subnet Mask.....: 255.255.255.0 Default Gateway.....: 172.20.18.1 DNS Server.....: 172.20.19.194 Observați **.21:** primele douăzeci de adrese au fost excluse, deci prima liberă este a douăzeci și una. Pe ruter, verificați evidența: pe R1copiază `R1# show ip dhcp binding` Adresa apare împreună cu adresa MAC a clientului. Aceasta este legătura care face ca aceeași stație să primească aceeași adresă și data viitoare.
3. **Urmăriți DORA** detaliat în Packet Tracer În *Simulation*, filtrat pe **DHCP**, rulați `ipconfig /renew` pe stație și urmăriți cele patru mesaje. Pentru fiecare, deschideți

plicul și notați: MesajIP sursăIP destinațieUnicast sau broadcast? Discover Offer Request Ack Întrebare Ce adresă IP sursă are mesajul Discover și de ce? Vezi răspunsul 0.0.0.0. Clientul nu are încă nicio adresă - exact de aceea întreabă. Destinația este 255.255.255.255, broadcast limitat, pentru că nu știe nici adresa serverului. Consecința importantă: fiind un broadcast, mesajul **nu trece prin niciun ruter**. De aceea, pentru segmentele fără server local, este obligatoriu un relay - exact ce urmează.

## 16.9 DHCP relay

Segmentul Producție nu are server DHCP local. Serverul este pe R1, dincolo de o legătură serială - iar cererea, fiind broadcast, nu trece.

1. **Convingeți-vă că nu merge** Pe PC-PROD, apăsați *cere prin DHCP*. Trebuie să obțineți: ce trebuie să apară DHCP: nu a răspuns niciun server. Stația și-a atribuit singură o adresă APIPA. IP address.....: 169.254.x.x Subnet Mask.....: 255.255.0.0 Default Gateway.....: 0.0.0.0 **Adresa care spune totul** O stație cu adresă din 169.254.0.0/16 nu a primit răspuns la Discover. Nu căutați în altă parte: fie nu există server, fie cererea nu ajunge la el (relay lipsă, VLAN greșit, cablu în portul greșit), fie rezervorul este epuizat. Este unul dintre puținele simptome din rețelistică care indică direct cauza.
2. **Configurați relay-ul** pe R2, interfața dinspre stațiecopiază R2(config)# interface gigabitEthernet 0/0/0 R2(config-if)# ip helper-address 172.20.18.1 R2(config-if)# end Adresa este a interfeței lui R1 pe care rulează serviciul DHCP. Ce face ruterul: preia cererea de broadcast, o transformă într-un **unicast** către server și atașează informația despre rețeaua din care a venit. Serverul alege rezervorul potrivit după acea informație - de aceea trebuie să existe pe R1 un rezervor definit pentru rețeaua Producție. Cereți din nou adresă pe PC-PROD. Acum trebuie să apară: ce trebuie să apară DHCP: adresă primită de la R1 (prin relay) IP address.....: 172.20.16.21 Subnet Mask.....: 255.255.254.0 Default Gateway.....: 172.20.16.1 Observați masca: 255.255.254.0, adică /23 - vine din rezervorul PRODUCTIE, nu din cel al Birourilor. Serverul a ales corect.

## 16.10 Patru probleme de diagnosticat

Provocați fiecare situație și rezolvați-o. Notați, de fiecare dată, **ce comandă v-a arătat problema**.

**Cum se citește o suprapunere în tabela de rutare.** Ruterul alege întotdeauna ruta cu *prefixul cel mai lung*. Dacă aveți și 172.20.16.0/22, și 172.20.16.0/23, a doua câștigă pentru adresele pe care le acoperă, iar prima preia restul. Nu este o eroare - este regula. Devine problemă doar când nu ați vrut asta.

## 16.11 Sarcini de lucru

Ce veți putea face după acest capitol.

Tabela 16.5 Patru probleme de diagnosticat.

| Situația<br>cată            | provo- | Cum o produceți                                                     | Simptom                                                 | Unde se vede                                                         |
|-----------------------------|--------|---------------------------------------------------------------------|---------------------------------------------------------|----------------------------------------------------------------------|
| relay lipsă                 |        | pe R2: <code>no ip helper-address 172.20.18.1</code>                | stațiile din Producție primesc <code>169.254.x.x</code> | <code>ipconfig</code> pe stație                                      |
| gateway nescos din rezervor |        | ștergeți excluderile și cereți adresă de pe mai multe stații        | o stație primește chiar adresa gateway-ului             | <code>show ip dhcp binding</code>                                    |
| mască greșită               |        | pe R2: <code>ip address 172.20.16.1 255.255.255.0</code>            | jumătate din stațiile Producției devin inaccesibile     | calculul adresei de rețea, pe hârtie                                 |
| subrețele suprapuse         |        | pe R2: <code>ip route 172.20.16.0 255.255.252.0 172.20.20.33</code> | rutare imprevizibilă, unele destinații inaccesibile     | <code>show ip route</code> - două rute către prefixe care se acoperă |

- Elaborați planul VLSM complet, pe hârtie, și justificați fiecare prefix ales
- Completați tabelul de adresare cu toate cele nouă subrețele
- Implementați planul pe toate cele trei rutere și verificați cu `show ip route`
- Configurați rezervoare DHCP pentru Birouri, Producție, Depozit și Oaspeți
- Configurați relay pe R2 și R3 și demonstrați că funcționează
- Documentați cele patru mesaje DORA, cu adrese sursă și destinație
- Provocați și rezolvați cele patru probleme din secțiunea 10
- Calculați și raportați: câte adrese ați consumat, câte au rămas libere, în ce blocuri

## 16.12 Provocare

### Exemplu

Conducerea anunță o filială nouă, care va avea nevoie de: un segment cu 100 de stații, unul cu 25 și două legături punct-la-punct spre sediul central.

Încap noile cerințe în blocul `172.20.16.0/21`, cu planul pe care l-ați făcut? Dacă da, de unde le alocați? Dacă nu, ce ați fi putut face diferit de la început? Refaceți planul astfel încât toate subrețelele unei locații să fie **sumarizabile într-o singură rută**. Cât spațiu în plus costă această disciplină? Care este ruta sumarizată pentru fiecare dintre cele trei locații, în noul plan?

**Rezolvare.** Punctul 2 este esența proiectării ierarhice de adrese. Alocarea strict optimă, „lipită”, minimizează risipa dar face sumarizarea imposibilă: subrețelele unei locații nu au un prefix comun. Alocarea pe blocuri - de exemplu un `/23` rezervat integral pentru fiecare locație, indiferent cât se folosește din el - risipește adrese, dar reduce tabela de rutare a fiecărui ruter de la nouă intrări la trei. Într-o rețea de întreprindere, compromisul se face aproape întotdeauna în favoarea sumarizării. Adresele private sunt gratuite; tabelele mari de rutare, actualizările de rutare inutile și depanarea grea nu sunt.

### 16.13 Ce se predă

Tabela 16.6 Ce se predă.

| Livrabil                                                           | Format   | Punctaj |
|--------------------------------------------------------------------|----------|---------|
| Planul VLSM complet, cu calculele și justificarea fiecărui prefix  | document | 30 %    |
| Fișierul Packet Tracer implementat și funcțional                   | .pkt     | 30 %    |
| Documentarea DORA, cu adresele fiecărui mesaj                      | document | 15 %    |
| Cele patru probleme provocate, cu simptom și comandă de diagnostic | document | 10 %    |
| Provocarea: planul refăcut pentru sumarizare, cu cele trei rute    | document | 15 %    |

Păstrați fișierul: laboratorul 5 pornește din exact această topologie.

### Întrebări de verificare

- Un segment are nevoie de 60 de stații plus gateway. Ce prefix alegeți?
  - /26
  - /27
  - /25
  - /24
- De ce se alocă subrețelele VLSM începând cu cea mai mare?
  - Ca să fie mai ușor de citit tabelul
  - Pentru că un bloc mare trebuie să înceapă la o adresă aliniată la dimensiunea lui; alocând întâi blocurile mici, alinierea se pierde
  - Pentru că DHCP-ul cere asta
  - Nu contează ordinea
- O stație primește adresa 169.254.8.12. Ce s-a întâmplat?
  - A primit adresa de la DHCP
  - Nu a primit răspuns la Discover și și-a atribuit singură o adresă APIPA
  - Gateway-ul este configurat greșit
  - Masca este prea mică

- 
4. Ce adresa IP sursa are mesajul DHCP Discover?
- a) Adresa statiei
  - b) 0.0.0.0, pentru ca statia inca nu are adresa
  - c) 255.255.255.255
  - d) Adresa gateway-ului
5. La ce foloseste ip helper-address?
- a) Configureaza serverul DNS
  - b) Transforma broadcast-ul DHCP intr-un unicast catre serverul dintr-o alta retea
  - c) Aloca adrese statice
  - d) Sumarizeaza rutele
6. Un segment /23 are masca:
- a) 255.255.255.0
  - b) 255.255.254.0
  - c) 255.255.252.0
  - d) 255.255.255.128



# 17. Laborator 5: Rutare: statică, rute flotante și OSPF

---

---

|                                             |
|---------------------------------------------|
| Ce știe un ruter și de unde                 |
| Echipamente necesare                        |
| Topologia                                   |
| Bancul de lucru                             |
| Partea I: rutare statică                    |
| Ruta implicită și ruta flotantă             |
| Sumarizare și Null0                         |
| Partea a II-a: OSPF                         |
| Costul, care decide drumul                  |
| Experimentul care justifică tot laboratorul |
| Sarcini de lucru                            |
| Provocare                                   |
| Ce se predă                                 |
| Întrebări de verificare                     |

---

---

Aceeași topologie, două filozofii. Întâi scrieți fiecare rută de mână și vedeți exact ce știe ruterul și de unde. Apoi ștergeți tot și lăsați OSPF să facă aceeași treabă singur - după care tăiați un cablu și cronometrați cine își revine mai repede.

## 17.1 Ce știe un ruter și de unde

Un ruter proaspăt configurat cunoaște un singur lucru: rețelele legate direct de el. Atât. Despre tot restul lumii nu are nici cea mai vagă idee, iar pachetele care merg acolo sunt aruncate.

Există exact două moduri de a-i spune restul.

**Rute statice.** Le scrieți voi, una câte una. Fac exact ce ați scris, previzibil. Nu consumă bandă și nu pot fi păcălite. La 20 de rețele și 5 rutere înseamnă 100 de comenzi, iar la fiecare modificare le rescrieți.

### Protocol dinamic (OSPF)

- Ruterele vorbesc între ele și își spun ce știu
- O rețea nouă se anunță o singură dată, pe un singur ruter
- Când o legătură cade, drumul se recalculează singur



- Mai complex de configurat și de depanat

#### Observație

**distanță administrativă (AD).** Cât de mult are încredere ruterul în sursa unei rute. Conectat direct: 0. Rută statică: 1. OSPF: 110. Când două surse propun rute către aceeași destinație, câștigă cea cu AD *mai mică*.

#### Observație

**metrică.** Cât de „scump” este un drum, în interiorul aceluiasi protocol. La OSPF, metrica este costul, calculat din banda interfețelor. Se compară numai între rute din aceeași sursă.

#### Observație

**longest prefix match.** Când mai multe rute se potrivesc cu o destinație, ruterul o alege pe cea cu prefixul *cel mai lung* - cea mai specifică. O rută /24 bate una /16, indiferent de AD sau metrică.

**Ordinea deciziei, o dată pentru totdeauna.** Întâi prefixul cel mai lung. Dacă sunt mai multe cu același prefix, câștigă AD-ul cel mai mic. Dacă și AD-ul e egal, câștigă metrica cea mai mică. Aceste trei rânduri explică aproape orice comportament ciudat al unei tabele de rutare.

## 17.2 Echipamente necesare

Tabela 17.1 Echipamente necesare.

| Buc. | Echipament   | Model                   | Categoria din Packet Tracer   | La ce folosește aici                   |
|------|--------------|-------------------------|-------------------------------|----------------------------------------|
| 4    | Ruter        | 4331                    | Network Devices<br>→ Routers  | R1, R2, R3 și ruterul ISP              |
| 3    | Switch       | 2960                    | Network Devices<br>→ Switches | câte unul pentru LAN-ul fiecărui ruter |
| 3    | Calculator   | PC-PT                   | End Devices                   | o stație de test în fiecare LAN        |
| 1    | Server       | Server-PT               | End Devices                   | „Internetul”, adresa 8.8.8.8           |
| 3    | Cablu serial | Serial DCE              | Connections                   | cele trei legături dintre rutere       |
| 5    | Cablu drept  | Copper Straight-Through | Connections                   | LAN-uri și legătura spre ISP           |

**Dacă ruterul nu are destule interfețe seriale.** În Packet Tracer, ruterul 4331 pornește fără module seriale. Oprți-l din butonul de alimentare din fila *Physical*, trageți

un modul HWIC-2T într-un slot liber și porniți-l la loc. În bancul de lucru din pagină, interfețele S0/1/0 și S0/1/1 există deja.

### 17.3 Topologia

Se continuă topologia din laboratorul 4: trei rutere legate în triunghi, fiecare cu propriul LAN, plus o ieșire spre Internet prin ruterul ISP.

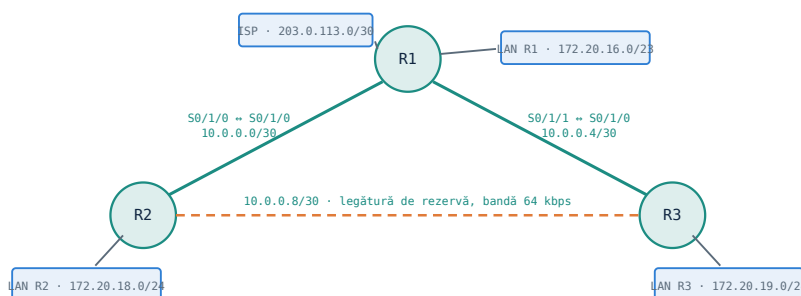


Figura 17.1 Trei routere, două legături directe și una de rezervă, lentă. Diferența de bandă este esențială: ea face ca alegerea căii să conteze.

Tabela 17.2 Topologia.

| Ruter | LAN                         | Interfețe WAN                                                                 | Router ID |
|-------|-----------------------------|-------------------------------------------------------------------------------|-----------|
| R1    | 172.20.16.0/23<br>G0/0/0 .1 | · S0/1/0: 10.0.0.1/30<br>· S0/1/1: 10.0.0.5/30<br>· G0/0/1:<br>203.0.113.1/30 | 1.1.1.1   |
| R2    | 172.20.18.0/24<br>G0/0/0 .1 | · S0/1/0: 10.0.0.2/30<br>· S0/1/1: 10.0.0.9/30                                | 2.2.2.2   |
| R3    | 172.20.19.0/24<br>G0/0/0 .1 | · S0/1/0: 10.0.0.6/30<br>· S0/1/1: 10.0.0.10/30                               | 3.3.3.3   |
| ISP   | 8.8.8.0/24<br>.1            | · G0/0/1<br>G0/0/0:<br>203.0.113.2/30                                         | -         |

### 17.4 Bancul de lucru

Topologia completă, cablată, cu stațiile deja adresate. Ruterele sunt goale - le configurați voi, întâi static, apoi cu OSPF.

**Obiectivele descriu starea finală.** Cele două legate de OSPF rămân stinse cât timp lucrați cu rute statice - este normal. Restul se bifează încă din partea I.

### 17.5 Partea I: rutare statică

1. **Adresați interfețele** pe R1copiază Router> enable Router# configure terminal Router(config)# hostname R1 R1(config)# no ip domain-lookup R1(config)# interface serial 0/1/0 R1(config-if)# ip address 10.0.0.1 255.255.255.252 R1(config-if)# clock

rate 128000 R1(config-if)# no shutdown R1(config-if)# exit R1(config)# interface serial 0/1/1 R1(config-if)# ip address 10.0.0.5 255.255.255.252 R1(config-if)# clock rate 128000 R1(config-if)# no shutdown R1(config-if)# exit R1(config)# interface gigabitEthernet 0/0/0 R1(config-if)# ip address 172.20.16.1 255.255.254.0 R1(config-if)# no shutdown R1(config-if)# exit R1(config)# interface gigabitEthernet 0/0/1 R1(config-if)# ip address 203.0.113.1 255.255.255.252 R1(config-if)# no shutdown R1(config-if)# end Repetați pentru R2, R3 și ISP, cu adresele din tabel. Pe legătura de rezervă R2–R3, declarați și banda, pentru ca OSPF să știe mai târziu că este lentă: pe R2 și pe R3, interfața S0/1/1 copiază R2(config)# interface serial 0/1/1 R2(config-if)# ip address 10.0.0.9 255.255.255.252 R2(config-if)# bandwidth 64 R2(config-if)# no shutdown **bandwidth nu schimbă viteza reală** Comanda declară doar ce bandă are legătura, pentru calculele protocoalelor de rutare. Este o etichetă informativă - dar de ea depinde ce drum alege OSPF, deci trebuie să fie corectă.

2. **Verificați punctul de plecare** pe R1 copiază R1# show ip route Trebuie să vedeți doar rute C (conectate) și L (locale). Ping-ul între LAN-uri nu funcționează - ruterele nu știu nimic unele despre altele. Verificați și acest lucru, de pe PC1: ping 172.20.19.10 trebuie să eșueze.
3. **Scrieți rutele, cu next-hop** pe R1 copiază R1(config)# ip route 172.20.18.0 255.255.255.0 10.0.0.2 R1(config)# ip route 172.20.19.0 255.255.255.0 10.0.0.6 R1(config)# ip route 10.0.0.8 255.255.255.252 10.0.0.2 pe R2 copiază R2(config)# ip route 172.20.16.0 255.255.254.0 10.0.0.1 R2(config)# ip route 172.20.19.0 255.255.255.0 10.0.0.10 R2(config)# ip route 10.0.0.4 255.255.255.252 10.0.0.1 pe R3 copiază R3(config)# ip route 172.20.16.0 255.255.254.0 10.0.0.5 R3(config)# ip route 172.20.18.0 255.255.255.0 10.0.0.9 R3(config)# ip route 10.0.0.0 255.255.255.252 10.0.0.5 **Atenție la mască** LAN-ul lui R1 este un /23, deci masca este 255.255.254.0, nu 255.255.255.0. Este exact genul de greșeală care produce un simptom bizar: jumătate din stații răspund, cealaltă jumătate nu.
4. **Testați și citiți tabela** pe R1 copiază R1# show ip route R1# show ip route static R1# ping 172.20.19.10 R1# traceroute 172.20.19.10 O intrare arată așa, și fiecare bucată înseamnă ceva: anatomia unei rute S 172.20.19.0/24 [1/0] via 10.0.0.6, Serial0/1/1 | | | | | | | | +- interfața de ieșire | | | +- next-hop: cui i se dă pachetul | | +- metrica (0 pentru rute statice) | | +- distanța administrativă (1 = statică) | +- destinația și prefixul +- sursa rutei: S = statică, C = conectată, O = OSPF
5. **Experiment: interfața de ieșire în loc de next-hop** pe R1 copiază R1(config)# no ip route 172.20.18.0 255.255.255.0 10.0.0.2 R1(config)# ip route 172.20.18.0 255.255.255.0 serial 0/1/0 R1(config)# end R1# show ip route R1# ping 172.20.18.10 Pe o legătură serială punct-la-punct funcționează perfect, pentru că nu există decât un singur destinatar posibil. Refaceți acum aceeași încercare pe o legătură Ethernet dintre două rutere și observați diferența. Întrebare De ce varianta cu interfață eșuează pe Ethernet, dar merge pe serial? Vezi răspunsul Pe Ethernet, ruterul știe pe ce interfață

să scoată pachetul, dar nu știe ce adresă MAC destinație să pună în cadru: pe acel segment pot exista zeci de dispozitive. Fără next-hop nu poate face ARP. Pe serial, problema nu apare: legătura are exact două capete, nu există adrese de nivel 2 și nici nevoie de rezoluție. Cadrul pleacă pur și simplu pe fir. Reveniți apoi la varianta cu next-hop.

## 17.6 Ruta implicită și ruta flotantă

1. **Ruta implicită spre Internet** pe R1copiază R1(config)# ip route 0.0.0.0 0.0.0.0 203.0.113.2 0.0.0.0 0.0.0.0 se potrivește cu absolut orice destinație - și tocmai de aceea are prefixul cel mai scurt posibil, deci pierde în fața oricărei alte rute. Este „dacă nu știu unde, trimite acolo”. Testați de pe PC1: ping 8.8.8.8. Pe ISP, configurați ruta de întoarcere spre firmă - altfel răspunsurile nu se mai întorc: pe ISPCopiază ISP(config)# ip route 172.20.16.0 255.255.252.0 203.0.113.1
2. **Ruta flotantă, de rezervă** Traficul spre LAN-ul lui R3 merge normal direct. Dacă legătura cade, vrem să treacă prin R2. Se obține dând rutei de rezervă o **distanță administrativă mai mare**: pe R1copiază R1(config)# ip route 172.20.19.0 255.255.255.0 10.0.0.2 200 Verificați cu show ip route: ruta cu AD 200 **nu apare** în tabelă atât timp cât cea cu AD 1 este validă. Ambele există în configurație, dar numai una intră în tabelă. Acum închideți legătura directă și priviți din nou: pe R1copiază R1(config)# interface serial 0/1/1 R1(config-if)# shutdown R1(config-if)# end R1# show ip route R1# traceroute 172.20.19.10 Ruta [200/0] via 10.0.0.2 a apărut, iar traceroute arată un salt în plus. Redeschideți interfața și verificați că ruta de rezervă se retrage singură.

## 17.7 Sumarizare și Null0

Cele trei LAN-uri sunt 172.20.16.0/23, 172.20.18.0/24 și 172.20.19.0/24. ISP-ul nu are niciun motiv să știe despre ele separat - îi ajunge o singură rută.

### Exemplu

**Rezolvare.** În binar, al treilea octet: 16 = 00010000, 18 = 00010010, 19 = 00010011. Blocurile acoperite sunt .16, .17 (din /23), .18 și .19 - adică patru valori consecutive începând de la 16. Primii 6 biți ai octetului sunt comuni, deci prefixul este 16 + 6 = /22. Ruta sumarizată: 172.20.16.0/22, adică 172.20.16.0 255.255.252.0.

Pe R1, adăugați o rută spre Null0 pentru porțiunea nefolosită a blocului sumarizat, ca traficul spre adrese inexistente să fie aruncat imediat:

### Exemplu 17.1 pe R1 copiază

- 
- 1
  - 2 R1(config)# ip route 172.20.16.0 255.255.252.0 null0
-

Testați ping 172.20.20.5 de pe ISP: pachetul ajunge la R1 și moare acolo, în loc să circule prin rețea căutând o destinație inexistentă. Destinațiile reale continuă să funcționeze, pentru că rutele lor au prefixul mai lung și câștigă.

IP DESTINAȚIE  
172.20.19.10

| SURSĂ | DESTINAȚIE     | DIRECȚIE    | TEST          |
|-------|----------------|-------------|---------------|
| C     | 10.0.0.0/30    | S0/1/0      | 172.20.19.8 ✗ |
| C     | 10.0.0.4/30    | S0/1/1      | 172.20.19.8 ✗ |
| S     | 172.20.18.0/24 | 10.0.0.2    | 172.20.19.0 ✗ |
| S     | 172.20.19.0/24 | 10.0.0.6    | 172.20.19.0 = |
| C     | 172.20.16.0/23 | G0/0/0      | 172.20.18.0 ✗ |
| S     | 172.20.16.0/22 | Null0       | 172.20.16.0 = |
| S     | 0.0.0.0/0      | 203.0.113.2 | 0.0.0.0 =     |

rezultat 172.20.19.0/24 via 10.0.0.6

Tabela este ordonată de la cel mai specific prefix spre cel mai general. Se aplică prima rută care se potrivește.

Figura 17.2 Tabela de rutare a lui R1 - testați destinații.

Încercați 172.20.19.10 (LAN-ul lui R3), 172.20.20.5 (în blocul sumarizat, dar inexistent) și 8.8.8.8 (Internet). Fiecare cade pe altă rută, din motive diferite - urmăriți coloana de test ca să vedeți exact de ce.

## 17.8 Partea a II-a: OSPF

- Ștergeți tot ce ați scris** pe fiecare ruter copiază R1(config)# no ip route 172.20.18.0 255.255.255.0 10.0.0.2 R1(config)# no ip route 172.20.19.0 255.255.255.0 10.0.0.6 R1(config)# no ip route 172.20.19.0 255.255.255.0 10.0.0.2 200 R1(config)# no ip route 10.0.0.8 255.255.255.252 10.0.0.2 Păstrați ruta implicită spre ISP și cea spre Null0. Verificați cu **show ip route** că au rămas doar rutele conectate - și că ping-ul între LAN-uri a încetat din nou să funcționeze. Este punctul de plecare curat pentru partea a doua.
- Activați OSPF** pe R1 copiază R1(config)# router ospf 1 R1(config-router)# router-id 1.1.1.1 R1(config-router)# network 172.20.16.0 0.0.1.255 area 0 R1(config-router)# network 10.0.0.0 0.0.0.3 area 0 R1(config-router)# network 10.0.0.4 0.0.0.3 area 0 R1(config-router)# passive-interface gigabitEthernet 0/0/0 R1(config-router)# default-information originate R1(config-router)# end pe R2 copiază R2(config)# router ospf 1 R2(config-router)# router-id 2.2.2.2 R2(config-router)# network 172.20.18.0 0.0.0.255 area 0 R2(config-router)# network 10.0.0.0 0.0.0.3 area 0 R2(config-router)# network 10.0.0.8 0.0.0.3 area 0 R2(config-router)# passive-interface gigabitEthernet 0/0/0 R2(config-router)# end Pe R3, la fel, cu **router-id 3.3.3.3** și rețelele 172.20.19.0 0.0.0.255, 10.0.0.4 0.0.0.3 și 10.0.0.8 0.0.0.3. **Măștile wildcard nu sunt măști de rețea** Comanda **network** din OSPF folosește *wildcard*: inversul măștii. Pentru un /24 se scrie 0.0.0.255, pentru un /30 0.0.0.3, pentru un /23 0.0.1.255. Regula

rapidă: scădeți fiecare octet al măștii din 255. **Cele două comenzi opționale care contează** `passive-interface` pe interfețele dinspre stații: rețeaua rămâne anunțată, dar nu se mai trimite Hello-uri inutile spre calculatoare - economie de bandă și o ușă de atac închisă.

`default-information originate` pe R1: anunță celorlalte rutere că el are ieșire spre Internet. Fără ea, R2 și R3 nu vor ști niciodată de ruta implicită.

3. **Verificați adiacențele** pe R1 copiați `R1# show ip ospf neighbor` ce trebuie să apară  
Neighbor ID Pri State Dead Time Address Interface 2.2.2.2 1 FULL/BDR 00:00:35  
10.0.0.2 Serial0/1/0 3.3.3.3 1 FULL/BDR 00:00:35 10.0.0.6 Serial0/1/1 Ambele adiacențe  
trebuie să fie în starea FULL. Dacă lipsesc: SimptomCauză cea mai probabilă Niciun  
vecin în listărețeaua nu e inclusă în **network**, wildcard greșit, sau interfața e `passive`  
Blocat în `INITHello`-urile pleacă dar nu se întorc: filtrare, VLAN greșit, cablu Blocat  
în `EXSTART` sau `EXCHANGEMTU` diferit între cele două interfețe Adiacența oscilează în  
intervale hello/dead diferite, sau legătură instabilă alte verificări utile copiați `R1# show`  
`ip protocols` `R1# show ip ospf interface brief` `R1# show ip route ospf`
4. **Citiți ce a învățat ruterul** `show ip route ospf` pe R1 trebuie să arate ceva de  
forma: ce trebuie să apară O 172.20.18.0/24 [110/65] via 10.0.0.2, Serial0/1/0 O  
172.20.19.0/24 [110/65] via 10.0.0.6, Serial0/1/1 110 este distanța administrativă a  
OSPF, 65 este costul cumulat: 64 pentru legătura serială plus 1 pentru interfața LAN  
de la capăt. Verificați pe R2 că a apărut și ruta implicită, marcată `0*E2` - semnul că a  
fost injectată din afara OSPF-ului. Testați acum toate ping-urile: totul funcționează  
fără nicio rută scrisă de mână. Aceasta este diferența, în trei comenzi pe fiecare ruter  
în loc de trei rute pe fiecare ruter - și diferența crește cu fiecare rețea nouă.

## 17.9 Costul, care decide drumul

OSPF alege drumul cu costul total cel mai mic. Costul unei interfețe se calculează din bandă:

Tabela 17.3 Costul, care decide drumul.

| Interfață                           | Bandă     | Cost = 100 Mbps /<br>bandă |
|-------------------------------------|-----------|----------------------------|
| Serial, implicit                    | 1544 kbps | 64                         |
| Serial cu <code>bandwidth 64</code> | 64 kbps   | 1562                       |
| FastEthernet                        | 100 Mbps  | 1                          |
| GigabitEthernet                     | 1 Gbps    | 1 - costul minim este 1    |

Ultimul rând este o problemă reală: cu referința implicită, o legătură de 100 Mbps și una de 10 Gbps sunt indistingibile. Se corectează ridicând referința:

Exemplu 17.2 pe toate ruterele, aceeași valoare copiați

```

1
2 R1(config)# router ospf 1
3 R1(config-router)# auto-cost reference-bandwidth 10000

```

### Observație

**Aceeși valoare pretutindeni.** Dacă un ruter are altă referință decât vecinii lui, calculează costuri incompatibile și rezultă rutare asimetrică sau bucle. IOS afișează un avertisment; nu îl ignorați.

### 17.9.1 Forțați o cale suboptimă

Exemplu 17.3 pe R1 copiază

```

1
2 R1# traceroute 172.20.19.10
3 R1(config)# interface serial 0/1/1
4 R1(config-if)# ip ospf cost 2000
5 R1(config-if)# end
6 R1# show ip route ospf
7 R1# traceroute 172.20.19.10

```

Cu costul 2000 pe legătura directă, drumul prin R2 ( $64 + 1562 + 1 = 1627$ ) devine mai ieftin decât cel direct ( $2000 + 1 = 2001$ ). **traceroute** arată acum un salt în plus. Reveniți cu **no ip ospf cost** și verificați că drumul se întoarce.

**Ce ați demonstrat.** Nu ați schimbat niciun cablu și nicio adresă. Ați schimbat un număr, iar întreaga rețea a recalculat drumurile singură, în ambele sensuri. Acesta este, în esență, tot avantajul rutării dinamice.

### 17.10 Experimentul care justifică tot laboratorul

Aveți acum aceeași rețea funcționând în două moduri. Comparați-le pe singurul criteriu care contează într-o defecțiune: **cât durează până revine traficul**.

1. De pe PC2, porniți un ping continuu spre PC3 (în Packet Tracer: **ping -t 172.20.19.10**).
2. Închideți legătura pe care circulă traficul: pe R1, **interface s0/1/1** urmat de **shutdown**.
3. Numărați câte pachete se pierd până când ping-ul revine.
4. Repetați experimentul cu rutare statică, folosind ruta flotantă configurată în partea I.

**Concluzia experimentului.** Ruta flotantă comută rapid, dar *doar* dacă interfața locală cade fizic. Dacă legătura rămâne electric activă și se rupe undeva în mijloc - un scenariu foarte frecvent pe legăturile închiriate - ruta statică nu observă nimic și traficul dispare în neant. OSPF observă, pentru că nu se bazează pe starea cablului, ci pe faptul că vecinul răspunde.

Tabela 17.4 Experimentul care justifică tot laboratorul.

|                                               | Rutare statică cu rută flotantă          | OSPF                                            |
|-----------------------------------------------|------------------------------------------|-------------------------------------------------|
| Detectarea căderii                            | doar dacă interfața trece în <b>down</b> | prin Hello: cel mult 40 s, tipic mult mai puțin |
| Timp până la comutare                         | imediat, dacă interfața a căzut fizic    | secunde: detectare + flooding + Dijkstra        |
| Dacă legătura e „sus” dar nu transportă       | nu detectează niciodată                  | detectează prin lipsa Hello-urilor              |
| Efort de configurare la adăugarea unei rețele | rută nouă pe fiecare ruter               | o comandă <b>network</b> , pe un singur ruter   |

### 17.11 Sarcini de lucru

Ce veți putea face după acest capitol.

- Configurați rutarea statică completă și demonstrați conectivitate între toate cele trei LAN-uri
- Demonstrați funcționarea unei rute flotante: arătați tabela înainte și după căderea legăturii principale
- Calculați și configurați ruta sumarizată, plus ruta spre Null0
- Ștergeți rutarea statică și configurați OSPF single-area pe toate ruterele
- Documentați tabela de vecini a fiecărui ruter, cu stările adiacenței
- Forțați OSPF să aleagă o cale suboptimă modificând costul; demonstrați cu **traceroute**
- Măsurați și raportați timpii de convergență în ambele scenarii
- Configurați **default-information originate** și verificați că R2 și R3 primesc ruta implicită, marcată 0\*E2

### 17.12 Provocare

#### Exemplu

Reveniți la rutare statică și construiți intenționat o buclă de rutare: traficul de la LAN-ul lui R2 spre o anumită destinație trebuie să circule  $R2 \rightarrow R1 \rightarrow R2 \rightarrow R1 \dots$  la nesfârșit. Ce rute trebuie scrise, exact, pe fiecare ruter? Vor circula pachetele la infinit? Argumentați. Rulați **traceroute** spre acea destinație și descrieți ce vedeți. Ar putea OSPF să producă aceeași buclă? De ce da sau de ce nu? Ce mecanism din nivelul 3 lipsește la nivelul 2 și de aceea buclele de acolo sunt fatale?

**Rezolvare.** **1.** Este suficient ca fiecare dintre cele două rutere să indice către celălalt pentru aceeași destinație: pe R2 `ip route 192.168.99.0 255.255.255.0 10.0.0.1` și pe R1 `ip route 192.168.99.0 255.255.255.0 10.0.0.2`. **2.** Nu. Fiecare trecere printr-un ruter decrementează TTL-ul; la zero pachetul este aruncat și se trimite un ICMP *time exceeded* înapoi la sursă. **3.** **traceroute** va afișa o alternanță perfect vizibilă R1, R2, R1, R2... până la epuizarea TTL-ului. Este cea mai clară demonstrație vizuală



a unei bucle de rutare. **4.** Nu, atât timp cât toate ruterele sunt în aceeași arie: fiecare are harta completă a topologiei și calculează drumuri minime pe un graf - algoritmul lui Dijkstra nu produce cicluri prin construcție. Buclele în OSPF apar doar la granițele dintre arii sau la redistribuirea între protocoale diferite, adică exact acolo unde vederea de ansamblu se pierde. **5.** Câmpul TTL. Antetul Ethernet nu are niciun echivalent, motiv pentru care o buclă de nivel 2 multiplică cadrele la infinit - și pentru care STP a trebuit inventat.

### 17.13 Ce se predă

Tabela 17.5 Ce se predă.

| Livrabil                                                                          | Format   | Punctaj |
|-----------------------------------------------------------------------------------|----------|---------|
| Fișier Packet Tracer cu rutarea statică completă, inclusiv rută flotantă și Null0 | .pkt     | 25 %    |
| Fișier Packet Tracer cu OSPF configurat și funcțional                             | .pkt     | 25 %    |
| Tabelele de vecini și de rutare, comentate                                        | document | 20 %    |
| Măsurătorile de convergență, cu concluzia proprie                                 | document | 15 %    |
| Provocarea: bucla construită, cu răspunsurile la cele cinci întrebări             | document | 15 %    |

### Întrebări de verificare

- Ce înseamnă [110/65] într-o intrare din tabela de rutare?
  - 110 hopuri și 65 de secunde
  - Distanța administrativă 110 și costul cumulativ 65
  - Costul 110 și metrica 65
  - Adresa 110.0.0.65
- O rută statică are AD 200 și există și o rută OSPF spre aceeași destinație. Care intră în tabelă?
  - Cea statică, pentru că rutele statice au prioritate
  - Cea OSPF, pentru că 110 este mai mic decât 200
  - Ambandoua
  - Depinde de metrică

3. Comanda network din OSPF foloseste:
  - a) Masca de retea
  - b) Masca wildcard
  - c) Nicio masca
  - d) Adresa de broadcast
4. Adiacenta OSPF ramane blocata in starea EXCHANGE. Ce verificati intai?
  - a) Adresa IP a vecinului
  - b) MTU-ul celor doua interfete
  - c) Parola de enable
  - d) Cablul
5. De ce o ruta statica nu detecteaza o legatura care este electric activa dar nu transporta trafic?
  - a) Pentru ca nu are metrica
  - b) Pentru ca se bazeaza exclusiv pe starea interfeței locale, nu pe raspunsul vecinului
  - c) Pentru ca AD este prea mare
  - d) Pentru ca nu foloseste ARP
6. Doua rute se potrivesc cu destinatia: una /16 cu AD 1 si una /24 cu AD 110. Care castiga?
  - a) Cea /16, pentru ca are AD mai mica
  - b) Cea /24, pentru ca prefixul mai lung se compara primul
  - c) Depinde de metrica
  - d) Niciuna



## 18. Laborator 6: Liste de acces și NAT

---

---

- Cum se citește o listă de acces
- Echipamente necesare
- Topologia și politica
- Bancul de lucru
- Pasul 0: rețeaua funcțională, fără filtre
- Traducerea politicii în reguli
- Contorul, cel mai bun instrument de depanare
- NAT: ieșirea în Internet
- Sarcini de lucru
- Provocare
- Ce se predă
- Întrebări de verificare

---

---

Politica de securitate a unei firme se scrie în cuvinte, dar rețeaua înțelege doar reguli. Lucrarea aceasta face traducerea: cinci cerințe formulate de conducere devin liste de acces, plasate corect pe interfețe și pe direcții. Apoi toată firma iese în Internet - pe o singură adresă publică.

### 18.1 Cum se citește o listă de acces

O listă de acces este o listă de reguli, parcursă **de sus în jos**, până la prima potrivire. Ce se potrivește primul decide, iar restul listei nici măcar nu se mai citește. La final, dacă nimic nu s-a potrivit, există un **deny** invizibil care aruncă pachetul.

#### Observație

**listă standard.** Vede doar *adresa sursă*. Se numerotează 1–99. Utilă când vrei să spunei „de la aceste stații, nicăieri” sau „numai de la aceste stații”. Se plasează cât mai aproape de *destinație*, tocmai pentru că nu știe care este destinația.

#### Observație

**listă extinsă.** Vede sursa, destinația, protocolul și portul. Se numerotează 100–199, sau primește un nume. Se plasează cât mai aproape de *sursă*, ca traficul interzis să nu

consume rețeaua degeaba.

### Observație

**maskă wildcard.** Inversul măștii de rețea: 0 înseamnă „bitul acesta trebuie să coincidă”, 1 înseamnă „orice”. Pentru o rețea /24 se scrie 0.0.0.255. Scurtături: **host** 10.1.10.5 înseamnă 10.1.10.5 0.0.0.0, iar **any** înseamnă 0.0.0.0 255.255.255.255.

### Observație

**direcția.** in filtrează pachetele care *intră* în ruter pe acea interfață, înainte de rutare. out filtrează pachetele care *ies*, după rutare. Aceeași listă pe aceeași interfață, dar pe direcții diferite, face lucruri complet diferite.

### Observație

**Trei greșeli care se fac de fiecare dată.** 1. *Ordinea.* O regulă generală pusă deasupra uneia specifice o face pe a doua inutilă - nu se va potrivi niciodată.  
2. *Deny-ul implicit.* O listă care conține numai reguli **deny** blochează absolut tot, inclusiv ce voiți să permiteți.  
3. *Direcția.* Lista corectă, pusă pe direcția greșită, nu vede niciodată traficul pe care trebuia să îl filtreze - și are contorul la zero.

## 18.2 Echipamente necesare

Tabela 18.1 Echipamente necesare.

| Buc. | Echipament   | Model                      | Categoria din Packet Tracer   | La ce folosește aici                             |
|------|--------------|----------------------------|-------------------------------|--------------------------------------------------|
| 2    | Ruter        | 4331                       | Network Devices<br>→ Routers  | R-FIRMA (rute-<br>rul de margine)<br>și ISP      |
| 1    | Switch       | 2960                       | Network Devices<br>→ Switches | SW1, cu trei<br>VLAN-uri                         |
| 3    | Calculator   | PC-PT                      | End Devices                   | câte o stație de<br>test per departa-<br>ment    |
| 2    | Server       | Server-PT                  | End Devices                   | SRV-WEB în<br>DMZ și „Inter-<br>netul” (8.8.8.8) |
| 1    | Cablu serial | Serial DCE                 | Connections                   | R-FIRMA ↔<br>ISP                                 |
| 6    | Cablu drept  | Copper<br>Straight-Through | Connections                   | restul legăturilor                               |

Topologia reia structura din laboratorul 3: trei VLAN-uri pe SW1, trunk spre ruter, subinterfețe pe G0/0/0. DMZ-ul se leagă pe o interfață fizică separată, G0/0/1.

### 18.3 Topologia și politica

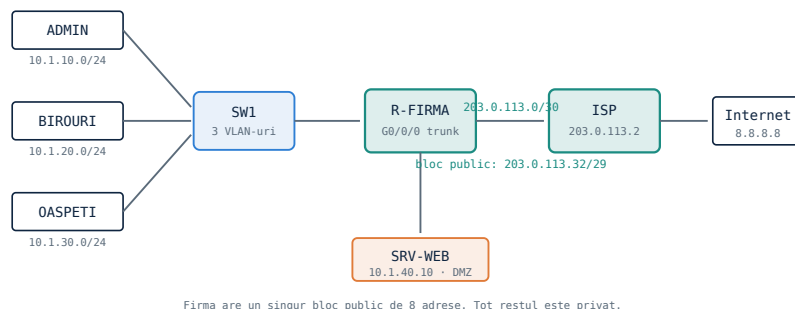


Figura 18.1 Rețeaua firmei, cu trei segmente interne, o zonă demilitarizată și o singură ieșire spre Internet.

Tabela 18.2 Topologia și politica.

| Segment                 | VLAN | Rețea           | Gateway   | Stație de test             |
|-------------------------|------|-----------------|-----------|----------------------------|
| ADMIN                   | 10   | 10.1.10.0/24    | 10.1.10.1 | PC-ADM<br>10.1.10.11       |
| BIROURI                 | 20   | 10.1.20.0/24    | 10.1.20.1 | PC-BIR<br>10.1.20.11       |
| OASPETI                 | 30   | 10.1.30.0/24    | 10.1.30.1 | PC-OSP<br>10.1.30.11       |
| DMZ                     | -    | 10.1.40.0/24    | 10.1.40.1 | SRV-WEB<br>10.1.40.10      |
| Legătura spre ISP       | -    | 203.0.113.0/30  | -         | R-FIRMA .1,<br>ISP .2      |
| Bloc public alo-<br>cat | -    | 203.0.113.32/29 | -         | .33 – .38 utiliza-<br>bile |

#### Observație

**Politica de securitate, așa cum a formulat-o conducerea.** *P1.* Oaspeții au acces la Internet și la serverul web din DMZ, dar la nicio altă resursă internă.

*P2.* Biroul are acces peste tot, cu excepția segmentului ADMIN.

*P3.* Serverul web din DMZ nu are voie să inițieze conexiuni spre rețeaua internă. Poate doar să răspundă.

*P4.* Ruterul poate fi administrat prin SSH doar de pe stațiile din ADMIN.

*P5.* Serverul web trebuie să fie accesibil din Internet, la o adresă publică fixă.

### 18.4 Bancul de lucru

### 18.5 Pasul 0: rețeaua funcțională, fără filtre

Înainte de a filtra ceva, totul trebuie să funcționeze. Este o disciplină care se învață greu, dar economisește ore: dacă aplicați ACL-uri peste o rețea care nu merge, nu veți ști

niciodată care problemă e a cui.

1. **VLAN-uri pe switch** pe SW1copiază Switch> enable Switch# configure terminal Switch(config)# hostname SW1 SW1(config)# vlan 10 SW1(config-vlan)# name ADMIN SW1(config-vlan)# vlan 20 SW1(config-vlan)# name BIROURI SW1(config-vlan)# vlan 30 SW1(config-vlan)# name OASPETI SW1(config-vlan)# exit SW1(config)# interface fastEthernet 0/1 SW1(config-if)# switchport mode access SW1(config-if)# switchport access vlan 10 SW1(config-if)# exit SW1(config)# interface fastEthernet 0/2 SW1(config-if)# switchport mode access SW1(config-if)# switchport access vlan 20 SW1(config-if)# exit SW1(config)# interface fastEthernet 0/3 SW1(config-if)# switchport mode access SW1(config-if)# switchport access vlan 30 SW1(config-if)# exit SW1(config)# interface gigabitEthernet 0/1 SW1(config-if)# switchport mode trunk SW1(config-if)# end
2. **Router-on-a-stick, DMZ și ieșirea spre ISP** pe R-FIRMAcopiază Router(config)# hostname R-FIRMA R-FIRMA(config)# interface gigabitEthernet 0/0/0 R-FIRMA(config-if)# no ip address R-FIRMA(config-if)# no shutdown R-FIRMA(config-if)# exit R-FIRMA(config)# interface gigabitEthernet 0/0/0.10 R-FIRMA(config-subif)# encapsulation dot1Q 10 R-FIRMA(config-subif)# ip address 10.1.10.1 255.255.255.0 R-FIRMA(config-subif)# exit R-FIRMA(config)# interface gigabitEthernet 0/0/0.20 R-FIRMA(config-subif)# encapsulation dot1Q 20 R-FIRMA(config-subif)# ip address 10.1.20.1 255.255.255.0 R-FIRMA(config-subif)# exit R-FIRMA(config)# interface gigabitEthernet 0/0/0.30 R-FIRMA(config-subif)# encapsulation dot1Q 30 R-FIRMA(config-subif)# ip address 10.1.30.1 255.255.255.0 R-FIRMA(config-subif)# exit R-FIRMA(config)# interface gigabitEthernet 0/0/1 R-FIRMA(config-if)# description DMZ R-FIRMA(config-if)# ip address 10.1.40.1 255.255.255.0 R-FIRMA(config-if)# no shutdown R-FIRMA(config-if)# exit R-FIRMA(config)# interface serial 0/1/0 R-FIRMA(config-if)# ip address 203.0.113.1 255.255.255.252 R-FIRMA(config-if)# clock rate 128000 R-FIRMA(config-if)# no shutdown R-FIRMA(config-if)# exit R-FIRMA(config)# ip route 0.0.0.0 0.0.0.0 203.0.113.2 R-FIRMA(config)# end pe ISPCopiază Router(config)# hostname ISP ISP(config)# interface serial 0/1/0 ISP(config-if)# ip address 203.0.113.2 255.255.255.252 ISP(config-if)# no shutdown ISP(config-if)# exit ISP(config)# interface gigabitEthernet 0/0/0 ISP(config-if)# ip address 8.8.8.1 255.255.255.0 ISP(config-if)# no shutdown ISP(config-if)# exit ISP(config)# ip route 203.0.113.32 255.255.255.248 203.0.113.1 ISP(config)# end Observați ce **nu** are ISP-ul: nicio rută spre rețelele private ale firmei. Este chiar rostul NAT-ului, pe care îl configurăm la final.
3. **Completați tabelul de referință** De pe fiecare stație internă, ping către toate celelalte și către SRV-WEB și Internet. Totul trebuie să funcționeze. Notați rezultatele - va fi referința față de care veți verifica efectul fiecărei liste. De la \ cătreADMINBIROURIIOASPETIDMZInternet ADMIN- BIROURI- OASPETI- DMZ-

## 18.6 Traducerea politicii în reguli

Fiecare politică se traduce răspunzând, în ordine, la trei întrebări: **ce tip de listă** îmi trebuie, **pe ce interfață** o pun și **pe ce direcție**.

### 18.6.1 P1 - Oaspeții

#### Exemplu

Sursa este mereu aceeași (10.1.30.0/24), dar destinația decide dacă traficul e permis sau nu. O listă standard vede doar sursa, deci **nu poate exprima** politica. Ne trebuie o listă **extinsă**.

Plasarea: lista extinsă se pune cât mai aproape de sursă, deci pe subinterfața G0/0/0.30, direcția **in** - traficul interzis nici măcar nu ajunge să fie rutat.

#### Exemplu 18.1 P1 pe R-FIRMA copiază

```

1
2 R-FIRMA(config)# ip access-list extended OASPETI-IN
3 R-FIRMA(config-ext-nacl)# remark permite accesul la serverul web
   ➔ din DMZ
4 R-FIRMA(config-ext-nacl)# 10 permit tcp 10.1.30.0 0.0.0.255 host
   ➔ 10.1.40.10 eq 80
5 R-FIRMA(config-ext-nacl)# remark blocheaza orice alta resursa
   ➔ interna
6 R-FIRMA(config-ext-nacl)# 20 deny ip 10.1.30.0 0.0.0.255 10.0.0.0
   ➔ 0.255.255.255
7 R-FIRMA(config-ext-nacl)# remark restul , adica Internetul , este
   ➔ permis
8 R-FIRMA(config-ext-nacl)# 30 permit ip 10.1.30.0 0.0.0.255 any
9 R-FIRMA(config-ext-nacl)# exit
10
11 R-FIRMA(config)# interface gigabitEthernet 0/0/0.30
12 R-FIRMA(config-subif)# ip access-group OASPETI-IN in
13 R-FIRMA(config-subif)# end

```

Testați acum, de pe PC-OSP:

**Al doilea rând merită citit de două ori.** Serverul web este accesibil oaspeților - dar *numai pe portul 80*. Ping-ul către el eșuează, și este corect să eșueze: politica spunea „acces la serverul web”, nu „acces la server”. O listă extinsă face exact distincția asta, iar un test cu ping v-ar face să credeți greșit că regula nu funcționează.

Ordinea celor trei reguli este întreaga soluție. Regula 10 este mai specifică decât 20, deci trebuie să fie deasupra ei; regula 30 este cea mai generală și încheie lista. Inversați regulile 10 și 30 și testați din nou: veți vedea exact cum se rupe politica.



Tabela 18.3 P1 - Oaspeții.

| Test            | Rezultat | De ce                                                                               |
|-----------------|----------|-------------------------------------------------------------------------------------|
| ping 10.1.10.11 | blocat   | regula 20: destinație în 10.0.0.0/8                                                 |
| ping 10.1.40.10 | blocat   | tot regula 20 - regula 10 permite doar <b>TCP portul 80</b> , iar ping-ul este ICMP |
| ping 8.8.8.8    | permis   | regula 30: orice altă destinație                                                    |

### 18.6.2 P2 - Biroul

Exemplu 18.2 P2 copiază

```

1
2 R-FIRMA(config)# ip access-list extended BIROURI-IN
3 R-FIRMA(config-ext-nacl)# 10 deny ip 10.1.20.0 0.0.0.255 10.1.10.0
   ↪ 0.0.0.255
4 R-FIRMA(config-ext-nacl)# 20 permit ip 10.1.20.0 0.0.0.255 any
5 R-FIRMA(config-ext-nacl)# exit
6 R-FIRMA(config)# interface gigabitEthernet 0/0/0.20
7 R-FIRMA(config-subif)# ip access-group BIROURI-IN in

```

Verificați: biroul nu mai ajunge la ADMIN, dar ajunge la DMZ și în Internet. Verificați și sensul invers - de pe PC-ADM către PC-BIR: **funcționează**, pentru că lista este aplicată doar pe traficul care *intră* dinspre birou.

### 18.6.3 P3 - DMZ-ul nu inițiază

#### Exemplu

„Poate doar să răspundă” înseamnă că traficul dinspre DMZ spre interior este permis **numai** dacă face parte dintr-o conversație pornită din interior. Un ACL clasic nu poate exprima asta - el vede pachete, nu conexiuni. Soluția minimală este cuvântul cheie **established**.

Exemplu 18.3 P3 copiază

```

1
2 R-FIRMA(config)# ip access-list extended DMZ-IN
3 R-FIRMA(config-ext-nacl)# 10 permit tcp host 10.1.40.10 10.0.0.0
   ↪ 0.255.255.255 established
4 R-FIRMA(config-ext-nacl)# 20 deny ip host 10.1.40.10 10.0.0.0
   ↪ 0.255.255.255
5 R-FIRMA(config-ext-nacl)# 30 permit ip host 10.1.40.10 any

```

```

6 R-FIRMA(config-ext-nacl)# exit
7 R-FIRMA(config)# interface gigabitEthernet 0/0/1
8 R-FIRMA(config-if)# ip access-group DMZ-IN in

```

#### Observație

**Limitele lui established.** Verifică doar biții ACK și RST din antetul TCP, deci un pachet fabricat cu ACK setat trece. Nu funcționează pentru UDP și ICMP. Într-o rețea reală, aici s-ar folosi un firewall cu stare sau un ACL reflexiv, cu **reflect** și **evaluate**.

### 18.6.4 P4 - Accesul de administrare

Traficul spre liniile VTY ale ruterului nu se filtrează pe interfață, ci direct pe linii, cu **access-class**. Aici este suficientă o listă **standard**: contează doar de unde vine conexiunea.

Exemplu 18.4 P4 copiază

```

1
2 R-FIRMA(config)# ip domain-name firma.local
3 R-FIRMA(config)# crypto key generate rsa
4 R-FIRMA(config)# username admin privilege 15 secret Admin123
5 R-FIRMA(config)# access-list 10 remark doar statiile ADMIN
6 R-FIRMA(config)# access-list 10 permit 10.1.10.0 0.0.0.255
7
8 R-FIRMA(config)# line vty 0 4
9 R-FIRMA(config-line)# transport input ssh
10 R-FIRMA(config-line)# login local
11 R-FIRMA(config-line)# access-class 10 in
12 R-FIRMA(config-line)# exec-timeout 5 0

```

### 18.6.5 Verificați-vă înțelegerea

Ultimul pachet este important: vine din BIROURI, deci nu se potrivește cu nicio regulă din această listă și cade pe **deny** implicit. Nu are însă nicio consecință - lista este aplicată pe 0/0/0.30, iar acel pachet nu trece niciodată pe acolo. **Un ACL contează doar acolo unde este aplicat.**

## 18.7 Contorul, cel mai bun instrument de depanare

Exemplu 18.5 verificare copiază

```

1
2 R-FIRMA# show access-lists
3 R-FIRMA# clear access-list counters

```

Ieșirea arată, pentru fiecare regulă, de câte ori s-a potrivit:

| lista de acces                                                                                                                                         |            |            |          |          |          |
|--------------------------------------------------------------------------------------------------------------------------------------------------------|------------|------------|----------|----------|----------|
| <pre> permit tcp 10.1.30.0 0.0.0.255 host 10.1.40.10 eq 80 deny ip 10.1.30.0 0.0.0.255 10.0.0.0 0.255.255.255 permit ip 10.1.30.0 0.0.0.255 any </pre> |            |            |          |          |          |
| PACHET                                                                                                                                                 | SURSĂ      | DESTINAȚIE | PROTOCOL | LINIA    | REZULTAT |
| oaspete → web DMZ                                                                                                                                      | 10.1.30.11 | 10.1.40.10 | tcp:80   | linia 1  | permit   |
| oaspete → SSH pe server                                                                                                                                | 10.1.30.11 | 10.1.40.10 | tcp:22   | linia 2  | deny     |
| oaspete → ping ADMIN                                                                                                                                   | 10.1.30.11 | 10.1.10.11 | icmp     | linia 2  | deny     |
| oaspete → Internet                                                                                                                                     | 10.1.30.11 | 8.8.8.8    | tcp:443  | linia 3  | permit   |
| birou → web DMZ                                                                                                                                        | 10.1.20.11 | 10.1.40.10 | tcp:80   | implicit | deny     |

La finalul oricărei liste există un deny any implicit, care nu se scrie și nu se vede în configurație.

Figura 18.2 Lista OASPETI-IN, testată pe cinci pachete.

### Exemplu 18.6 ce trebuie să apară

```

1
2 Extended IP access list OASPETI-IN
3     10 remark permite accesul la serverul web din DMZ
4     20 permit tcp 10.1.30.0 0.0.0.255 host 10.1.40.10 eq 80
5     40 deny ip 10.1.30.0 0.0.0.255 10.0.0.0 0.255.255.255 (8 match
6         ↪ (es))
7     60 permit ip 10.1.30.0 0.0.0.255 any (4 match(es))

```

**Cum se citește.** Dacă o regulă are **zero potriviri**, traficul nu ajunge la ea: fie lista e pe interfața greșită, fie pe direcția greșită, fie o regulă de deasupra prinde deja traficul. Dacă are potriviri, dar rezultatul nu e cel dorit, atunci regula e scrisă greșit. *Cele două cazuri se repară complet diferit*, iar contorul vă spune în care sunteți.

Recompletați acum tabelul de conectivitate de la secțiunea 5 și comparați cu referința. Fiecare celulă care s-a schimbat trebuie să corespundă unei politici - dacă s-a schimbat ceva neprevăzut, aveți o regulă prea largă.

## 18.8 NAT: ieșirea în Internet

Toate adresele interne sunt private, deci nerutabile în Internet. ISP-ul nici măcar nu are rute spre ele. Ne trebuie traducere.

1. **Marcați interfețele** fiecare interfață e „inside” sau „outside”copiază R-FIRMA(config)# interface gigabitEthernet 0/0/0.10 R-FIRMA(config-subif)# ip nat inside R-FIRMA(config-subif)# exit R-FIRMA(config)# interface gigabitEthernet 0/0/0.20 R-FIRMA(config-subif)# ip nat inside R-FIRMA(config-subif)# exit R-FIRMA(config)# interface gigabitEthernet 0/0/0.30 R-FIRMA(config-subif)# ip nat inside R-FIRMA(config-subif)# exit R-FIRMA(config)# interface gigabitEthernet 0/0/1 R-FIRMA(config-if)# ip nat

inside R-FIRMA(config-if)# exit R-FIRMA(config)# interface serial 0/1/0 R-FIRMA(config-if)# ip nat outside R-FIRMA(config-if)# exit Fără aceste marcaje, NAT-ul nu face absolut nimic - indiferent cât de corecte sunt regulile. Este prima verificare atunci când „NAT-ul nu merge”.

2. **PAT pentru toate stațiile interne** PAT cu adresa interfeței de ieșirecopiază R-FIRMA(config)# access-list 100 remark ce se translateaza R-FIRMA(config)# access-list 100 permit ip 10.1.10.0 0.0.0.255 any R-FIRMA(config)# access-list 100 permit ip 10.1.20.0 0.0.0.255 any R-FIRMA(config)# access-list 100 permit ip 10.1.30.0 0.0.0.255 any R-FIRMA(config)# ip nat inside source list 100 interface serial 0/1/0 overload Iată a doua utilizare a listelor de acces, promisă în cursul 7: aici lista nu filtrează nimic - doar **descrie** traficul care trebuie translatat. Aceeași sintaxă, alt rol.
3. **NAT static pentru serverul web (P5)** o adresă publică fixă pentru SRV-WEBcopiază R-FIRMA(config)# ip nat inside source static 10.1.40.10 203.0.113.33 Testați din „Internet”: de pe serverul NET, ping 203.0.113.33 - răspunde serverul din DMZ. Iar ping 10.1.40.10 de acolo nu merge și nu are cum: adresa privată nu există în Internet. Alternativ, dacă vreți să publicați doar portul 80 și să păstrați adresa publică pentru altceva: port forwardingcopiază R-FIRMA(config)# ip nat inside source static tcp 10.1.40.10 80 203.0.113.1 80
4. **Priviți traducerea cu ochii voștri** verificare NATcopiază R-FIRMA# show ip nat translations R-FIRMA# show ip nat statistics R-FIRMA# clear ip nat translation \* În bancul din pagină: dați de pe PC-BIR ping 8.8.8.8 și citiți tabelul *Traseul ultimului pachet*. Coloana *IP sursă* se schimbă exact la trecerea prin R-FIRMA: până acolo este 10.1.20.11, după aceea 203.0.113.1. PozițieIP sursăIP destinație între SW1 și R-FIRMA10.1.20.118.8.8 între R-FIRMA și ISP203.0.113.18.8.8 răspunsul, între ISP și R-FIRMA răspunsul, între R-FIRMA și SW1 Completați ultimele două rânduri singuri, apoi verificați în Packet Tracer, în *Simulation*. **Excepția de la regula celor două adrese** În laboratorul 1 ați învățat că adresa IP nu se schimbă niciodată pe drum. NAT este exact încălcarea acelei reguli - și motivul pentru care este criticat de puriști. Este, în același timp, motivul pentru care IPv4 mai trăiește.

| STAȚIE INTERNĂ |  | SERVER EXTERN    |
|----------------|--|------------------|
| 192.168.1.10   |  | 93.184.216.34:80 |

| INTERN                 | TRANSLATAT | DESTINAȚIE |
|------------------------|------------|------------|
| nicio traducere activă |            |            |

Ruterul păstrează portul translatat cât timp conexiunea e activă. Adresa publică este una singură: 203.0.113.1 – porturile sunt cele care deosebesc stațiile.

Figura 18.3 PAT, ca reper: cum arată tabela.

## 18.9 Sarcini de lucru

**Ce veți putea face după acest capitol.**

- Construiți topologia și demonstrați conectivitate completă înainte de orice filtru
- Completați tabelul de referință al conectivității
- Implementați politicile P1–P4 și demonstrați efectul fiecăreia, prin teste
- Recompletați tabelul de conectivitate și explicați fiecare diferență
- Configurați PAT pentru toate segmentele interne
- Configurați NAT static pentru serverul web și demonstrați accesul din Internet
- Documentați traducerea adreselor, segment cu segment
- Provocați o regulă cu zero potriviri, explicați de ce și reparați-o

## 18.10 Provocare

### Exemplu

Conducerea revine cu trei cerințe suplimentare. Implementați-le pe toate, fără a strica politicile existente.

**Oaspeții nu au voie la Internet în afara programului de lucru** (luni–vineri, 8:00–18:00). **Un singur calculator din BIROURI**, cel de la contabilitate (10.1.20.50), **trebuie să ajungă la segmentul ADMIN** - dar numai pe portul 3389. **Descărcările prin FTP sunt interzise pentru toată lumea, mai puțin pentru ADMIN.**

Pentru fiecare cerință, precizați: tipul de listă, poziția exactă a regulii în lista existentă, interfața și direcția.

**Rezolvare. 1.** Vă trebuie un **time-range** și o regulă condiționată temporal: indicație R-FIRMA(config)# time-range PROGRAM R-FIRMA(config-time-range)# periodic weekdays 8:00 to 18:00 ! apoi, în lista OASPETI-IN, regula 30 devine: R-FIRMA(config-ext-nacl)# no 30 R-FIRMA(config-ext-nacl)# 30 permit ip 10.1.30.0 0.0.0.255 any time-range PROGRAM **2.** Regula trebuie să fie **deasupra** regulii 10 din BIROURI-IN, care blochează tot traficul spre ADMIN. Aici se vede de ce numerotarea explicită a regulilor este esențială - puteți insera la poziția 5, fără să rescrieți lista: indicație R-FIRMA(config)# ip access-list extended BIROURI-IN R-FIRMA(config-ext-nacl)# 5 permit tcp host 10.1.20.50 10.1.10.0 0.0.0.255 eq 3389 **3.** FTP folosește portul de control 21 și, în modul activ, portul de date 20. O regulă care blochează doar eq 21 oprește stabilirea sesiunii, deci este suficientă în practică - dar gândiți-vă ce se întâmplă cu FTP pasiv, unde portul de date este negociat dinamic și poate fi oricare peste 1024. Este exact limitarea despre care vorbea cursul 7 și motivul pentru care firewall-urile moderne inspectează conținutul, nu doar anteturile.

## 18.11 Ce se predă

### Întrebări de verificare

1. Ce vede o lista de acces standard?

Tabela 18.4 Ce se predă.

| Livrabil                                                                       | Format   | Punctaj |
|--------------------------------------------------------------------------------|----------|---------|
| Fișierul Packet Tracer cu politicile P1–P5 implementate                        | .pkt     | 35 %    |
| Cele două tabele de conectivitate, înainte și după, cu explicarea diferențelor | document | 25 %    |
| Documentarea translatării NAT, segment cu segment                              | document | 20 %    |
| Provocarea: cele trei cerințe noi, cu regulă, poziție, interfață și direcție   | document | 20 %    |

- a) Sursa si destinatia
  - b) Doar adresa sursa
  - c) Doar portul
  - d) Adresa MAC
2. O lista contine doar reguli deny. Ce trece prin ea?
  - a) Tot ce nu e mentionat explicit
  - b) Nimic: la final exista un deny implicit
  - c) Doar traficul ICMP
  - d) Depinde de directie
3. O regula din ACL are contorul la zero, desi traficul respectiv exista. Ce verificati?
  - a) Ca regula e scrisa gresit
  - b) Ca lista e aplicata pe interfata si directia corecta, sau ca o regula de deasupra prinde deja traficul
  - c) Adresa MAC a sursei
  - d) Masca de subretea
4. De ce se plaseaza o lista extinsa cat mai aproape de sursa?
  - a) Ca sa fie mai usor de scris
  - b) Ca traficul interzis sa fie oprit inainte de a consuma resursele retelei
  - c) Pentru ca IOS o cere
  - d) Nu conteaza plasarea
5. Ce face comanda `ip nat inside source list 100 interface serial 0/1/0 overload?`

- a) Filtreaza traficul din lista 100
  - b) Translateaza adresele descrise de lista 100 spre adresa interfetei seriale, folosind porturi distincte
  - c) Creeaza o ruta statica
  - d) Blocheaza NAT-ul
6. Un oaspete poate deschide site-ul din DMZ, dar ping-ul catre acelasi server esueaza. E o problema?
- a) Da, ceva e configurat gresit
  - b) Nu: regula permite doar TCP portul 80, iar ping-ul este ICMP
  - c) Da, lipseste NAT-ul
  - d) Nu, ping-ul nu functioneaza niciodata prin ruter

## 19. Laborator 7: Securitatea rețelei de acces și wireless

---

---

- Regula de aur a acestei lucrări
- Echipamente necesare
- Topologia
- Bancul de lucru
- Pasul 0: rețeaua care merge
- Atacul 1: switch-ul care devine hub
- Atacul 2: serverul DHCP fals
- Atacul 3: ARP poisoning
- Închiderea celorlalte uși
- Rețeaua wireless
- Lista de verificare
- Sarcini de lucru
- Provocare
- Ce se predă
- Întrebări de verificare

---

---

Ultima lucrare face ceva ce se evită de obicei: execută trei atacuri reale pe rețeaua proprie - un switch transformat în hub, un server DHCP fals și un dispozitiv care se dă drept gateway - apoi le închide, pe rând, ușile. La final, adăugați o rețea wireless și îi comparați securitatea cu cea a cablului.

### 19.1 Regula de aur a acestei lucrări

#### Observație

**Citiți înainte de a începe.** Toate atacurile din această lucrare se execută *exclusiv* pe topologia proprie, în laborator sau în bancul de lucru din pagină. Aceleași tehnici aplicate pe o rețea reală - a facultății, a unei firme, a unui vecin - constituie infracțiune, indiferent de intenție și indiferent dacă „nu s-a stricat nimic”.

Scopul lucrării nu este să învățați să atacați, ci să înțelegeți *de ce* funcționează atacurile, pentru că numai așa configurațiile defensive încetează să fie o listă memorată și devin decizii înțelese.



Cele trei atacuri au același numitor comun: protocoalele rețelei locale au fost proiectate într-o epocă în care oricine avea acces fizic la cablu era, prin definiție, de încredere. Niciunul dintre ele nu are autentificare. Toate contramăsurile de mai jos adaugă, într-o formă sau alta, exact ce lipsește: o sursă de adevăr despre cine are voie să spună ce.

Tabela 19.1 Regula de aur a acestei lucrări.

| Atacul        | De ce funcționează                                                                   | Contramăsura                   |
|---------------|--------------------------------------------------------------------------------------|--------------------------------|
| CAM overflow  | tabela de comutare e finită;                                                         | <code>port-security</code>     |
| Rogue DHCP    | plină, switch-ul inundă tot DHCP nu are autentificare; clientul crede primul răspuns | <code>ip dhcp snooping</code>  |
| ARP poisoning | ARP acceptă răspunsuri nesolicitate, fără verificare                                 | <code>ip arp inspection</code> |

## 19.2 Echipamente necesare

Tabela 19.2 Echipamente necesare.

| Buc. | Echipament         | Model           | Categoria din Packet Tracer           | La ce folosește aici                          |
|------|--------------------|-----------------|---------------------------------------|-----------------------------------------------|
| 1    | Ruter              | 4331            | Network Devices<br>→ Routers          | gateway și server DHCP legitim                |
| 1    | Switch de acces    | 2960            | Network Devices<br>→ Switches         | SW-ACCES, echipamentul pe care îl securizați  |
| 1    | Switch suplimentar | 2960            | Network Devices<br>→ Switches         | „switch-ul adus de cineva”, pentru BPDU Guard |
| 2    | Calculator         | PC-PT           | End Devices                           | PC-1 și PC-2, clienți DHCP                    |
| 1    | Server             | Server-PT       | End Devices                           | SERVER, cu adresă statică                     |
| 1    | Server (atacator)  | Server-PT       | End Devices                           | ATACATOR, cu serviciul DHCP pornit            |
| 1    | Punct de acces     | Access Point-PT | Network Devices<br>→ Wireless Devices | rețeaua wireless                              |
| 1    | Laptop             | Laptop-PT       | End Devices                           | clientul wireless - necesită modul WPC300N    |
| 1    | Server AAA         | Server-PT       | End Devices                           | opțional, pentru partea cu RADIUS             |

### Observație

**Laptopul nu are placă wireless din fabrică.** În Packet Tracer, deschideți laptopul → fila *Physical* → opriți-l din butonul de alimentare → trageți afară modulul Ethernet existent → puneți în loc WPC300N → porniți-l la loc. Este pasul pe care îl ratează toată lumea la prima încercare.

## 19.3 Topologia

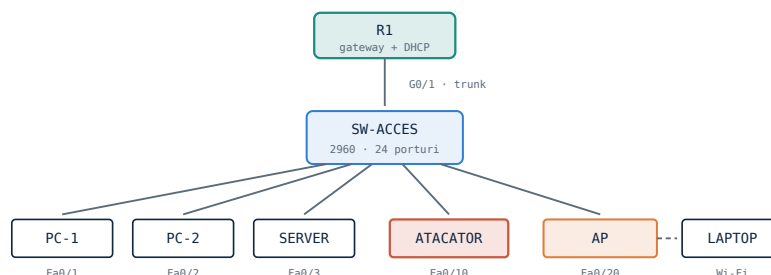


Figura 19.1 Topologia lucrării. Stația ATACATOR este un calculator obișnuit, conectat la un port obișnuit - exact cum ar arăta un laptop adus de cineva în sala de ședințe.

Tabela 19.3 Topologia.

| Dispozitiv                  | Adresă                                 | Port           | VLAN    |
|-----------------------------|----------------------------------------|----------------|---------|
| R1 (gateway și server DHCP) | 192.168.100.1/24, pe G0/0/0.10         | G0/0/0 (trunk) | 10      |
| PC-1, PC-2                  | prin DHCP                              | Fa0/1, Fa0/2   | 10      |
| SERVER                      | 192.168.100.10, static                 | Fa0/3          | 10      |
| ATACATOR                    | 192.168.100.66, static, cu DHCP pornit | Fa0/10         | 10      |
| AP wireless                 | -                                      | Fa0/20         | 10      |
| VLAN nativ al trunk-ului    | -                                      | -              | 99, gol |

## 19.4 Bancul de lucru

Rețeaua este deja cablată și adresată, iar **ATACATOR are deja un server DHCP fals pornit** - este punctul de plecare ostil. Switch-ul suplimentar stă deoparte, neconectat; îl veți lega voi, la secțiunea cu BPDU Guard, din panoul *Cablurile*.

## 19.5 Pasul 0: rețeaua care merge

Exemplu 19.1 pe R1 copiază

```

2 Router(config)# hostname R1
3 R1(config)# interface gigabitEthernet 0/0/0
4 R1(config-if)# no ip address
5 R1(config-if)# no shutdown
6 R1(config-if)# exit
7 R1(config)# interface gigabitEthernet 0/0/0.10
8 R1(config-subif)# encapsulation dot1Q 10
9 R1(config-subif)# ip address 192.168.100.1 255.255.255.0
10 R1(config-subif)# exit
11 R1(config)# interface gigabitEthernet 0/0/0.99
12 R1(config-subif)# encapsulation dot1Q 99 native
13 R1(config-subif)# exit
14
15 R1(config)# ip dhcp excluded-address 192.168.100.1 192.168.100.19
16 R1(config)# ip dhcp pool LAN
17 R1(dhcp-config)# network 192.168.100.0 255.255.255.0
18 R1(dhcp-config)# default-router 192.168.100.1
19 R1(dhcp-config)# dns-server 192.168.100.10
20 R1(dhcp-config)# end

```

---

#### Exemplu 19.2 pe SW-ACCES copiază

---

```

1
2 Switch(config)# hostname SW-ACCES
3 SW-ACCES(config)# vlan 10
4 SW-ACCES(config-vlan)# name UTILIZATORI
5 SW-ACCES(config-vlan)# vlan 99
6 SW-ACCES(config-vlan)# name NATIV-GOL
7 SW-ACCES(config-vlan)# exit
8
9 SW-ACCES(config)# interface range fastEthernet 0/1 - 21
10 SW-ACCES(config-if-range)# switchport mode access
11 SW-ACCES(config-if-range)# switchport access vlan 10
12 SW-ACCES(config-if-range)# exit
13
14 SW-ACCES(config)# interface gigabitEthernet 0/1
15 SW-ACCES(config-if)# switchport mode trunk
16 SW-ACCES(config-if)# switchport trunk native vlan 99
17 SW-ACCES(config-if)# switchport trunk allowed vlan 10,99
18 SW-ACCES(config-if)# end

```

---

**interface range** vă scutește de 21 de repetări. Tot ce scrieți în modul config-if-range

se aplică pe toate porturile din interval deodată. Promptul vă amintește unde sunteți. Este comanda pe care o veți folosi cel mai des într-o zi de configurare reală.

Verificați că rețeaua funcționează: de pe PC-2, `ipconfig /renew` apoi `ping 192.168.100.10`.

## 19.6 Atacul 1: switch-ul care devine hub

Tabela de comutare a unui switch este finită. Dacă se umple, switch-ul nu mai poate învăța adrese noi și începe să inunde traficul necunoscut pe toate porturile - adică se comportă ca un hub, iar atacatorul vede tot.

1. **Situația inițială** pe SW-ACCES copiază SW-ACCES# `show mac address-table SW-ACCES# show mac address-table count` Notați câte intrări există și care este capacitatea. Generați trafic între PC-1 și SERVER și observați că traficul **nu** ajunge la ATACATOR.
2. **Demonstrația efectului** Nici Packet Tracer, nici bancul din pagină nu au un generator de cadre cu adrese MAC false, așa că demonstrăm efectul, nu atacul: goliți tabela și priviți ce se întâmplă cu primul cadru. pe SW-ACCES copiază SW-ACCES# `clear mac address-table dynamic` Trimiteți imediat un cadru de la PC-1 spre SERVER. Cu tabela goală, destinația este necunoscută, deci cadrul pleacă pe **toate** porturile, inclusiv pe Fa0/10. *Aceasta este exact starea în care un atac CAM overflow menține switch-ul, permanent.*
3. **Contramăsura: port security** pe toate porturile de acces copiază SW-ACCES(config)# `interface range fastEthernet 0/1 - 21 SW-ACCES(config-if-range)# switchport mode access SW-ACCES(config-if-range)# switchport port-security SW-ACCES(config-if-range)# switchport port-security maximum 2 SW-ACCES(config-if-range)# switchport port-security mac-address sticky SW-ACCES(config-if-range)# switchport port-security violation restrict SW-ACCES(config-if-range)# end SW-ACCES# show port-security SW-ACCES# show port-security address` **De ce maximum 2 și nu 1** Un singur MAC pare mai sigur, dar rupe două situații complet normale: un telefon IP cu un calculator conectat în spatele lui, și o mașină virtuală cu adresă proprie. Valoarea 2 sau 3 este compromisul uzual în rețelele de birou.
4. **Comparați cele trei acțiuni la violare** AcțiuneCadrele în excesContorNotificare-Portul protectaruncatenunurămâne activ `restrict` aruncat daSNMP + syslogrămâne activ `shutdown` daSNMP + syslog **err-disabled** Testați ambele comportamente: cu `restrict`, verificați că se schimbă doar contorul din `show port-security interface fastEthernet 0/1`; cu `shutdown`, portul trece în `err-disabled` și trebuie reactivat manual: reactivarea unui port `err-disabled` copiază SW-ACCES(config)# `interface fastEthernet 0/1 SW-ACCES(config-if)# shutdown SW-ACCES(config-if)# no shutdown` **Alegeți cu cap acțiunea shutdown** pare cea mai sigură, dar înseamnă că orice utilizator care își schimbă laptopul rămâne fără rețea până când vine cineva să reactiveze

portul. În majoritatea rețelelor de birou, **restrict** este compromisul corect: blochează traficul suspect, dar anunță în loc să pedepsească.

### 19.7 Atacul 2: serverul DHCP fals

Un atacator pornește pe calculatorul lui un server DHCP și răspunde mai repede decât cel legitim. Toate stațiile care se conectează după aceea primesc de la el adresă, mască - și, mai ales, un gateway care este chiar el.

1. **Executați atacul** În bancul din pagină, ATACATOR are deja serviciul pornit. În Packet Tracer, înlocuiți stația cu un *Server-PT*, dați-i adresa **192.168.100.66** și activați DHCP din fila *Services* → *DHCP*, cu aceste valori: ParametruValoare la atacatorValoare legitimă Default Gateway192.168.100.66 - el însuși192.168.100.1 DNS Server192.168.100.66192.168.100.10 Start IP192.168.100.150192.168.100.20 Pe PC-1, cereți o adresă nouă: pe PC-1copiază ipconfig /release ipconfig /renew ipconfig Priviți câmpul *Default Gateway*. Dacă arată **192.168.100.66**, atacul a reușit: tot traficul lui spre exterior trece acum prin atacator, care îl poate citi și transmite mai departe. **De ce funcționează** DHCP nu are nicio autentificare. Clientul acceptă prima ofertă primită și nu are cum să distingă un server legitim de unul fals. Este exact aceeași categorie de problemă ca la ARP: un protocol proiectat într-o epocă în care rețeaua locală era considerată un spațiu de încredere.
2. **Contramăsura: DHCP snooping** pe SW-ACCEScopiază SW-ACCES(config)# ip dhcp snooping SW-ACCES(config)# ip dhcp snooping vlan 10 SW-ACCES(config)# no ip dhcp snooping information option SW-ACCES(config)# interface gigabitEthernet 0/1 SW-ACCES(config-if)# ip dhcp snooping trust SW-ACCES(config-if)# exit SW-ACCES(config)# interface range fastEthernet 0/1 - 21 SW-ACCES(config-if-range)# ip dhcp snooping limit rate 10 SW-ACCES(config-if-range)# end SW-ACCES# show ip dhcp snooping Ideea, într-o propoziție: **numai porturile declarate de încredere au voie să trimită răspunsuri DHCP**. Uplink-ul spre ruter este de încredere; toate porturile de utilizator nu sunt, iar mesaje Offer venite de acolo sunt aruncate de switch. Repetați atacul, de pe PC-2: pe PC-2copiază ipconfig /release ipconfig /renew ipconfig De data aceasta gateway-ul trebuie să fie **192.168.100.1**. Verificați și pe ruter, cu **show ip dhcp binding**, că alocarea a venit de la el. Priviți și **show ip dhcp snooping binding**: switch-ul a construit o evidență cu perechile (MAC, IP, VLAN, port) observate în tranzacțiile legitime. Această tabelă este fundația pasului următor.

### 19.8 Atacul 3: ARP poisoning

Atacatorul trimite mesaje ARP nesolicitate, în care pretinde că adresa IP a gateway-ului îi aparține. Stațiile își actualizează cache-ul și încep să îi trimită lui tot traficul destinat exteriorului.

1. **Situația normală** pe PC-1copiază ping 192.168.100.1 arp -a Notăți adresa MAC

asociată gateway-ului. Comparați-o cu adresa MAC reală a interfeței lui R1, din `show arp` pe ruter sau din panoul de configurare al stației.

2. **Efectul atacului** Nici Packet Tracer, nici bancul din pagină nu au un generator de ARP fals, așa că demonstrația se face conceptual: dacă atacatorul ar trimite un ARP reply cu `192.168.100.1 → MAC-ul lui`, cache-ul lui PC-1 ar arăta acea adresă MAC în dreptul gateway-ului. De atunci înainte, fiecare cadru destinat exteriorului ar pleca spre atacator, care l-ar putea citi și apoi transmite mai departe - astfel încât nimic să nu pară stricat. Verificați-vă înțelegerea După atac, ce s-ar schimba în ieșirea comenzii `arp -a` de pe PC-1? Ce ar rămâne **neschimbat** și de ce? Vezi răspunsul S-ar schimba doar adresa MAC din dreptul lui `192.168.100.1`. Adresa IP a gateway-ului ar rămâne aceeași, ping-ul spre el ar funcționa în continuare, navigarea ar merge, iar utilizatorul nu ar observa absolut nimic. Aceasta este calitatea care face atacul periculos: nu strică nimic vizibil. Singurul indiciu, pentru cine se uită, este că aceeași adresă MAC ar apărea în dreptul mai multor adrese IP - sau că MAC-ul gateway-ului s-ar schimba fără niciun motiv.
3. **Contramăsura: Dynamic ARP Inspection** pe SW-ACCES - necesită DHCP snooping activcopiază SW-ACCES(config)# ip arp inspection vlan 10 SW-ACCES(config)# interface gigabitEthernet 0/1 SW-ACCES(config-if)# ip arp inspection trust SW-ACCES(config-if)# exit SW-ACCES(config)# arp access-list SERVERE SW-ACCES(config-arp-nacl)# permit ip host 192.168.100.10 mac host 000A.0B0C.0D0E SW-ACCES(config)# ip arp inspection filter SERVERE vlan 10 SW-ACCES(config)# end SW-ACCES# show ip arp inspection **Atenție la stațiile cu adresă statică** DAI verifică mesajele ARP față de evidența construită de DHCP snooping. O stație configurată static nu apare acolo, deci mesajele ei ARP legitime ar fi aruncate. Trebuie declarată explicit, printr-un ARP ACL - ca în exemplul de mai sus pentru SERVER. Este cel mai frecvent motiv pentru care „DAI a stricat rețeaua”.

## 19.9 Închiderea celorlalte uși

### 19.9.1 VLAN hopping

---

#### Exemplu 19.3 pe SW-ACCES copiază

---

```

1
2 SW-ACCES(config)# interface range fastEthernet 0/1 - 21
3 SW-ACCES(config-if-range)# switchport mode access
4 SW-ACCES(config-if-range)# switchport nonegotiate
5 SW-ACCES(config-if-range)# exit
6
7 SW-ACCES(config)# interface gigabitEthernet 0/1
8 SW-ACCES(config-if)# switchport nonegotiate
9 SW-ACCES(config-if)# exit

```

```

10
11 SW-ACCES(config)# vlan 999
12 SW-ACCES(config-vlan)# name NEFOLOSIT
13 SW-ACCES(config-vlan)# exit
14 SW-ACCES(config)# interface fastEthernet 0/21
15 SW-ACCES(config-if)# switchport access vlan 999
16 SW-ACCES(config-if)# shutdown

```

Tabela 19.4 VLAN hopping.

| Măsura                                                    | Ce atac închide                                                 |
|-----------------------------------------------------------|-----------------------------------------------------------------|
| <code>switchport nonegotiate</code> pe porturile de acces | switch spoofing: portul nu mai poate fi convins să devină trunk |
| VLAN nativ 99, gol (configurat la pasul 0)                | double tagging: nu mai există stații în VLAN-ul nativ           |
| <code>switchport trunk allowed vlan 10,99</code>          | limitează ce poate injecta cineva pe trunk                      |
| porturile nefolosite închise, în VLAN mort                | accesul fizic la o priză liberă nu mai dă nimic                 |

### 19.9.2 Atacuri asupra Spanning Tree

Exemplu 19.4 pe SW-ACCES copiază

```

1
2 SW-ACCES(config)# interface range fastEthernet 0/1 - 21
3 SW-ACCES(config-if-range)# spanning-tree portfast
4 SW-ACCES(config-if-range)# spanning-tree bpduguard enable

```

Testați în bancul de lucru: alegeți **SW-STRAIN**, în panoul *Cablurile* apăsați *leagă* pe Fa0/1 și alegeți SW-ACCES : Fa0/15. Apoi, pe SW-ACCES:

Exemplu 19.5 pe SW-ACCES copiază

```

1
2 SW-ACCES# show interfaces fastEthernet 0/15 status

```

Portul apare imediat **err-disabled**: un port de utilizator nu are niciun motiv să primească BPDU-uri, iar dacă primește, acolo s-a conectat un switch - legitim sau nu. Scoateți cablul și reactivați portul cu `shutdown / no shutdown`.

### 19.9.3 Administrarea switch-ului

Exemplu 19.6 pe SW-ACCES copiază

```

1

```

```

2 SW-ACCES(config)# ip domain-name retele.local
3 SW-ACCES(config)# crypto key generate rsa
4 SW-ACCES(config)# username admin privilege 15 secret Admin123
5 SW-ACCES(config)# enable secret Enable123
6 SW-ACCES(config)# service password-encryption
7 SW-ACCES(config)# line vty 0 4
8 SW-ACCES(config-line)# transport input ssh
9 SW-ACCES(config-line)# login local
10 SW-ACCES(config-line)# exec-timeout 5 0
11 SW-ACCES(config-line)# exit
12 SW-ACCES(config)# banner motd #Acces autorizat numai. Activitatea
    ↪ este inregistrata.#
13 SW-ACCES(config)# end
14 SW-ACCES# copy running-config startup-config

```

---

### 19.10 Rețeaua wireless

Un punct de acces este, la nivelul 2, un **hub în aer**: tot ce transmite un client ajunge la toți ceilalți din raza lui. Diferența față de un hub obișnuit este că nu trebuie să intri în clădire ca să te conectezi la el. De aceea criptarea nu este opțională.

În bancul din pagină, laptopul este deja asociat: verificați cu `ping 192.168.100.10` și priviți tabelul *Traseul ultimului pachet* - segmentul prin AP apare la fel ca oricare altul, pentru că din punctul de vedere al rețelei chiar este la fel.

1. **Punct de acces cu WPA2-PSK** în Packet Tracer Adăugați un *Access Point-PT* pe Fa0/20 și un laptop cu placă wireless. Configurați AP-ul din fila *Config* → *Port 1*: Parametru Valoare SSIDRL-LAB AuthenticationWPA2-PSK EncryptionAES Pass PhraseParolaLung4Lab2026 Channell1, 6 sau 11 Pe laptop, din *Desktop* → *PC Wireless*, conectați-vă la rețea. Verificați cu `ipconfig` că a primit adresă prin DHCP și dați ping către SERVER.
2. **Patru experimente de securitate** ExperimentCe observațiConcluzia Setări autentificarea pe Disabledoricine se conectează instantaneu rețea deschisă este exact atât: deschisă Setări WEP cu o cheie de 10 caractere hexafuncționează, iar Packet Tracer o acceptă fără reținereWEP dă impresia de securitate; în realitate se sparge în minute Dezactivați difuzarea SSID-uluirețeaua nu mai apare în listă, dar conectarea manuală funcționeazăascunderea nu este securitate: SSID-ul apare în cadrele clienților legitimi Activați filtrarea MAC și adăugați doar laptopulalte dispozitive sunt respinseutil ca igienă, inutil față de cineva care poate copia o adresă MAC
3. **Autentificare individuală, cu RADIUS** Adăugați un server AAA în rețeaua cablată, activați serviciul AAA din fila *Services* și creați doi utilizatori. Înregistrați AP-ul ca client RADIUS, apoi schimbați autentificarea AP-ului pe WPA2 cu server RADIUS.



Conectați laptopul cu numele de utilizator și parola proprii. Ștergeți apoi utilizatorul din server și încercați din nou. **Diferența care contează** Cu **WPA2-PSK**, toată lumea folosește aceeași parolă. Când pleacă un angajat, trebuie schimbată pentru toți - deci, în practică, nu se schimbă niciodată.

Cu **WPA2-Enterprise**, fiecare are credențiale proprii. Un utilizator se revocă în zece secunde, individual, iar jurnalele arată cine s-a conectat, când și de pe ce dispozitiv.

### 19.11 Lista de verificare

Aceasta este, de fapt, livrabilul cel mai util al întregului semestru: ce se configurează pe orice switch de acces, înainte de a-l pune în producție.

Tabela 19.5 Lista de verificare.

| #  | Măsura                                                          | De ce                                                            |
|----|-----------------------------------------------------------------|------------------------------------------------------------------|
| 1  | nume, banner, parole cu <b>enable secret</b>                    | un echipament fără nume nu a fost configurat conștient niciodată |
| 2  | SSH în loc de Telnet, cu login local și <b>exec-timeout</b>     | Telnet trimite parolele în clar                                  |
| 3  | <b>access-class</b> pe liniile VTY                              | limitează de unde se poate încerca conectarea                    |
| 4  | porturile de acces: <b>mode access + nonegotiate</b>            | închide switch spoofing                                          |
| 5  | VLAN nativ gol, listă explicită de VLAN-uri pe trunk            | închide double tagging                                           |
| 6  | <b>port-security</b> cu <b>restrict</b> și sticky               | limitează CAM overflow și MAC spoofing                           |
| 7  | <b>portfast + bpduguard</b> pe porturile de utilizator          | închide atacurile STP                                            |
| 8  | <b>ip dhcp snooping</b> , cu uplink-ul de încredere             | închide rogue DHCP                                               |
| 9  | <b>ip arp inspection</b> , plus ARP ACL pentru stațiile statice | închide ARP poisoning                                            |
| 10 | porturile nefolosite: închise, în VLAN mort                     | o priză liberă nu mai este o ușă                                 |

### 19.12 Sarcini de lucru

**Ce veți putea face după acest capitol.**

- Construiți topologia și verificați funcționarea normală, înainte de orice atac
- Demonstrați efectul unei tabele CAM golite și configurați port security pe toate porturile de acces
- Executați atacul cu server DHCP fals și demonstrați că DHCP snooping îl oprește
- Explicați în scris mecanismul ARP poisoning și configurați Dynamic ARP Inspection, inclusiv ARP ACL pentru stația statică

- Aplicați toate măsurile împotriva VLAN hopping și a atacurilor STP; demonstrați BPDU Guard în acțiune
- Configurați accesul de administrare prin SSH, cu restricție de sursă
- Construiți rețeaua wireless cu WPA2-PSK, apoi migrați-o pe autentificare RADIUS
- Redactați lista finală de verificare pentru securizarea unui switch de acces, cu justificarea fiecărei linii

### 19.13 Provocare

#### Exemplu

Sunteți chemat să evaluați securitatea rețelei unei firme mici. Găsiți următoarea configurație pe switch-ul de acces:

```
configurația găsită
hostname Switch
!
interface range FastEthernet0/1 - 24
switchport mode dynamic auto
!
interface GigabitEthernet0/1
switchport mode trunk
switchport trunk native vlan 1
!
line vty 0 4
password cisco
login
transport input all
!
enable password cisco
```

Cerințe:

Identificați **toate** problemele de securitate din această configurație - sunt cel puțin șapte. Pentru fiecare, precizați ce atac devine posibil și din ce curs provine. Scrieți configurația corectată, completă. Ordonăți problemele după gravitate și argumentați ordinea.

**Rezolvare.** `switchport mode dynamic auto` pe toate porturile - DTP activ, deci **switch spoofing** posibil de pe orice port de utilizator (cursul 12). VLAN nativ 1 pe trunk, adică același cu VLAN-ul stațiilor - **double tagging** posibil (cursul 12). Fără `switchport trunk allowed vlan` - trunk-ul transportă toate cele 4094 de VLAN-uri. `transport input all` - Telnet este permis, deci parolele circulă în clar (cursul 8). `enable password` în loc de `enable secret` - parola este stocată reversibil. Parola `cisco`, la ambele - spartă instantaneu de orice dicționar (cursul 12). Fără `access-class` pe liniile VTY - oricine din rețea poate încerca să se conecteze. Fără port security, fără DHCP snooping, fără DAI - CAM overflow, rogue DHCP și ARP poisoning sunt toate posibile. Fără `portfast` și `bpduguard` - atac STP posibil de pe orice port. Porturile nefolosite active și în VLAN 1 - oricine intră în priză are acces imediat la rețea. Numele implicit `Switch` și lipsa oricărui banner - semne că echipamentul nu a fost niciodată configurat conștient. Pentru ordonarea după gravitate, gândiți-vă la trei criterii: cât de ușor este de exploatat, ce obține atacatorul, și cât de repede ar fi observat. Parola slabă combinată cu Telnet permis este, probabil, mai gravă decât lipsa DAI - pentru că duce direct la control total asupra echipamentului, fără ca cineva să afle.

### 19.14 Ce se predă

#### Întrebări de verificare

1. De ce Dynamic ARP Inspection are nevoie de DHCP snooping activ?
  - a) Pentru ca ambele ruleaza pe acelasi procesor

Tabela 19.6 Ce se predă.

| Livrabil                                                                | Format   | Punctaj |
|-------------------------------------------------------------------------|----------|---------|
| Fișier Packet Tracer complet securizat, cu rețeaua wireless funcțională | .pkt     | 30 %    |
| Documentarea celor trei atacuri: mecanism, efect observat, contramăsură | document | 25 %    |
| Lista de verificare pentru securizarea unui switch, cu justificări      | document | 15 %    |
| Comparația WPA2-PSK / WPA2-Enterprise, cu argumente proprii             | document | 10 %    |
| Raportul de audit din provocare                                         | document | 20 %    |

- b) Pentru ca DAI verifica mesajele ARP fata de evidenta de perechi IP-MAC construita de DHCP snooping
  - c) Pentru ca DHCP snooping cripteaza ARP-ul
  - d) Nu are nevoie, sunt independente
2. O statie configurata static nu mai comunica dupa activarea DAI. De ce?
- a) DAI nu suporta adrese statice
  - b) Statia nu apare in evidenta DHCP snooping, deci mesajele ei ARP sunt aruncate - trebuie declarata printr-un ARP ACL
  - c) Adresa MAC este gresita
  - d) Portul trebuie sa fie trust
3. Ce face BPDU Guard?
- a) Impiedica un port sa devina port radacina
  - b) Inchide portul daca primeste un BPDU
  - c) Filtreaza toate BPDU-urile
  - d) Accelereaza convergenta STP
4. De ce port security cu maximum 1 poate crea probleme intr-o retea de birou?
- a) Consuma prea multa memorie
  - b) Rupe scenariile normale cu telefon IP plus calculator, sau cu masini virtuale
  - c) Nu functioneaza pe porturi de acces

- d) Necesita DHCP snooping
5. Ce port se declara trusted la DHCP snooping?
- a) Toate porturile de utilizator
  - b) Doar portul spre serverul DHCP legitim, de obicei uplink-ul
  - c) Niciunul
  - d) Portul atacatorului, ca sa fie monitorizat
6. Care este diferenta practica esentiala dintre WPA2-PSK si WPA2-Enterprise?
- a) Viteza de criptare
  - b) In Enterprise fiecare utilizator are credentiale proprii, deci poate fi revocat individual si apare in jurnale
  - c) PSK nu foloseste AES
  - d) Enterprise functioneaza doar pe 5 GHz



# Listă de notații

---

|                            |                                                                  |
|----------------------------|------------------------------------------------------------------|
| <b>568A / 568B</b>         | cele două ordini standard ale firelor în mufa RJ-45              |
| <b>6to4</b>                | tunel automat IPv6-peste-IPv4, prefix 2002::/16                  |
| <b>802.1Q</b>              | standardul de etichetare a cadrelor pe trunk                     |
| <b>802.1X / RADIUS</b>     | autentificare individuală, modul Enterprise                      |
| <b>ABR / ASBR</b>          | ruter de frontieră de arie / de sistem autonom                   |
| <b>access-class</b>        | comanda care leagă o listă de liniile VTY                        |
| <b>access-group</b>        | comanda care leagă o listă de o interfață și o direcție          |
| <b>ACE</b>                 | o singură regulă din listă: un criteriu plus o acțiune           |
| <b>ACL</b>                 | listă ordonată de condiții pentru identificarea traficului       |
| <b>adiacență</b>           | relația stabilită între doi vecini OSPF                          |
| <b>adresă de broadcast</b> | toți biții de host pe 1; nu se atribuie                          |
| <b>adresă de rețea</b>     | toți biții de host pe 0; nu se atribuie                          |
| <b>adresă ierarhică</b>    | adresă din care se poate deduce direcția, fără destinația exactă |

|                        |                                                           |
|------------------------|-----------------------------------------------------------|
| <b>adresă MAC</b>      | 48 de biți, unici pe planetă, scriși în ROM-ul plăcii     |
| <b>APIPA</b>           | 169.254.0.0/16 - semnul că DHCP nu a răspuns              |
| <b>arie</b>            | subdiviziune a domeniului OSPF, cu Dijkstra propriu       |
| <b>ARP</b>             | traduce adresa IP în adresă MAC                           |
| <b>ARP poisoning</b>   | răspunsuri ARP false, care redirectionează traficul       |
| <b>baud</b>            | simboluri pe secundă; nu e același lucru cu bps           |
| <b>beacon</b>          | cadru periodic prin care AP-ul își anunță existența       |
| <b>bit</b>             | cea mai mică unitate de informație: 0 sau 1               |
| <b>BPDU</b>            | mesajul prin care switch-urile schimbă informații STP     |
| <b>BPDU Guard</b>      | închide portul dacă de acolo apare un BPDU                |
| <b>Bridge ID</b>       | prioritate (16 biți) + adresă MAC (48 biți)               |
| <b>BSS / ESS</b>       | o celulă / mai multe celule cu același SSID               |
| <b>BSSID</b>           | adresa MAC a punctului de acces                           |
| <b>buget optic</b>     | puterea emisă minus toate pierderile de pe traseu         |
| <b>cadru (frame)</b>   | unitatea de date a nivelului 2                            |
| <b>CAM overflow</b>    | umplerea tabelului switch-ului cu adrese false            |
| <b>categorie</b>       | banda de frecvențe garantată a unui cablu torsadat        |
| <b>CGNAT</b>           | al doilea nivel de NAT, în rețeaua furnizorului           |
| <b>CIDR</b>            | notația /n; numărul de biți de 1 din mască                |
| <b>codare de linie</b> | regula prin care biții devin trepte de tensiune           |
| <b>conntrack</b>       | tabela de conexiuni a nucleului Linux                     |
| <b>constelație</b>     | harta simbolurilor unei modulații, în amplitudine și fază |
| <b>CSMA/CA</b>         | evitarea coliziunilor, în loc de detectarea lor           |
| <b>CSMA/CD</b>         | protocolul de acces la un mediu partajat                  |

|                                |                                                            |
|--------------------------------|------------------------------------------------------------|
| <b>DAI</b>                     | verificarea mesajelor ARP față de acea evidență            |
| <b>dB</b>                      | raport exprimat logaritmice; pierderile se adună           |
| <b>deny any implicit</b>       | regula invizibilă de la sfârșitul oricărei liste           |
| <b>designated port</b>         | portul care servește un segment, unul per legătură         |
| <b>DHCP relay</b>              | transformă broadcast-ul DHCP în unicast spre alt segment   |
| <b>DHCP snooping</b>           | evidența perechilor port–MAC–IP, din tranzacții DHCP       |
| <b>diafonie</b>                | semnal care se strecoară dintr-o pereche în alta           |
| <b>Diffie-Hellman</b>          | construirea unei chei comune fără a o transmite            |
| <b>Dijkstra / SPF</b>          | algoritmul drumurilor minime                               |
| <b>dispersie modală</b>        | lățirea impulsului din cauza traseelor de lungimi diferite |
| <b>distance-vector</b>         | știe doar direcția și distanța, de la vecini               |
| <b>distanță administrativă</b> | cât de multă încredere are ruterul în sursa rutei          |
| <b>DMZ</b>                     | zonă separată pentru serviciile expuse public              |
| <b>DNAT</b>                    | Destination NAT - rescrie destinația, în PREROUTING        |
| <b>domeniu de broadcast</b>    | grupul care aude un anunț general                          |
| <b>domeniu de coliziune</b>    | grupul în care două transmisii se pot ciocni               |
| <b>DORA</b>                    | Discover, Offer, Request, Acknowledgment                   |
| <b>double tagging</b>          | cadru cu două etichete, care exploatează VLAN-ul nativ     |
| <b>DR / BDR</b>                | ruterul desemnat și rezerva lui, pe segment multiacces     |



|                             |                                                               |
|-----------------------------|---------------------------------------------------------------|
| <b>DROTHER</b>              | ruter care nu e nici DR, nici BDR                             |
| <b>DSSS / OFDM</b>          | tehnici de împrăștiere a semnalului                           |
| <b>DTP</b>                  | protocolul de negociere automată a trunk-urilor               |
| <b>established</b>          | opțiune care prinde doar pachetele TCP cu ACK sau RST         |
| <b>EtherChannel</b>         | mai multe legături fizice văzute ca una logică                |
| <b>EtherType</b>            | ce protocol de nivel 3 este încapsulat                        |
| <b>extinsă</b>              | listă care filtrează după sursă, destinație, protocol și port |
| <b>FCS / CRC</b>            | sumă de control; detectează, nu corectează                    |
| <b>FDM / TDM / WDM</b>      | partajarea mediului pe frecvență, timp sau lungime de undă    |
| <b>flood</b>                | trimiterea pe toate porturile în afară de cel de intrare      |
| <b>flooding</b>             | retransmiterea unui LSP către toți ceilalți vecini            |
| <b>furtună de broadcast</b> | multiplicarea infinită a unui cadru într-o buclă              |
| <b>gateway</b>              | ruterul care marchează granița rețelei locale                 |
| <b>gratuitous ARP</b>       | răspuns ARP trimis fără să fi fost cerut                      |
| <b>GRE</b>                  | tunel generic de nivel 3, fără criptare                       |
| <b>Hello</b>                | mesaj periodic de descoperire și keep-alive                   |
| <b>IDS / IPS</b>            | detectare în paralel / prevenție în serie                     |
| <b>IGP / EGP</b>            | protocol din interiorul / dintre sisteme autonome             |
| <b>inside global</b>        | adresa publică sub care apare ea în Internet                  |
| <b>inside local</b>         | adresa privată reală a stației noastre                        |
| <b>interfață</b>            | un punct de conectare la rețea: placă, port, subinterfață     |
| <b>ISM</b>                  | bandă utilizabilă fără licență                                |
| <b>jitter</b>               | variația latenței; vine din așteptarea în cozi                |

**LAN / MAN / WAN**

rețele clasificate după acoperirea geografică

**lanț** punctul din traseul pachetului unde se aplică reguli**latență** timpul de la un capăt la altul; are patru componente**lease** durata pentru care o adresă e rezervată unui client**link-local** adresă FE80::/10, valabilă doar în segment**link-state** știe graful complet al topologiei**listă reflexivă** listă care își generează singură regulile de întoarcere**LLC / MAC** cele două subniveluri ale nivelului 2**longest prefix match**

regula după care cea mai specifică rută câștigă

**LSP / LSA** descrierea legăturilor unui ruter**MAC** hash calculat cu o cheie, care garantează integritatea**man-in-the-middle**

atacator care intermediază două sesiuni separate

**mască de rețea**

separă partea de rețea de partea de host

**MASQUERADE**

SNAT cu adresa curentă a interfeței de ieșire

**media convertor**

traduce între medii: cupru ↔ fibră ↔ wireless

**mediu ghidat** semnalul urmează un traseu fizic: cablu sau fibră**metrică** calitatea unei rute, măsurată de protocolul care a produs-o**MIMO** fluxuri spațiale independente, pe mai multe antene**modulație** transportarea datelor prin modificarea unei purtătoare**MSS** dimensiunea maximă a unui segment, negociată la deschidere**MTU** dimensiunea maximă a încărcăturii utile: 1500 octeți**NAT** rescrierea adreselor la trecerea printr-un ruter**NAV** contor de ocupare a mediului, dedus din câmpul Durată

|                                |                                                                    |
|--------------------------------|--------------------------------------------------------------------|
| <b>next-hop</b>                | adresa IP a următorului ruter de pe traseu                         |
| <b>Null0</b>                   | interfața care aruncă tot ce primește                              |
| <b>octet (byte)</b>            | grup de 8 biți; valori de la 0 la 255                              |
| <b>OFDMA</b>                   | subpurtaătoare alocate mai multor utilizatori simultan             |
| <b>OUI</b>                     | primii 24 de biți: identifică producătorul                         |
| <b>PAgP / LACP</b>             | protocoalele de negociere a agregării                              |
| <b>PAT / overload</b>          | NAT care translatează și portul, n : 1                             |
| <b>PDU</b>                     | unitatea de date a unui nivel: cadru, pachet, segment              |
| <b>PoE</b>                     | alimentare electrică transportată pe cablul de date                |
| <b>politică</b>                | acțiunea implicită a unui lanț, când nicio regulă nu se potrivește |
| <b>port</b>                    | număr de 16 biți care identifică aplicația                         |
| <b>port de acces</b>           | aparține unui singur VLAN; cadre neetichetate                      |
| <b>port security</b>           | limitarea numărului de adrese MAC per port                         |
| <b>port trunk</b>              | transportă mai multe VLAN-uri, etichetate 802.1Q                   |
| <b>preambul / SFD</b>          | sincronizarea receptorului și marcarea începutului                 |
| <b>protocol</b>                | set de reguli comune fără de care mesajul nu poate fi înțeles      |
| <b>proxy ARP</b>               | ruterul răspunde în locul unei destinații îndepărtate              |
| <b>rainbow table</b>           | lanțuri de hash-uri precalculate                                   |
| <b>reflexie totală internă</b> | fenomenul care ține lumina în miezul fibrei                        |
| <b>remark</b>                  | comentariu în interiorul unei liste                                |
| <b>RFC 1918</b>                | cele trei blocuri de adrese private                                |
| <b>rogue DHCP</b>              | server DHCP neautorizat, care se anunță gateway                    |
| <b>Root Guard</b>              | împiedică un port să devină port rădăcină                          |
| <b>root port</b>               | portul cu costul minim spre rădăcină, unul per switch              |

|                       |                                                               |
|-----------------------|---------------------------------------------------------------|
| <b>Router ID</b>      | identificatorul unic al unui ruter în domeniul OSPF           |
| <b>RTS / CTS</b>      | rezervarea explicită a mediului înainte de transmisie         |
| <b>running-config</b> | configurația activă, din RAM; se pierde la repornire          |
| <b>runt frame</b>     | cadru sub 64 de octeți, rezultat dintr-o coliziune            |
| <b>rută</b>           | tripletul destinație, direcție, distanță                      |
| <b>rută flotantă</b>  | rută de rezervă, cu AD mărită deliberat                       |
| <b>rută implicită</b> | 0.0.0.0/0 - se potrivește cu orice, se aplică ultima          |
| <b>SAE</b>            | autentificarea din WPA3, rezistentă la atac offline           |
| <b>salt</b>           | segment aleator adăugat parolei înainte de hashing            |
| <b>SIFS / DIFS</b>    | interval scurt (prioritar) / lung (transmisie nouă)           |
| <b>SLAAC</b>          | autoconfigurarea adresei IPv6, fără DHCP                      |
| <b>slot time</b>      | timpul dus-întors pe cel mai lung segment                     |
| <b>SNAT</b>           | Source NAT - rescrie sursa, în POSTROUTING                    |
| <b>socket</b>         | perechea adresă IP + port                                     |
| <b>SSID</b>           | numele rețelei, ales de administrator                         |
| <b>standard</b>       | listă care filtrează doar după adresa sursă                   |
| <b>startup-config</b> | configurația din NVRAM, încărcată la pornire                  |
| <b>stateful</b>       | firewall care ține minte sesiunile deschise                   |
| <b>stație ascunsă</b> | două stații care nu se aud, dar se ciocnesc la AP             |
| <b>sticky</b>         | adrese învățate dinamic, scrise în configurație               |
| <b>STP / STA</b>      | protocolul, respectiv algoritmul, care elimină buclele logice |
| <b>subinterfață</b>   | diviziune logică a unei interfețe fizice de ruter             |
| <b>subnetare</b>      | împărțirea unui bloc în blocuri mai mici                      |
| <b>sumarizare</b>     | înlocuirea mai multor rute cu una singură, mai generală       |
| <b>SVI</b>            | interfață virtuală de VLAN pe un switch de nivel 3            |
| <b>SYN cookie</b>     | apărare care codifică starea în numărul de secvență           |

|                            |                                                            |
|----------------------------|------------------------------------------------------------|
| <b>SYN flood</b>           | atac care umple coada de conexiuni pe jumătate deschise    |
| <b>tabela CAM</b>          | asocierile MAC $\leftrightarrow$ port, construite automat  |
| <b>tabela de topologie</b> | harta ariei, identică pe toate ruterele                    |
| <b>TTL</b>                 | contorul care oprește bucelele de rutare                   |
| <b>tunelare</b>            | încapsularea unui protocol în altul, pentru a traversa     |
| <b>UTP / STP</b>           | cablu torsadat fără ecran, respectiv cu ecran              |
| <b>VLAN</b>                | domeniu de broadcast definit prin porturi, nu prin cablare |
| <b>VLAN hopping</b>        | ocolirea izolării dintre VLAN-uri                          |
| <b>VLAN ID</b>             | numărul VLAN-ului, pe 12 biți: 1–4094 utilizabile          |
| <b>VLAN nativ</b>          | VLAN-ul care circulă neetichetat pe un trunk               |
| <b>VLSM</b>                | subnetare cu măști de lungimi diferite                     |
| <b>wildcard</b>            | mască în care 0 înseamnă „fix” și 1 înseamnă „orice”       |
| <b>încapsulare</b>         | adăugarea, la fiecare nivel, a antetului propriu           |

# Anexă. Răspunsuri la întrebările de verificare

---

Litera indică varianta corectă; explicația arată de ce celelalte nu sunt.

## Capitolul 1. Transmisiuni analogice și digitale

1. **b.** Nivelul 3 lucreaza cu pachete si adrese IP. Cadrul apartine nivelului 2, segmentul nivelului 4.
2. **b.** Exact opusul este adevarat despre dimensiune: fiecare nivel adauga antet. Castigul este modularitatea si posibilitatea de a localiza problemele.
3. **b.** Tranzitia de la mijlocul fiecarui interval permite receptorului sa isi refaca ceasul din semnalul de date, indiferent de continut. Costul este banda dubla.
4. **b.** Tabelul este construit astfel incat niciun cuvânt de cod sa nu contina mai mult de trei zerouri consecutive; cele 16 cuvinte ramase devin simboluri de control.
5. **c.** Regula este 2 la puterea n.  $2^6 = 64$ . Aceeasi regula va da, la cursul 5, numarul de adrese dintr-o subretea.
6. **b.** Switch-ul creeaza cate un domeniu de coliziune pentru fiecare port, dar retransmite broadcast-urile pe toate porturile, deci domeniul de broadcast ramane unul singur.
7. **b.** QPSK are 4 puncte de constelatie, deci  $\log_2(4) = 2$  biti per simbol:  $2 \times 4800 = 9600$  bps.
8. **b.** Nivelul Internet din TCP/IP corespunde nivelului 3 din OSI: adresare globala si alegerea caii, adica exact ce face IP.

9. **c.** Portul identifica aplicatia. Adresa IP duce pachetul pana la calculator; portul il duce mai departe, pana la programul care il asteapta.
10. **b.** Transformatoarele din placile de retea nu transmit componenta continua. Un semnal care sta mult timp la acelasi nivel pur si simplu nu ajunge la celalalt capat.
11. **c.**  $128 + 64 = 192$ . Este exact primul octet al retelelor 192.168.x.x, pe care le veti vedea la fiecare laborator.
12. **b.** Conversia se face pe grupe de patru biti, fara niciun calcul. Reprezentarea este mult mai scurta decat binarul si mult mai comoda decat zecimalul.

## Capitolul 2. Medii de transmisie

1. **b.** Transmisia este diferentia. Daca perturbatia afecteaza identic ambele fire, diferenta ramane neschimbata si zgomotul se anuleaza singur.
2. **b.** Doua echipamente de acelasi nivel transmit pe aceiasi pini si asculta pe aceiasi pini. Cablul crossover inverseaza perechile si repara situatia.
3. **b.** Miezul de circa 9 micrometri permite un singur mod de propagare, deci impulsurile nu se latesc din cauza traseelor de lungimi diferite.
4. **c.** Propagarea pe 100 m este sub o microsecunda - neglijabila. Domina transmisia cadrului si procesarea din echipamente.
5. **b.** WDM este FDM aplicat domeniului optic: fiecare flux primeste alta culoare, iar prisme optice le separa la capat.
6. **b.** Limita de 100 m vine din bugetul de atenuare si din timpul de propagare, nu din categoria cablului. Cat 8 nu schimba nimic.
7. **b.** Decibelul este un raport logaritmic: -3 dB inseamna aproximativ jumatate din putere, -10 dB a zecea parte, -20 dB a suta parte.
8. **b.** Imunitatea electrica a fibrei, care este marele ei avantaj, este in acelasi timp motivul pentru care nu poate duce PoE.

## Capitolul 3. Nivelul legătură de date

1. **b.** Operatia se numeste flood si este singurul mod prin care switch-ul poate garanta ca destinatia primeste cadrul. Raspunsul destinatiei il va invata apoi pe portul corect.
2. **b.** 64 de octeti = 512 bit times = slot time la 10 si 100 Mbps. Statia trebuie sa inca transmita cand coliziunea se intoarce la ea.
3. **c.** Bitul cel mai putin semnificativ din primul octet indica multicast. 01 este impar, deci multicast; FF:FF:... este broadcast, un caz particular.

4. **c.** Switch-ul extinde domeniul de broadcast: toate cele 24 de porturi fac parte din acelasi domeniu. Separarea cere un ruter sau VLAN-uri.
5. **b.** Peste 0x0600 campul se interpreteaza ca EtherType. 0x0800 = IPv4, 0x0806 = ARP, 0x86DD = IPv6, 0x8100 = cadru etichetat 802.1Q.
6. **b.** Adresa sursa este singura care spune sigur de unde a venit cadrul, deci pe ce port se afla acea statie.
7. **b.** Coliziunea presupune un mediu partajat. Cu o legatura punct-la-punct si fire separate pe fiecare sens, nu exista fizic conditia de coliziune.
8. **b.** Cut-through decide dupa primii octeti, deci nu apuca sa verifice FCS-ul. Cadrul corupt este propagat si va fi aruncat abia la destinatie - dupa ce a consumat banda.

#### Capitolul 4. Optimizarea rețelelor locale

1. **c.** Campul are 12 biti, deci 4096 de valori; 0 si 4095 sunt rezervate, deci raman 4094 utilizabile.
2. **b.** Aceasta este definitia VLAN-ului nativ si, in acelasi timp, baza atacului de tip double tagging din cursul 12.
3. **b.** Un pachet IP prins intr-o bucla este aruncat cand TTL ajunge la zero. Un cadru Ethernet nu are aceasta protectie si circula la infinit, multiplicandu-se.
4. **c.** Portul radacina este calea spre root bridge; radacina nu are nevoie de o cale spre sine, deci toate porturile ei sunt desemnate.
5. **b.** Modul passive doar raspunde la pachetele LACP primite. Daca niciun capat nu initiaza, negocierea nu incepe niciodata.
6. **b.** Bridge ID-ul este prioritate urmata de adresa MAC. Prioritatea se compara prima; adresa MAC decide doar la egalitate.
7. **b.** Fara aceasta stare, portul ar trece in forwarding cu tabela goala si ar inunda tot traficul exact in momentul cel mai delicat.
8. **c.** Switch-ul trimite toate cadrele unei conversatii pe aceeasi legatura, ca sa nu se dezordoneze. Banda agregata ajuta la multe conversatii simultane, nu la una singura.

#### Capitolul 5. Protocolul IP

1. **b.** Cu /26 raman 6 biti de host; pusi toti pe 1 dau 63. Reteaua este .0, broadcast-ul .63, iar urmatoarea subretea incepe la .64.
2. **b.** Daca alocati intai blocuri mici, alinierea se pierde si urmatorul bloc mare nu mai poate incepe decat mult mai departe. Spatiul dintre ele devine inutilizabil.



3. **b.** Cadrul trebuie sa ajunga fizic la ruter, deci ii trebuie adresa MAC a ruterului. Adresa IP destinatie ramane cea finala si nu se schimba.
4. **b.** Adresele IP sunt end-to-end si raman constante; adresele MAC sunt locale segmentului si se rescriu la fiecare salt.
5. **b.** 169.254.0.0/16 este blocul APIPA. Problema este in calea spre server: relay lipsa, VLAN gresit sau rezervor epuizat.
6. **b.** Fiecare server care a raspuns a rezervat o adresa. Fara acest broadcast, adresele respinse ar ramane blocate pana la expirare.
7. **b.**  $2^3 = 8$  adrese totale, minus adresa de retea si cea de broadcast, raman 6 utilizabile.
8. **b.** La zero, pachetul este aruncat si se trimite un ICMP inapoi la sursa. Pe acest mecanism se bazeaza si traceroute.

## Capitolul 6. Rutare

1. **b.** Prima valoare este distanta administrativa, deci sursa rutei - 110 inseamna OSPF. A doua este metrica specifica protocolului.
2. **b.** Exact acesta este mecanismul rutei flotante: ii dam deliberat o AD mai mare decat a protocolului dinamic, ca sa ramana in rezerva.
3. **b.** Cea mai lunga potrivire de prefix este criteriul suprem, aplicat inaintea distantei administrative si a metricii.
4. **b.** Pe un mediu multiacces exista mai multi destinatari posibili. Fara next-hop, ruterul nu poate face ARP si nu poate completa cadrul.
5. **b.** In binar, al doilea octet este 000001xx: primii 14 biti sunt comuni, deci prefixul sumarizat este /14.
6. **b.** Ruta spre Null0 este mai specifica decat cea sumarizata, deci castiga si opreste traficul chiar la ruterul care sumarizeaza.
7. **c.** running-config traieste in RAM. Fara copy running-config startup-config, toata munca dispare la repornire.
8. **b.** Fiecare ruter decide independent, pe baza propriei tabele. Pachetul poate ajunge perfect, dar raspunsul nu are pe unde sa se intoarca.

## Capitolul 7. Liste de acces

1. **b.** Orice ACL se termina cu un deny any invizibil. O lista fara nicio regula permit blocheaza tot.

2. **b.** Lista standard filtreaza doar dupa sursa. Plasata langa sursa, ar bloca traficul acelei statii spre orice destinatie, nu doar spre cea vizata.
3. **b.** Ultimii doi biti sunt ignorati, deci se acopera patru adrese consecutive incepand de la .4: .4, .5, .6 si .7.
4. **a.** /26 lasa 6 biti liberi, deci  $2^6 = 64$  adrese. Wildcard-ul este 64 minus 1, adica 0.0.0.63. Sau: 255.255.255.255 minus masca 255.255.255.192.
5. **b.** Regula este: o lista per interfata, per protocol rutat, per directie. Deci doua pentru IPv4 - una inbound si una outbound.
6. **b.** Este cel mai simplu mod de a permite traficul de intoarcere fara a permite initierea conexiunilor din exterior. Nu functioneaza pentru UDP sau ICMP.
7. **b.** Zero potriviri inseamna ca traficul nu ajunge la regula: fie lista e pe alta interfata sau directie, fie o regula anterioara a facut deja match.
8. **b.** access-class aplicat pe line vty filtreaza sesiunile de administrare, indiferent pe ce interfata sosesc. transport input ssh doar interzice Telnet, dar nu limiteaza sursa.
9. **b.** La o lista numerotata, no access-list 254 sterge toata lista, iar regulile noi se adauga obligatoriu la sfarsit. Numele plus numerele de secventa rezolva ambele probleme.

## Capitolul 8. Securizarea rețelei

1. **b.** INPUT trateaza traficul destinat masinii; FORWARD trateaza traficul care doar o tranziteaza. Confuzia dintre ele este cea mai frecventa cauza a regulilor fara efect.
2. **b.** Conexiunea TCP este full-duplex: sunt doua fluxuri independente, care se inchid separat. Serverul confirma FIN-ul clientului, dar poate continua sa trimita date pana cand trimite propriul FIN.
3. **b.** Fiecare SYN falsificat consuma o intrare in coada de conexiuni pe jumatate deschise, pana la expirarea temporizatorului. Atacatorul nu alocă nimic de partea lui.
4. **b.** DH rezolva exclusiv problema stabilirii cheii. Impotriva unui atacator plasat in mijloc nu ajuta deloc - acolo e nevoie de autentificare, adica de verificarea ampretei cheii.
5. **b.** Un firewall stateless judeca fiecare pachet izolat. Unul stateful mentine o tabela de conexiuni, deci stie ca un pachet de raspuns apartine unei sesiuni initiate din interior.
6. **b.** Politica DROP data prima blocheaza instantaneu si sesiunea dumneavoastra SSH. Regulile permissive trebuie sa existe deja cand politica intra in vigoare.
7. **b.** LOG este singura tinta netterminala. De aceea se pune imediat deasupra regulii de DROP pe care o documenteaza, cu exact aceleasi conditii.

8. **b.** O scanare de porturi împotriva unei mașini care face DROP durează de zeci de ori mai mult și nu oferă confirmări. În interiorul rețelei, însă, REJECT e preferabil, ca aplicațiile să primească eroare imediat.
9. **b.** Criptografia asimetrică este folosită doar la autentificare și la stabilirea cheii. Traficul propriu-zis merge pe cheia simetrică. Același principiu stă la baza HTTPS.

## Capitolul 9. NAT și tunelare

1. **b.** Tabela NAT reține perechea adresă:port internă și perechea adresă:port publică. Portul distinge conversațiile, deci limita practică este numărul de porturi disponibile.
2. **b.** Dacă destinația s-ar rescrie după decizia de rutare, routerul ar fi decis deja ca pachetul e pentru el. De aceea DNAT se face în PREROUTING, iar SNAT în POSTROUTING.
3. **b.** Inside = gazda e a noastră. Global = așa este văzută din Internet. Deci: mască publică a unei stații interne, adică rezultatul translatații.
4. **b.** Este pasul uitat cel mai des. Fără marcarea interfețelor, routerul nu știe în ce direcție să translateze și nu creează nicio intrare.
5. **b.** IP în Ethernet este stratificare normală. IPv6 în IPv4, sau Ethernet în IP, este tunelare. Testul rapid: apar două antete de același tip în același pachet.
6. **b.** Este simptomul clasic: pachetele mici trec, cele mari nu. Se rezolvă cu ip mtu și ip tcp adjust-mss pe interfața de tunel.
7. **b.** Prefixul 6to4 este 2002::/16, urmat de cei 32 de biți ai adresei IPv4 scrise hexazecimal: 141 = 8D, 85 = 55, 200 = C8, 19 = 13.
8. **b.** Un tunel protejează exact porțiunea pe care o traversează. După capatul tunelului, traficul continuă în forma lui originală, în clar.
9. **b.** Cu un singur :: numărul grupurilor nule se deduce prin scădere din opt. Cu două, ar exista mai multe interpretări valide ale aceleiași scrieri.

## Capitolul 10. Rutare dinamică

1. **b.** Tabela de topologie este harta completă. Dijkstra rulează peste ea și abia rezultatele - cele mai bune cai - ajung în tabela de rutare.
2. **b.** Fără o valoare finită pentru infinit, o buclă de rutare ar crește metrica la nesfârșit. Limita de 15 este pretul plătit ca buclă să se termine repede.
3. **b.** Este cel mai simplu mecanism antibuclă la distance-vector: dacă R2 a aflat despre X de la R1, nu îi mai spune lui R1 că știe drumul spre X.
4. **b.** Fără DR, fiecare router ar forma adiacență cu fiecare. Cu DR și BDR, toate celelalte formează adiacență doar cu acestea două.

5. **b.** Rolul se schimba doar daca DR-ul curent cade sau daca procesul OSPF este resetat. Este un comportament deliberat, ca sa se evite reorganizari inutile.
6. **b.** Ex-Start si Exchange sunt etapele de sincronizare a bazelor de date, iar OSPF refuza sa continue daca MTU-urile anuntate nu coincid. Numerele de proces, in schimb, sunt locale si pot diferi linistit.
7. **b.** Costul minim este 1, deci orice legatura de 100 Mbps sau mai rapida primeste aceeasi valoare. De aceea in retelele moderne se ajusteaza reference-bandwidth, cu aceeasi valoare pe toate ruterele.
8. **b.** Toate ariile trebuie conectate la aria 0, iar traficul inter-arie o traverseaza obligatoriu. Cand conexiunea fizica lipseste, se foloseste o legatura virtuala sau un tunel.
9. **b.** network 10.0.0.0 0.0.0.3 area 0 selecteaza interfetele din 10.0.0.0/30. Este o sursa clasica de erori, pentru ca in restul configuratiei Cisco se foloseste masca normala.

## Capitolul 11. Rețele wireless

1. **b.** Ambele motive sunt fizice si nu pot fi ocolite. Daca nu putem detecta coliziunile, ramane sa le prevenim - de aici Collision Avoidance.
2. **b.** Este ascultarea virtuala a mediului: o statie poate sti ca mediul e ocupat fara sa auda efectiv transmisia, doar pentru ca a auzit anuntul.
3. **b.** CTS-ul emis de punctul de acces ajunge la toata lumea din raza lui si le face sa isi seteze NAV-ul, deci sa taca pe durata anuntata.
4. **b.** Daca toate statiile ar transmite imediat ce mediul se elibereaza, s-ar ciocni sistematic. Numarul aleator le desincronizeaza.
5. **b.** Doar 1, 6 si 11 nu se suprapun. Suprapunerea partiala este mai grava decat folosirea aceluiasi canal: pe acelasi canal retelele se aud si isi impart mediul, in timp ce o suprapunere partiala este doar zgomot.
6. **b.** Este suficient sa astepti ca un client legitim sa se conecteze. In plus, dispozitivele configurate pentru o retea ascunsa isi striga numele oriunde s-ar afla.
7. **b.** SSID-ul comun face ca roaming-ul sa fie invizibil pentru client; BSSID-ul, care este adresa MAC a AP-ului, ramane distinct si permite retelei sa urmareasca trecerea.
8. **b.** Nu algoritmul a fost problema, ci modul in care a fost folosit: IV prea scurt, transmis in clar, si posibilitatea de a injecta trafic pentru a-l epuiza rapid.
9. **b.** Cu WPA2-PSK, cine capteaza handshake-ul poate incerca offline milioane de parole. Cu SAE, fiecare incercare cere interactiune cu retea, deci devine detectabila si limitabila.

## Capitolul 12. Securitatea LAN și atacuri de rețea

1. **b.** Switch-ul invata exclusiv din adresa MAC sursa. Falsificand-o la fiecare cadru, tabela se umple cu intrari inutile si switch-ul incepe sa inunde tot traficul.
2. **b.** Nimic nu se strica: traficul ajunge unde trebuie, dar si oriunde altundeva. De aceea atacul e greu de observat fara sa te uiti la show mac address-table count.
3. **b.** Lipsa autentificarii este proiectata in protocol. Contramasura nu repara ARP, ci adauga in switch o verificare externa: DHCP snooping plus Dynamic ARP Inspection.
4. **b.** De aceea cele doua se activeaza impreuna. Statiile cu adresa statica, care nu apar in acea evidenta, trebuie adaugate manual cu arp access-list.
5. **b.** Primul switch scoate eticheta exterioara tocmai pentru ca este VLAN-ul nativ. De aceea recomandarea este ca VLAN-ul nativ sa fie unul nefolosit.
6. **b.** Trucul functioneaza doar la dus, prin scoaterea etichetei native. La intors nu exista niciun mecanism echivalent, deci atacul serveste doar la injectarea de trafic.
7. **b.** PortFast grabeste trecerea in forwarding, dar nu protejeaza. BPDU Guard inchide portul daca de acolo apare un BPDU - adica exact daca cineva incearca sa participe la STP.
8. **b.** Salt-ul nu previne atacul, ci ii schimba economia: in loc de o singura tabela pentru toata baza de date, e nevoie de cate una per cont. O parola slaba ramane slaba.
9. **b.** Traficul intre doua statii din acelasi VLAN este comutat la nivelul 2 si nu ajunge niciodata la un ruter. De aici necesitatea masurilor de securitate chiar pe switch.

## Lucrarea 13. Primii pași: topologie, adresare și primul ping

1. **b.** Interfetele unui ruter sunt inchise implicit, spre deosebire de cele ale unui switch. no shutdown le activeaza.
2. **b.** Rezolutia ARP se face inainte de trimiterea primului cadru; daca dureaza mai mult decat timpul de asteptare al ping-ului, primul pachet se pierde. Urmatoarele merg, pentru ca adresa e in cache.
3. **b.** Adresa IP destinatie ramane cea a serverului, dar adresa MAC destinatie este a ruterului. Statia nu are motiv sa cunoasca MAC-ul unei masini din alta retea.
4. **b.** Adresa IP spune unde vrea sa ajunga pachetul si nu se modifica. Adresa MAC spune cui i se preda acum cadrul si se rescrie la fiecare salt.
5. **b.** Prima spune daca exista semnal pe fir; a doua daca protocolul de linie functioneaza. up/down inseamna cablu bun, dar protocol nepotrivit.
6. **b.** Fara gateway, orice destinatie din afara propriei retele este de neatins. In interiorul retelei, comunicatia este directa si nu are nevoie de ruter.

**Lucrarea 14. Comutarea în LAN: tabela MAC, ARP, domenii**

1. **b.** Switch-ul învață din adresa MAC sursa a fiecărui cadru care trece. Cererea ARP îl învață pe emitator, răspunsul ARP pe destinatar.
2. **c.** Fiecare port formează, împreună cu stația conectată, un domeniu de coliziune propriu. VLAN-urile influențează domeniile de broadcast, nu pe cele de coliziune.
3. **c.** Cele două stații se află pe același segment și se aud direct; retransmiterea cadrului ar fi inutilă și ar putea produce duplicate.
4. **b.** Switch-ul vede un singur fir spre hub. Tot ce se află dincolo de el pare, din punctul lui de vedere, conectat la același port.
5. **b.** Adresa este învățată dinamic la prima conectare și apoi tratată ca și cum ar fi fost configurată manual - comod, dar de verificat înainte de a salva configurația.
6. **b.** Full-duplex presupune că fiecare sens are calea lui și că nu există coliziuni. Un mediu partajat impune half-duplex și CSMA/CD.

**Lucrarea 15. VLAN-uri, trunk-uri și rutare inter-VLAN**

1. **b.** Din punct de vedere al nivelului 2 este ca și cum ar fi conectate la switch-uri complet separate. Ca să comunice, au nevoie de un dispozitiv de nivel 3.
2. **b.** Eticheta conține TPID 0x8100 și numărul VLAN-ului. Excepția este VLAN-ul nativ, care circula netichetat.
3. **b.** Un atacator dintr-un VLAN nativ care coincide cu VLAN-ul lui poate injecta cadre cu două etichete și poate ajunge în alt VLAN.
4. **b.** Subinterfețele nu au stare proprie: dacă interfața fizică este închisă administrativ, toate mor odată cu ea.
5. **b.** Cele două cauze clasice: VLAN-ul nu este permis pe trunk, sau nu există deloc pe switch-ul din mijloc.
6. **b.** Nu este nimic magic: alegerea rădăcinii se face după prioritate, iar la egalitate după adresa MAC. O prioritate mai mică câștigă.

**Lucrarea 16. Proiectarea adreselor: VLSM și DHCP**

1. **a.** /26 oferă 62 de adrese utilizabile: 60 de stații plus gateway înseamnă 61, deci încăpe. Un /27 ar oferi doar 30 și nu ar ajunge.
2. **b.** Un /23 trebuie să înceapă la o adresă multiplu de 512. Dacă ați alocat deja câteva /29-uri la început, următorul aliniament posibil e mult mai departe și spațiul dintre ele se pierde.

3. **b.** 169.254.0.0/16 este blocul APIPA. Problema este în calea spre server: relay lipsa, VLAN gresit sau rezervor epuizat.
4. **b.** Sursa este 0.0.0.0 și destinația 255.255.255.255. Fiind broadcast limitat, mesajul nu trece prin rutare - de aici necesitatea relay-ului.
5. **b.** Ruterul preia cererea, o trimite unicast serverului și atasează informația despre rețeaua de origine, ca serverul să știe din ce rezervor să aloce.
6. **b.** /23 înseamnă 23 de biți de rețea: 255.255.254.0. Este cea mai frecventă greșeală de la această lucrare, și are simptomul perfid că jumătate din stații răspund.

### **Lucrarea 17. Rutare: statică, rute flotante și OSPF**

1. **b.** Prima valoare este distanța administrativă, deci sursa rutei - 110 înseamnă OSPF. A doua este metrica specifică protocolului.
2. **b.** Exact acesta este mecanismul rutei flotante: îi dăm deliberat o AD mai mare decât a protocolului dinamic, ca să rămână în rezervă.
3. **b.** Pentru un /24 se scrie 0.0.0.255, pentru un /30 se scrie 0.0.0.3. Este o sursă frecventă de erori pentru cine vine de la rutele statice.
4. **b.** O nepotrivire de MTU împiedică schimbul complet al bazelor de date. Este cauza clasică pentru blocarea în EXSTART sau EXCHANGE.
5. **b.** Ruta statică dispare din tabelă doar când interfața trece în down. OSPF, în schimb, încearcă să vorbească periodic cu vecinul și observă când acesta tace.
6. **b.** Ordinea deciziei este: prefix cel mai lung, apoi AD, apoi metrica. AD-ul contează doar între rute cu același prefix.

### **Lucrarea 18. Liste de acces și NAT**

1. **b.** De aceea o listă standard se plasează cât mai aproape de destinație: pusă lângă sursă, ar bloca tot traficul acelei surse, indiferent unde merge.
2. **b.** Orice listă se termină cu deny any implicit. O listă formată numai din deny blochează absolut tot.
3. **b.** Zero potriviri înseamnă că traficul nu ajunge la regulă. Dacă ar ajunge și ar fi gresită, contorul ar crește, dar rezultatul ar fi altul.
4. **b.** O listă extinsă știe exact ce trebuie oprit, deci poate opri devreme fără efecte colaterale. Lista standard nu-și permite asta.
5. **b.** Cuvântul overload înseamnă PAT: mai multe adrese interne partajează o adresă publică, deosebite prin numărul de port.

6. **b.** Este exact comportamentul cerut de politica. Un test cu ping v-ar duce la concluzia gresita ca regula nu merge.

### **Lucrarea 19. Securitatea rețelei de acces și wireless**

1. **b.** DAI nu are de unde sti ce adresa IP ii apartine legitim fiecarui port. Evidenta construita de DHCP snooping este exact aceasta sursa de adevar.
2. **b.** Este cel mai frecvent efect secundar al activarii DAI. Solutia corecta este un arp access-list care declara explicit perechea IP-MAC, nu marcarea portului ca trusted.
3. **b.** Un port de utilizator nu are motiv sa primeasca BPDU-uri. Daca primeste, inseamna ca acolo s-a conectat un switch - legitim sau nu - si portul se inchide.
4. **b.** Un telefon IP cu PC in spate inseamna doua adrese MAC pe acelasi port. Valoarea 2 sau 3 este compromisul uzual.
5. **b.** Regula este simpla: doar pe unde vine serverul legitim. Orice port de utilizator ramane untrusted, iar raspunsurile DHCP de acolo sunt aruncate.
6. **b.** Cu o parola comuna, plecarea unui angajat ar cere schimbarea ei pentru toata lumea - deci, in practica, nu se schimba niciodata.