SÎRBU FLORICA CRISTINA

ÎNTRE PRACTICĂ ȘI TEORIE ARHITECTURA REȚELELOR DE CALCULATOARE

ISBN 978-973-0-42529-1
PLOIEȘTI
2025

CUPRINS

INTRODUCERE

În contextul prezent al dezvoltării reţelelor de calculatoare, este inutil să mai subliniem importanţa acestui domeniu.

Lucrarea de faţă se adresează în principal profesorilor de informatică, dar şi administratorilor de reţele din cadrul şcolilor. Sunt presupuse, din partea cititorului, cunoştinţe de bază de programare, precum şi privind funcţionarea sistemelor de operare.

Am ales această temă fiind una de larg interes, amplă din punct de vedere al volumului de informaţii

Lucrarea stiintifica este structurată pe 3 capitole:

Capitolul 1: Reţele de calculatoare – defineşte reţeaua, precum şi modelele şi nivelurile reţelei.

Capitolul 2: Aplicaţii în reţele – prezintă poşta electronică şi transferul fişierelor de reţea.

La sfârşitul fiecărui capitol este prezentată câte o aplicaţie astfel încât:

- în primul capitol este aplicaţia „*Simularea unei reţele de calculatoare cu ajutorul aplicaţiei Packet Tracer de la Cisco*";
- în cel de-al doilea capitol se află „Modelarea traficului pe o placă de reţea a unui PC cu ajutorul proceselor stocastice autosimilare";
- tot in cel de-al doilea capitol este descris softul educaţional "Simulare reţea".

Menţionez că toate figurile reprezintă sursă proprie, realizate cu ajutorul programului ChemSketch.

CAPITOLUL 1. REŢELE DE CALCULATOARE

1.1. NOŢIUNI GENERALE

Tot ceea ce are legătură într-un fel sau altul cu calculatoare are două caracteristici: se dezvoltă foarte repede şi este foarte complex. Reţelele de calculatoare nu fac excepţie.

Ca urmare, este extrem de uşor pentru oricine să se piardă în nenumăratele detalii în permanentă schimbare.

Considerăm că, în orice domeniu, o bună prezentare trebuie să pornească de la principiile de bază. Principiile de bază sunt (relativ) simple şi evoluează mult mai lent decât construcţiile tehnice elaborate pe baza lor.

Prin reţea de calculatoare înţelegem un sistem (constând din componente hard şi soft) care interconectează calculatoarele, permiţând unor programe ce se execută pe aceste calculatoare să comunice între ele.

O reţea de calculatoare reprezintă o colecţie de calculatoare autonome interconectate. Două calculatoare sunt considerate interconectate dacă pot realiza schimb de informaţii între ele. Un calculator este autonom dacă funcţionarea acestuia nu poate fi influenţată de un altul.

Definiţie: Reţeaua de calculatoare reprezintă un ansamblu de echipamente de calcul (regulatoare, elemente de execuţie, senzori/traductoare, calculatoare etc.) intinse geografic, interconectate prin intermediul unor medii de comunicaţie, asigurându-se în acest fel utilizarea în comun de către un număr mare de utilizatori a tuturor resurselor fizice (hardware), logice (software şi aplicaţii de bază) şi informaţionale (baze de date) de care dispune ansamblul de calculatoare conectate. O altă noţiune utilizată este lucrul în reţea care reprezintă conceptul de conectare a unor calculatoare care partajează resurse.

Resursele pot fi:

- date (baze de date);
- aplicaţii (Word, Excel, DVD player etc.);
- periferice (elemente de automatizare, imprimante, scannere etc.).

Orice calculator care face parte dintr-o reţea de calculatoare este numit nod sau gazdă (eng., host).

Pentru a putea comunica între ele, calculatoarele trebuie să respecte un anumit set de reguli, adică un protocol. În general, pentru comunicare sunt utilizate protocoale, de exemplu, în discuţia dintre două sau mai multe persoane, de regulă, o singură persoană vorbeşte la un moment dat.

Prin conectarea calculatoarelor se pot realiza o multitudine de activităţi:

- transferul datelor de la un calculator la altul;
- se poate accesa o bază de date existentă pe un alt calculator;
- se pot transmite mesaje;
- se pot utiliza resursele hardware (imprimante, plottere, scannere etc.);
- software ale unui alt calculator.

Toate aceste calculatoare legate între ele alcătuiesc o reţea.Avantajele utilizării reţelei de calculatoare:

- Comunicarea;
- Maximizarea eficienţei în transferurile de date;
- Scăderea costurilor în funcţionare;
- Protejarea simplă şi eficientă a datelor;
- Asigurarea disponibilităţii;
- Optimizarea supraîncărcării calculatorului;
- Optimizarea costurilor de întreţinere.

În zilele noastre, reţelele de calculatoare pot fi privite ca fiind:

- mulţime de calculatoare conectate prin linii de comunicaţii = cel mai simplu mod de a defini o reţea (calculatoare + linii de comunicaţii);
- infrastructură software/hardware care permite accesul partajat la resursele de calcul (calculatoare, fişiere, imprimante etc.);
- reţea pentru comunicaţii de date – un mediu prin care utilizatorii dispersaţi geografic pot comunica (poşta electronică, televiziune digitală, telefonie digitală, telefonie mobilă, teleconferinţe etc.).

1.2. REŢELE CALCULATOARE

1.2.1. Modele de reţele

Există mai multe criterii după care putem clasifica reţelele de calculatoare. După tipul tehnologiilor utilizate în transmisia datelor reţelele de calculatoare, se clasifică în[1]:

a) ***Reţele cu difuzare***

Acestea posedă un singur canal de comunicaţie partajat de toate calculatoarele din reţea. Atunci când un calculator trimite un mesaj unui altuia prin intermediul reţelei, mesajul va fi trimis tuturor nodurilor. Fiecare va cerceta la recepţionare dacă respectivul mesaj îi este destinat. În caz afirmativ conţinutul mesajului este citit, altfel este ignorat. Există şi posibilitatea ca un mesaj să fie trimis tuturor calculatoarelor din reţea (eng., broadcast) sau unui grup de calculatoare (eng., multicast).

b) ***Reţele punct-la-punct*** **(*point-to-point*).**

Un calculator dintr-o astfel de reţea trimite pachetul doar către calculatorul destinaţie, eventual folosind noduri intermediare. Pot exista mai multe drumuri dintre expeditor şi destinatar, iar alegerea acestuia se realizează cu ajutorul algoritmilor de rutare. În general, reţelele mai mici utilizează difuzarea de mesaje, iar cele de dimensiuni mai mari sunt de tip punct-la-punct.

Reţelele cu difuzare pot fi la rândul lor împărţite după modul de alocare a canalului de comunicaţie în *statice* şi *dinamice*. Un caz de alocare statică constă în folosirea de intervale de timp în care o gazdă poate emite mesaje. Astfel, fiecare trebuie să-şi aştepte rândul, însă, dacă un calculator nu are ce pachete să trimită, atunci când îi soseşte rândul, se iroseşte timpul alocat acestuia. De aceea, în practică, sunt utilizate preponderent alocările dinamice (la cerere) ale canalului de comunicaţie.

Metodele de alocare dinamică pot fi centralizate sau descentralizate. În cazul centralizat, există o entitate care stabileşte, după un anumit algoritm, care nod poate transmite date în reţea. Pentru cazul descentralizat, fiecare nod va trebui să stabilească când va transmite pachetele în reţea.

După întinderea geografică pe care o acoperă, reţele de calculatoare se împart în:

Reţele locale - LAN (Local Area Network). Acestea se întind într-un birou, o clădire, sau în câteva clădiri apropiate şi sunt de dimensiuni restrânse. Spre exemplu, acestea se întâlnesc în cadrul organizaţiilor şi sunt utilizate pentru partajarea informaţiilor şi a diverselor dispozitive (imprimante, scannere etc.)

[1] Ştefan-Ciprian Tanasă, Retele de calculatoare, Universitatea „Al. I. Cuza”, Iaşi, 2006

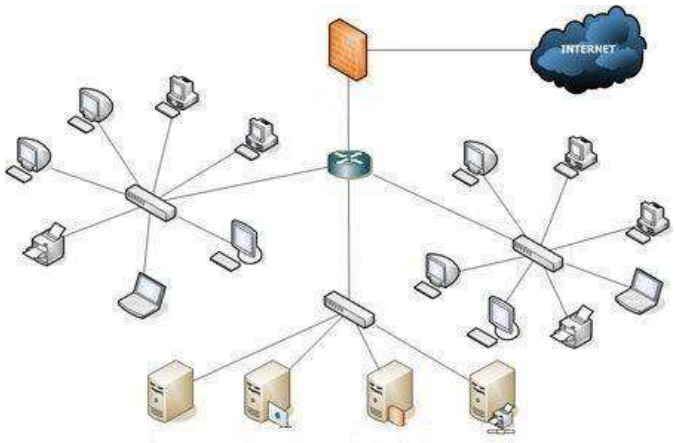


Fig. 1. Reţea locală – LAN

Reţele metropolitane - MAN (Metropolitan Area Network) se întind, de regulă, pe suprafaţa unui oraş.

Reţele larg răspândite - WAN (Wide Area Network) acoperă o suprafaţă geografică mare, de exemplu, o ţară sau un continent.

LAN 1 LAN 2

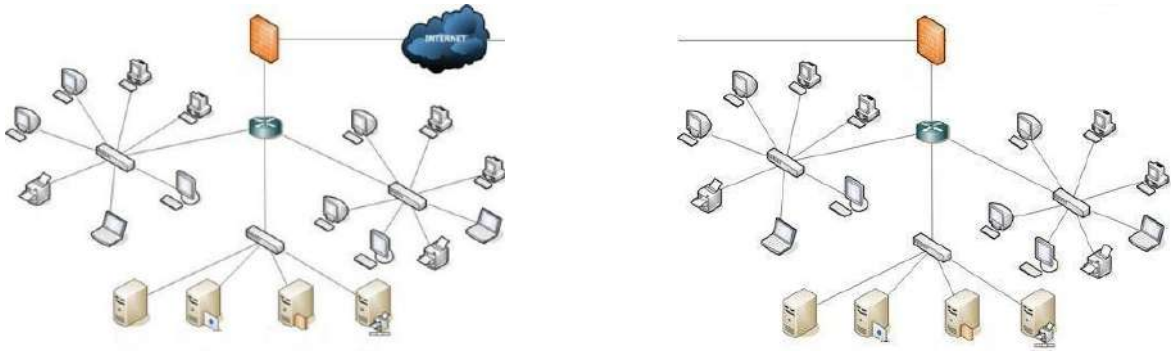


Fig. 2. WAN

După distanţa la care operează reţeaua:

- reţele locale LAN;
- reţele metropolitane MAN;
- reţele de arie întinsă WAN;
- Internet-ul (GAN= Global Area Network).

După topologie:

- reţele tip magistrală (bus);
- reţele tip stea (star);
- reţele tip inel (ring);
- reţele combinate.

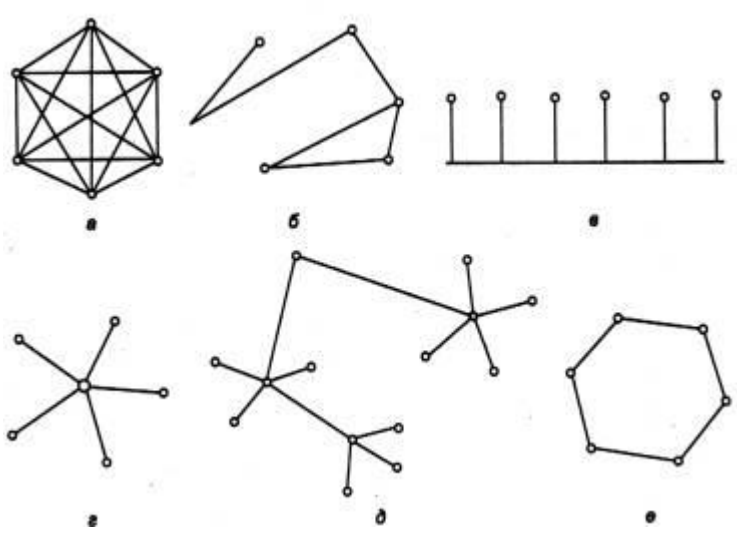


Fig. 3. Topologii standarde de construire a reţelei

După tipul sistemului de operare utilizat:

- reţele peer-to-peer;
- reţele bazate pe server.

După tipul mediului de transmisie a semnalelor:

- reţele prin medii ghidate (cablu coaxial, perechi de fire rasucite, fibra optica);
- reţele prin medii neghidate (transmitere in infrarosu, unde radio, microunde).

După tipul utilizatorilor:

- private (de uz industrial, civil, militar)
- publice

După tehnologia folosită:

- Ethernet;
- token ring;
- token bus;
- arcnet etc.

1.2.1.1 Topologii de reţea

Se numeşte topologie fizică a unei reţele forma în care sunt organizate conexiunile dintre diversele echipamente.

Se numeste topologie logică a unei reţele traseul pe care îl urmează datele prin rețea.

O reţea nu trebuie neapărat sa aibă aceeaşi topologie fizică şi logică (ex. se poate spune despre o reţea că are topologie fizică „stea“, dar topologia logică este „magistrală“).

Topologiile de reţea se clasifică în:

- topologii simple (forme geometrice simple)
- topologii complexe (combinaţii de forme geometrice simple)

Topologii simple de reţea

Topologia Magistrală

Topologia magistrală - bus sau liniară - este cea mai simplă şi mai uzuală metodă de conectare a calculatoarelor în reţea. Dintre cele mai importante caracteristici amintim:

- constă dintr-un singur cablu, numit trunchi care conectează toate calculatoarele din reţea pe o singură linie;
- comunicaţia pe magistrală presupune înţelegerea următoarelor concepte:
  - transmisia semnalului: la un moment dat numai un singur calculator poate transmite mesaje;
  - reflectarea semnalului;
  - terminatorul, utilizat pentru a opri reflectarea semnalului;
- este o topologie pasivă (calculatoarele nu acţionează pentru transmiterea datelor intre calculatoare).

Dacă un calculator se defectează, nu se degenerează restul reţelei, cu condiţia ca placa de reţea a calculatorului respectiv să nu fie defectă; cablul din această topologie poate fi prelungit cu un conector tubular (BNC); un dispozitiv numit repetor este utilizat pentru a conecta două cabluri; el mai are şi rolul de a amplifica semnalul înainte de a-l transmite mai departe; - reprezintă o conexiune multipunct - informaţiile emise de un calculator sunt recepţionate de toate celelalte calculatoare; facilităţi de reconfigurare (toate calculatoarele conectate au drepturi egale); costul redus al suportului şi al dispozitivelor de cuplare.

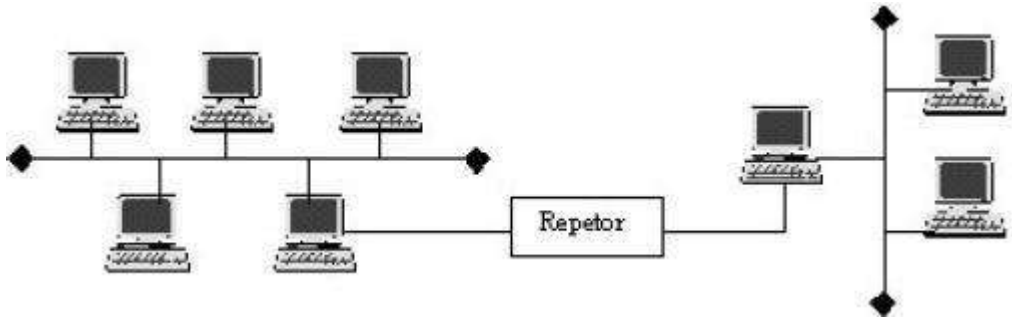


Fig. 4. Prelungirea unei reţele prin repetitor

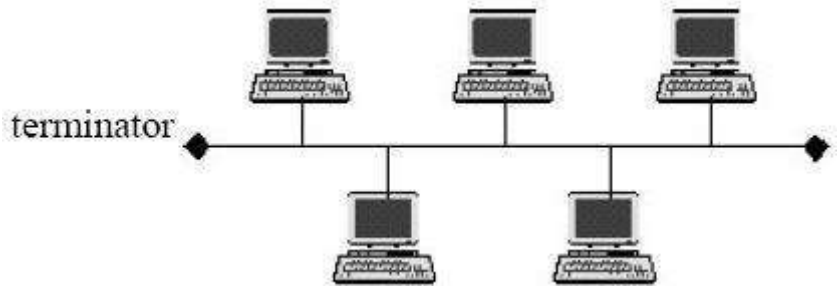


Fig. 5. Topologia Magistrală

Cea mai cunoscută topologie bus este *Ethernet*.

Topologia Stea

Topologia stea - star - atunci când se utilizează această topologie, toate calculatoarele sunt conectate la un nod central care joacă un rol particular în funcţionarea reţelei. Orice comunicaţie între două calculatoare va trece prin acest nod central, care se comportă ca un comutator faţă de ansamblul reţelei. Printre caracteristicile mai importante amintim:

- calculatoarele sunt conectate prin segmente de cablu la o componentă centrală numită hub - Host Unit Broadcast;
- calculatoarele pot comunica direct între ele numai prin intermediul concentratorului;
- aceste reţele oferă resurse şi administraţie centralizate;
- reţelele mari necesită o lungime de cablu mare;
- dacă nodul central (hub - ul) se defectează, cade întreaga reţea;
- dacă un calculator sau cablul care îl conectează la hub se defectează, numai calculatorul respectiv este în imposibilitatea de a transmite sau recepţiona date în reţea;
- poate utiliza în mare parte cablajul telefonic vechi existent într-o societate;
- transferul informaţiei se face punct la punct, dar, cu ultimele tipuri de comutatoare, este posibil şi un transfer multipunct.

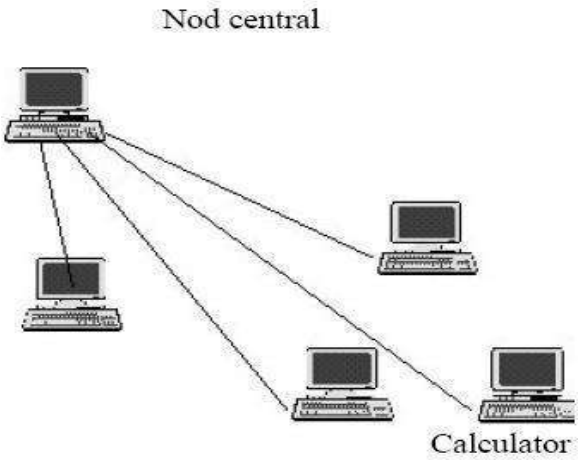


Fig. 6. Topologia stea

Topologia Inel

Topologia inel - ring - într-o astfel de configuraţie, toate calculatoarele sunt legate succesiv între ele, două câte două, ultimul calculator fiind conectat cu primul. Dintre caracteristicile mai importante enumerăm:

- conectează calculatoarele printr-un cablu în formă de buclă (nu există capete libere);
- este o topologie activă - este acea topologie în care calculatoarele regenerează semnalul şi transferă datele în reţea - fiecare calculator funcţionează ca un repetor, amplificând semnalul şi transmiţându-l mai departe, iar dacă îi este destinat îl copiază;
- mesajul transmis de către calculatorul sursă este retras din buclă de către acelaşi calculator atunci când îi va reveni după parcurgerea buclei;
- defectarea unui calculator afectează întreaga reţea;
- transmiterea datelor se face prin metoda jetonului (token passing).

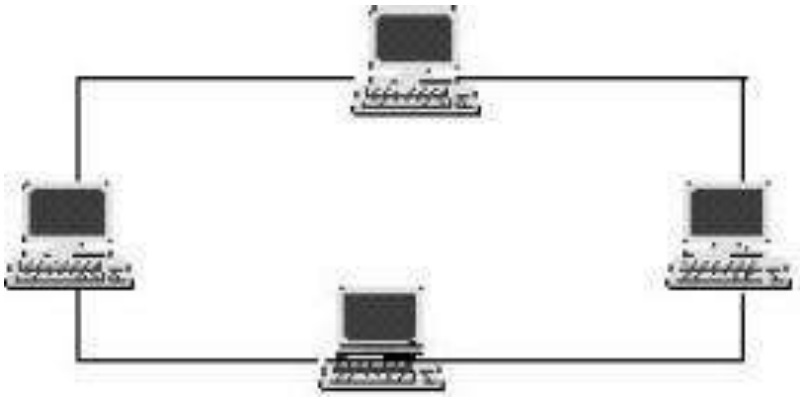


Fig. 7. Topologia inel

Cea mai cunoscută topologie inel este Token - ring de la IBM.

Topologia tree (arbore)

Topologia arbore conectează calculatoarele, stabilind o ierarhie între calculatoarele componente, pornind de la un calculator principal.

Topologia arbore prezintă dezavantajul limitării lungimii maxime a unui segment. În plus, dacă apar probleme pe conexiunea principală, sunt afectate toate calculatoarele de pe acel segment. Avantajul topologiei arbore constă în faptul că segmentele individuale au legături directe.

Comunicarea între calculatoare se face în funcţie de nivelul pe care se află fiecare.

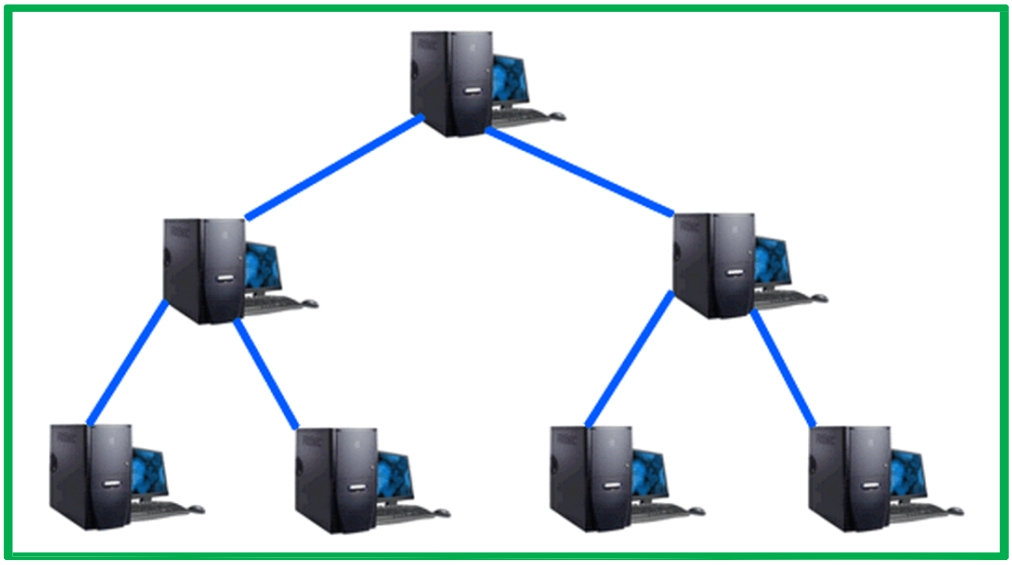


Fig. 8. Topologia Tree

Topologia mesh

Topologia mesh (plasă) este o reţea destinată transportării datelor, instrucţiunilor şi serviciilor de transport voce prin nodurile de reţea. Datorită acestei topologii, putem dispune de conexiuni continue, chiar dacă există legături deteriorate sau blocate. Într-o reţea mesh, dacă toate nodurile sunt interconectate, atunci reţeaua se numeşte complet conectată (fully connected).

Reţelele mesh diferă de celelalte topologii de reţele prin faptul că toate componentele pot să facă legătură între ele prin „sărituri”; ele, în general, nu sunt mobile. Reţelele mesh pot fi văzute ca reţele de tip ad-hoc. Reţelele mobile-ad hoc (MANET‟S – Mobile Ad hoc networks) şi reţelele mesh sunt înrudite, dar reţelele MANET mai au totuşi probleme de mobilitate a nodurilor. Reţelele mesh au proprietatea de autorevendicare: reţeaua poate fi în stare funcţională, chiar dacă un nod se defectează sau dacă sunt probleme cu conexiunea. Acest concept se aplică la reţelele fără fir, la reţelele prin cablu şi la softul de interacţiune. Topologia mesh fără fir este

cea mai frecventă topologie folosită în actualitate. Aceste reţele au fost dezvoltate iniţial pentru aplicaţii militare, dar au fost supuse unei evoluţii semnificative în ultimii zece ani. Progresul echipamentului de transmisiuni de date a permis reţelelor mesh să ofere un larg spectru de servicii, cum ar fi cele de client-acces. Nodurile mesh au devenit mai performante, unele modele pot suporta mai multe cartele radio, fiecare operând la diferite frecvenţe.

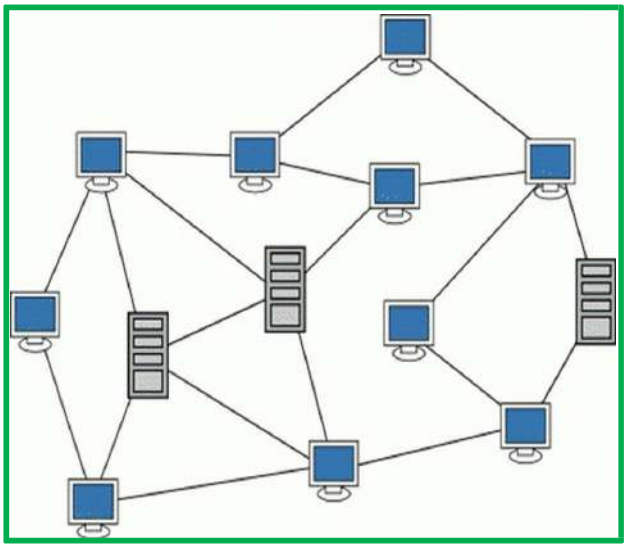


Fig. 9. Topologia Mesh

Alte topologii

În afara acestor topologii standard, există şi alte variante, dintre care cele mai uzuale sunt:

- topologia magistrală-stea: există mai multe reţele cu topologie stea, conectate prin intermediul unor trunchiuri liniare de tip magistrală. Dacă un calculator se defectează, acest lucru nu va afecta buna funcţionare a reţelei, dar, dacă se defectează un hub, toate calculatoarele conectate la el nu vor putea să mai comunice cu restul reţelei (figura 10);
- topologia inel-stea; este asemănătoare topologiei magistrală-stea. Deosebirea constă prin conectarea hub-urilor: în topologia magistrală-stea ele sunt conectate prin trunchiuri lineare de magistrală, iar în topologia inel-stea sunt conectate printr-un hub principal (figura 11).

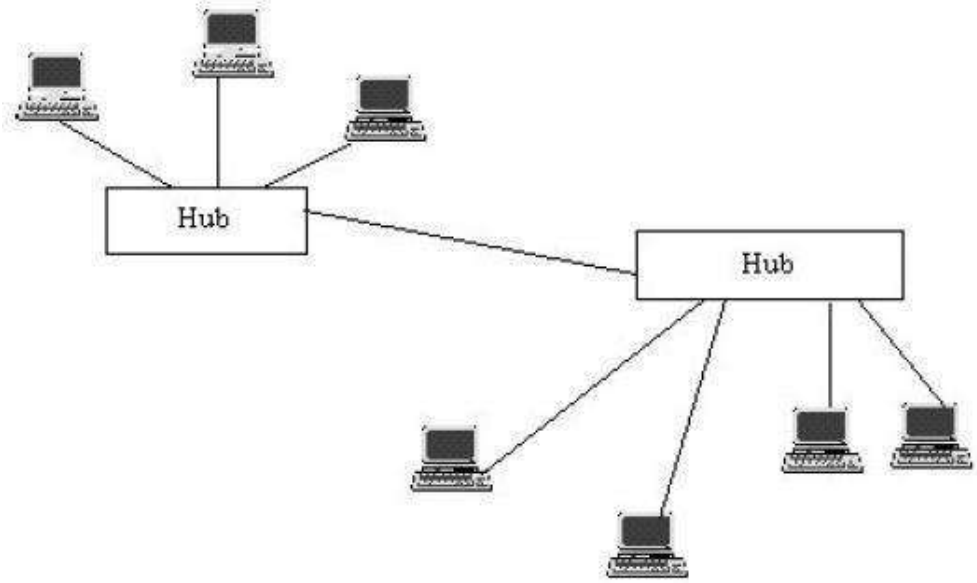


Fig. 10. Topologia Magistrală-Stea

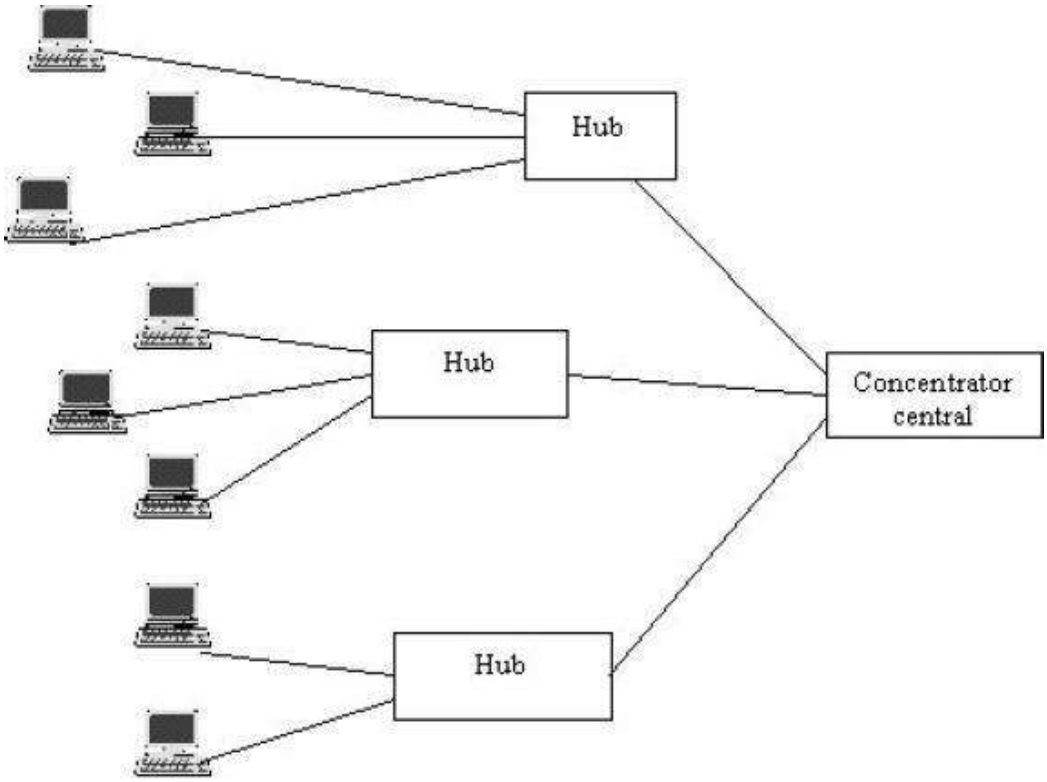


Fig. 11. Topologia Inel-Stea

Reţelele peer-to-peer

Reţelele peer-to-peer (de la egal la egal) sunt acele reţele în care partajarea resurselor nu este făcută de către un singur calculator, ci toate aceste resurse sunt puse la comun de către calculatoarele din reţea.

Aceste reţele au anumite caracteristici:

- numărul maxim de calculatoare care pot fi conectate este de 10 calculatoare;

- implică costuri mici şi, de aceea, sunt des folosite de firmele mici;
- se utilizeaza atunci când zona este mică, securitatea datelor nu este o problemă, organizaţia nu are o creştere în viitorul apropiat;
- toate calculatoarele sunt egale; este şi client, şi server, neexistând un administrator responsabil pentru întreaga reţea.

Reţele bazate pe server

Reţelele bazate pe server (client/server) sunt acele reţele care au în componenţă un server specializat: de fişiere şi de tipărire, de poştă, de aplicaţii, de fax, de comunicaţii.

Printre avantajele retelelor bazate pe server amintim:

- partajarea resurselor;
- securitate;
- salvarea de siguranţă a datelor;
- redundanţă;
- număr de utilizatori.

Într-o reţea combinată există două tipuri de sisteme de operare pentru a oferi, ceea ce mulţi utilizatori consideră a fi o reţea completă. Toate reţelele au anumite componente, funcţii şi caracteristici comune, precum:

- *serverele* sunt acele calculatoare care oferă resurse partajate pentru utilizatorii reţelei;
- *clienţii* sunt acele calculatoare care accesează resursele partajate în reţea de un server;
- *mediile de comunicaţie* reprezintă modul în care sunt conectate calculatoarele în reţea (tipul cablului utilizat, a modemului);
- *datele partajate* reprezintă fişierele puse la dispoziţie de serverele de reţea;
- *resurse*: fişiere, imprimante şi alte componente care pot fi folosite de utilizatorii reţelei.

Alţi termeni frecvent utilizaţi sunt:

- *subreţea*: termenul este potrivit în contextul unei reţele larg răspândite geografic şi se referă la colecţia de rutere şi linii de comunicaţie aflate în proprietatea operatorului de reţea;
- *reţea*: reprezintă combinaţia dintre o subreţea şi gazdele sale (host-uri). În cazul unui LAN, reţeaua este formată din cablu şi gazde;
- *inter-reţea*: ea se formează atunci când se leagă între ele reţele diferite. Legarea unui LAN şi a unui WAN sau legarea a două LAN-uri formează o inter-reţea.

Obs. Structura simplificată a Internetului este, de fapt, o topologie mesh incompletă. În imaginea de mai jos este dată schema comunicaţiilor de date majore din zona Europei de Est:

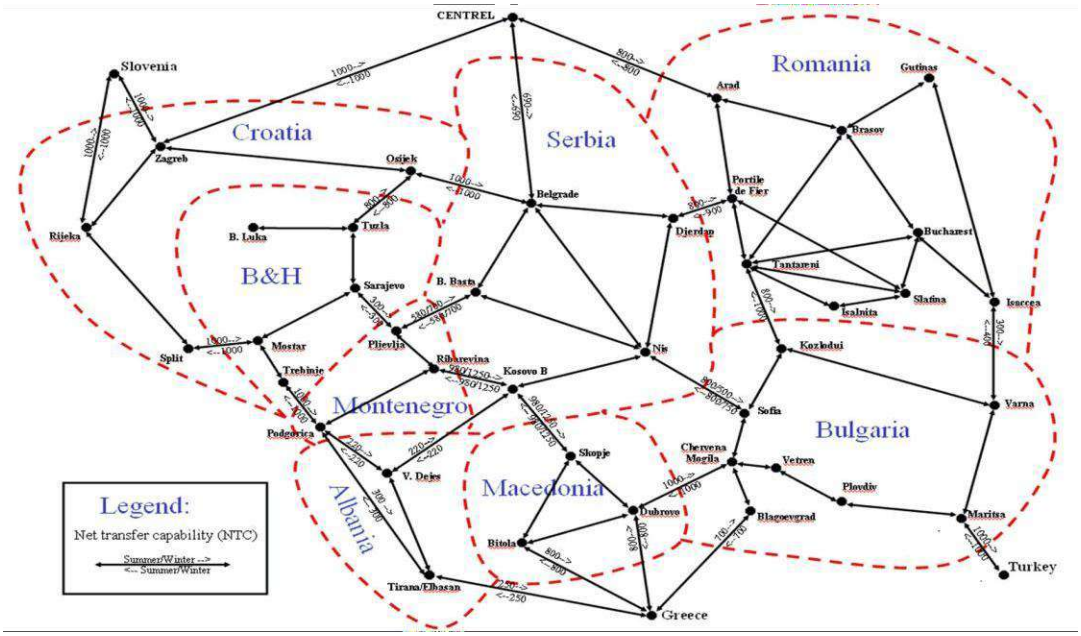


Fig. 12. Comunicaţii in Europa de Est

O schemă mai complexă (dar tot parţială) a legăturilor de date care compun Internetul, cu legăturile colorate după zona de adrese IP, se găseşte în imaginea următoare:

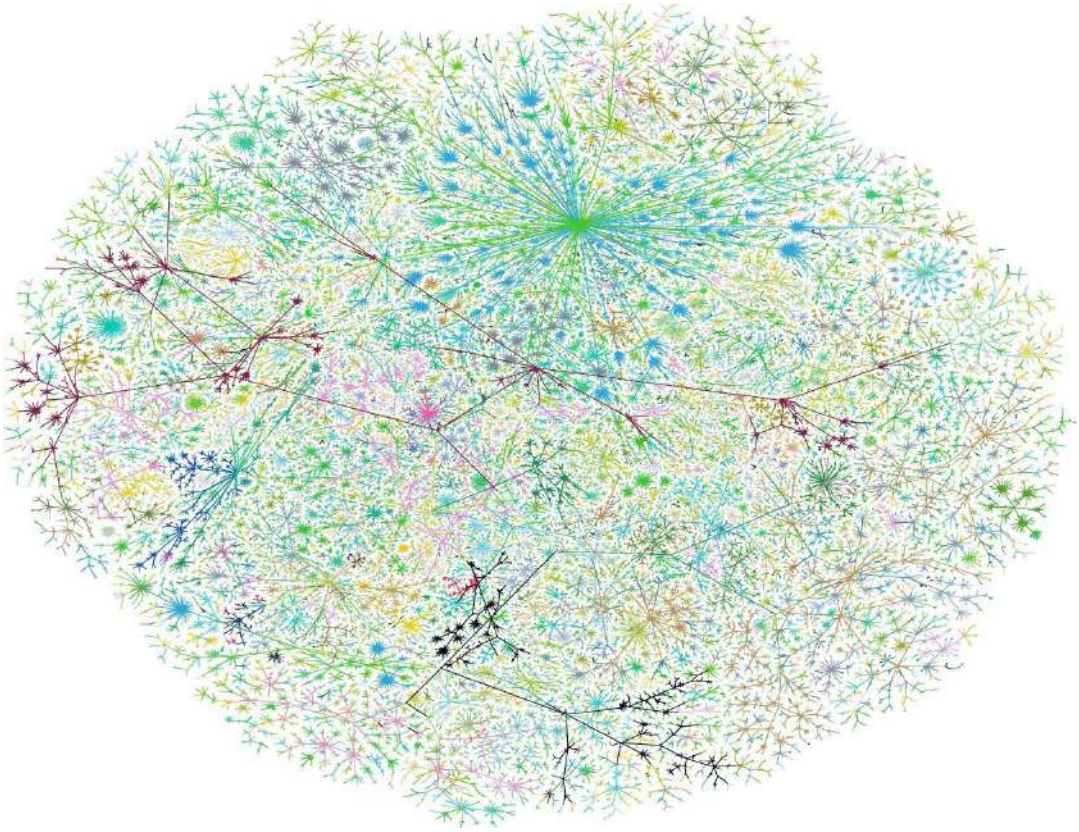


Fig. 13. Comunicaţii in Europa de Est după IP

1.2.2. Nivelurile reţelei

1.2.2.1. Nivelul OSI

Se numeşte sistem deschis un sistem care este interconectat sau poate fi cu usurinţă interconectat cu alte sisteme.

Obs. Prin sistem se înţelege un sistem de calcul sau o grupare de sisteme de calcul.

Se numeşte sistem închis un sistem care nu este interconectat cu alte sisteme şi pentru care, din diverse raţiuni, de cele mai multe ori de securitate, nici nu se doreşte interconectarea cu alte sisteme.

La momentul actual se poate considera că orice sistem deschis ar putea fi, virtual vorbind, conectat direct sau indirect cu orice alt sistem deschis. Din acest motiv, cel puţin la nivel de bază, toate sistemele deschise trebuie să respecte un set de reguli care vor specifica modul în care se face interconectarea.

Acest set de reguli este cuprins într-o formă generalizată în modelul OSI[2] (Open Systems Interconnection – Interconectarea Sistemelor Deschise).

Modelul OSI a fost propus în anii „80 de către organizaţia ISO (International Organization for Standardization – Organizaţia Internaţională pentru Standardizare[3], mai precis: în 1977 s-a propus crearea unui standard, în 1983 a fost creat un model de bază, iar în 1984 standardul a fost publicat simultan de către ISO, sub numele ISO 7498 şi de către ITU-T, sub numele X.200; în 1994 a fost publicată ediţia a doua, iar în 1996 a fost corectată şi republicată ediţia a doua).

Conform modelului OSI, sistemele care se vor interconecta vor funcţiona în pereche (o sursă a comunicaţiei şi o destinaţie de comunicaţie). Atât în sursă, cât şi în destinaţie, va exista un număr de maxim 7 nivele, fiecare dintre ele funcţionând autonom şi având atribuţii specifice.

Obs. 1. Sistemele de calcul obişnuite conţin toate cele 7 nivele.

Obs. 2. Router-ele conţin doar primele 3 nivele.

Obs. 3. Switch-urile conţin doar primele 2 nivele.

Obs. 4. Hub-urile conţin doar primul nivel.

---

[2] Descrierea modelul OSI publicată de ISO, denumită codificat ISO 7498 se găseşte aici: http://standards.iso.org/ittf/licence.html
Descrierea modelului OSI publicată de ITU-T, denumită codificat X.200, se găseşte aici: http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.200-199407-I!!PDF-E&type=items
Lista tuturor standardelor ITU-T (echivalente disponibile complet gratuit ale standardelor ISO) care compun şi completează modelul OSI (precum şi alte standarde) se găseşte aici: http://www.itu.int/rec/T-REC-X/en

[3] Site-ul ISO este: http://www.iso.org
Lista tuturor standardelor emise de ISO şi disponibile public se găseşte aici: http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

Definiţie. Un protocol este un set de reguli pe care fiecare calculator trebuie să-l respecte pentru a comunica cu un altul. O altă definiţie mai tehnică: un protocol de comunicare reprezintă un set de reguli care determină formatul şi modalitatea în care datele sau informaţia pot fi trimise sau primite.

Pe lângă modul de împărţire pe verticală, în modelul OSI se mai apelează la unul pe orizontală, adică fiecare start este subdivizat pe orizontală - în aceste locuri aflându-se protocoalele. Ca principiu, un protocol M dintr-un strat 4 al calculatorului sursă va comunica în calculatorul destinaţie cu protocolul M din stratul 4 al maşinii respective. Spre exemplu, TCP de strat 4 comunică cu TCP de strat 4 din calculatorul cu care a stabilit o conexiune. Imaginea de mai jos, cred eu, evidenţiază cel mai bine modul de comunicare între protocoale.

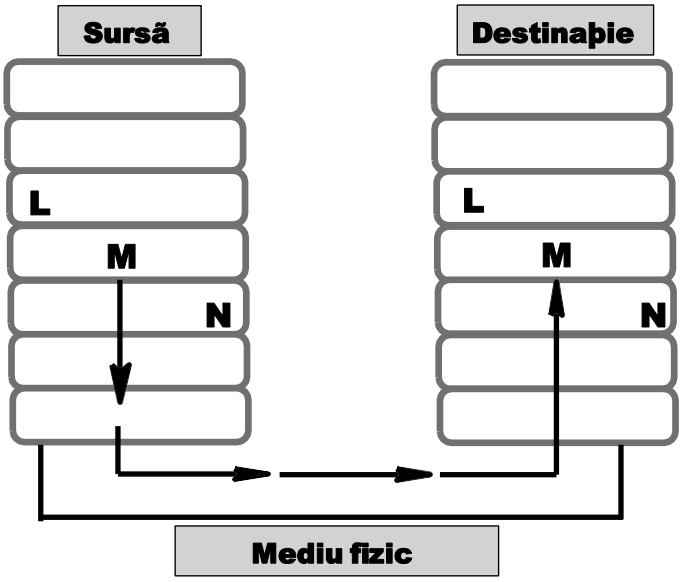


Fig. 14. Model de comunicare

Scopurile propuse de modelul OSI

Care sunt scopurile pentru care a fost propus acest sistem? Deşi astăzi sunt şi alte sisteme în funcţiune, cei mai mulţi distribuitori de echipamente de comunicaţie folosesc OSI pentru a educa utilizatorii în folosirea echipamentelor. Se consideră că OSI este cel mai bun mijloc prin care se poate face înţeles modul în care informaţia este trimisă şi primită. În modelul OSI sunt şapte straturi şi fiecare strat are funcţii diferite în reţea, această repartiţie purtând numele stratificare (engl. layering). Se pot enunţa astfel, schematic, câteva dintre avantajele folosirii OSI:

- descompune fenomenul de comunicare în reţea în părţi mai mici şi, implicit mai simple;
- standardizează componentele unei reţele, permiţând dezvoltarea, indiferent de producător;

- permite comunicarea între diferite tipuri de hardware şi software;
- permite o înţelegere mai uşoară a fenomenelor de comunicaţie.

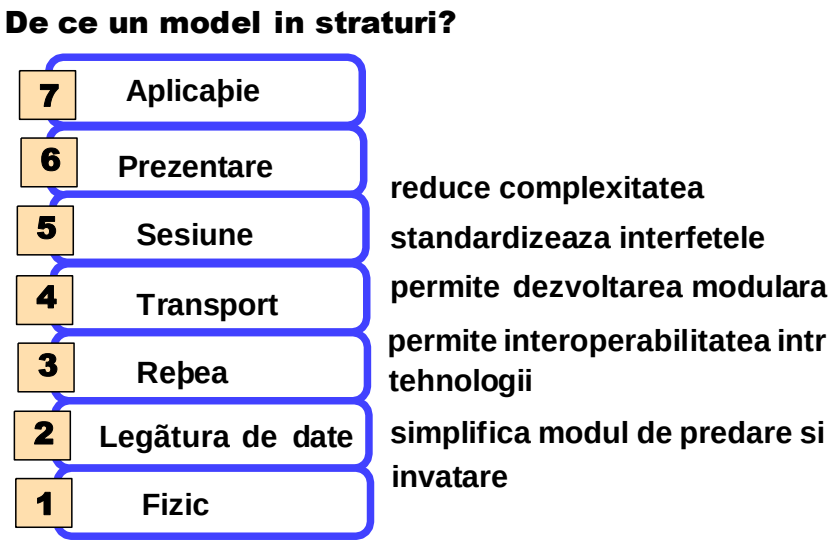


Fig. 15. Modelul OSI

Cele 7 nivele ale modelului OSI se numesc:

7. Nivelul Aplicaţie
6. Nivelul Prezentare
5. Nivelul Sesiune
4. Nivelul Transport
3. Nivelul Reţea
2. Nivelul Legătură de date
1. Nivelul Legătură fizică

Modul în care sunt conectate cele 7 nivele este reprezentat în figura următoare:

Fig. 16. Legăturile între nivele

Obs. Modelul OSI este unul cu specificaţii vagi, generale, dar toate implementările practice mai detaliate (ex.: stivele de protocoale TCP/IP, IPX/SPX) se suprapun peste acest model.

Un exemplu asupra modului în care se face suprapunerea între TCP/IP şi OSI se poate vedea în figura următoare:

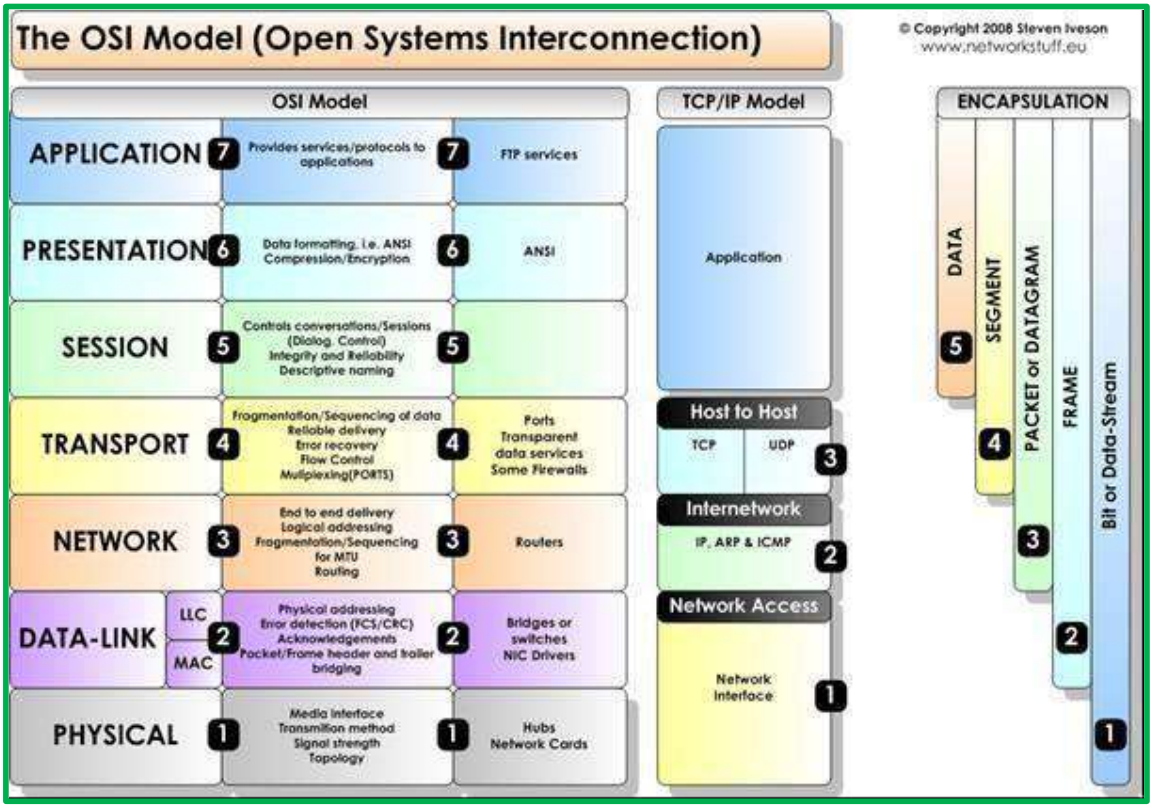


Fig. 17. TCP/IP si OSI

Modelul OSI respectă următoarele 2 reguli:

1. Orice nivel al modelului comunică direct numai cu nivelul imediat superior şi imediat inferior.

2. Prin intermediul tuturor nivelelor inferioare, fiecare nivel este indirect conectat cu nivelul echivalent de la celălalt capăt al comunicaţiei.

Obs. Fiecare nivel este implementat autonom, implementarea având ca singură obligaţie conservarea intrărilor şi a ieşirilor către nivelul superior, respectiv inferior.

Obs. Această modularizare a nivelelor oferă o enormă flexibilitate, practic fiecare individ sau organizaţie care implementează un sistem de comunicaţii putând să facă aproape orice schimbări în limitele observaţiei anterioare.

Obs. În ciuda faptului că este un model teoretic, buna cunoaştere a modelului OSI sprijină foarte mult activităţile practice legate de sistemele de comunicaţie (proiectare, implementare, configurare, administrare, diagnoză, depanare).

Circuitul informaţiei în modelul OSI

Nivelul Aplicaţie (7)

Conform modelului OSI, nivelul aplicaţie este beneficiar al serviciilor oferite de nivelele inferioare. La acest nivel se găsesc diversele aplicaţii care utilizează comunicaţii de date (browsere, servere web, clienţi şi servere FTP, clienţi şi servere de chat, clienţi şi servere de ştiri etc.).

Nivelul prezentare (6)

La acest nivel este asigurată coerenţa şi inteligibilitatea datelor (pentru utilizatorul uman şi/sau pentru aplicaţiile de la nivelul aplicaţie). Acest nivel este asigurat de formatele de date şi de fisiere, de bibliotecile şi programele utilizate pentru vizualizarea acestora, de codecuri şi diverse filtre.

Nivelul sesiune (5)

Este nivelul la care sunt create, menţinute şi închise sesiunile de lucru. Tot la acest nivel se fac identificarea şi autentificarea utilizatorilor şi/sau aplicaţiilor care utilizează sesiunile de lucru respective.

Nivelul transport (4)

Este nivelul la care informaţia trimisă de la sursă către destinaţie este separată în secvenţe de date de lungimi variabile, care vor fi trimise în mod independent către destinaţie. La capătul opus al comunicaţiei, la destinaţie, secvenţele de date vor fi reasamblate pentru a reconstitui informaţia originală.

Obs. La nivelul transport nu se are în vedere traseul pe care informaţia va ajunge de la sursă la destinaţie, ci doar se asigură livrarea datelor (în sensul utilizării unui mecanism de confirmare a livrării – acknowledgement).

Nivelul rețea (3)

La nivelul reţea, secvenţele de date sunt fragmentate mai departe în pachete de date (packets).

Obs. Spre deosebire de secvenţele de date, pachetele de date au o lungime relativ constantă şi limitată, iar dimensiunea unui pachet de date este dependentă de protocolul folosit pentru comunicaţie (ex. IP, IPX etc.).

Obs. Pachetele de date conţin, în afară de datele propriu-zise, şi un antet care include adresa IP a sursei, adresa IP a destinaţiei, dimensiunea pachetului, informaţii despre protocolul folosit, precum şi alte informaţii descriptive.

La nivelul reţea al destinaţiei, acest antet va fi îndepărtat, iar datele din pachete vor fi reasamblate în secvenţe de date.

Obs. Procesul de reasamblare, atât pentru pachete, cât şi pentru secvenţe, este conceput de aşa natură încât secvenţele şi ulterior datele să poată fi reconstituite corect, indiferent de ordinea în care au ajuns pachetele/au fost reconstituite secvenţele.

La nivelul 3 se face şi alegerea căii pe care o va urma pachetul de date pentru a ajunge (probabil) la destinaţie.

Obs. Dat fiind că nu este garantată livrarea şi că nu se cunoaşte de la început calea pe care pachetele vor ajunge (dacă vor ajunge) la destinaţie, pachetele de date sunt prevăzute cu un sistem de autodistrugere cu contor (contorul este denumit TTL-Time To Live) care măsoară numărul de salturi efectuate (sau de noduri prin care a trecut pachetul)–hop count. Acest mecanism este implementat pentru a asigura că un pachet de date care nu îşi găseşte destinaţia nu va „trăi“ veşnic (în sensul de a fi retrimis la infinit dintr-un nod într-altul).

În principiu, găsirea căii către destinaţie se bazează pe ideea că, în fiecare nod, pachetul ar trebui trimis către cel mai probabil nod următor.

Nivelul legătură de date (2)

La acest nivel, pachetele sunt ambalate (împachetate) şi primesc încă un antet pe lângă antetul de pachet. Pachetele, împreună cu acest antet suplimentar, se numesc cadre de date (frames). Antetul cadrului conţine adresa MAC a nodului curent, adresa MAC a nodului următor, lungimea cadrului de date, protocolul de care acesta aparţine şi alte informatii descriptive. La fiecare salt, cadrul de date este descompus şi recompus, iar antetul cadrului este reactualizat.

Structura cadrului de date este dependentă de arhitectura de reţea folosită şi poate conţine informaţii precum tipul de protocol folosit, suma de control etc.

Nivelul 2 conţine 2 subnivele:

- Subnivelul Controlul legăturii logice (Logical Link Control);
- Subnivelul Controlul accesului la mediul de comunicaţii (Media Access Control).

La nivelul 2 al destinaţiei, se elimină antetul cadrului de date, rămânând doar pachetul de date. La nivelul 2 al modelului OSI este asigurată livrarea punct la punct. Asigurarea constă în faptul că, pentru fiecare cadru de date recepţionat, este trimis înapoi un răspuns de confirmare (aknowledgment). În cazul în care nu se primeşte confirmarea pentru un anumit pachet/cadru, acesta va fi retrimis după un interval de timp, ales aleator.

Nivelul legătură fizică (1)

La acest nivel, biţii sau grupurile de biţi din cadrele de date sunt codificate corespunzător cu tehnologia de comunicaţii existentă între cele 2 puncte (sub formă de unde radio/impulsuri electrice/impulsuri optice etc.).

Diverse modalităţi de codificare sunt ilustrate în figura următoare:

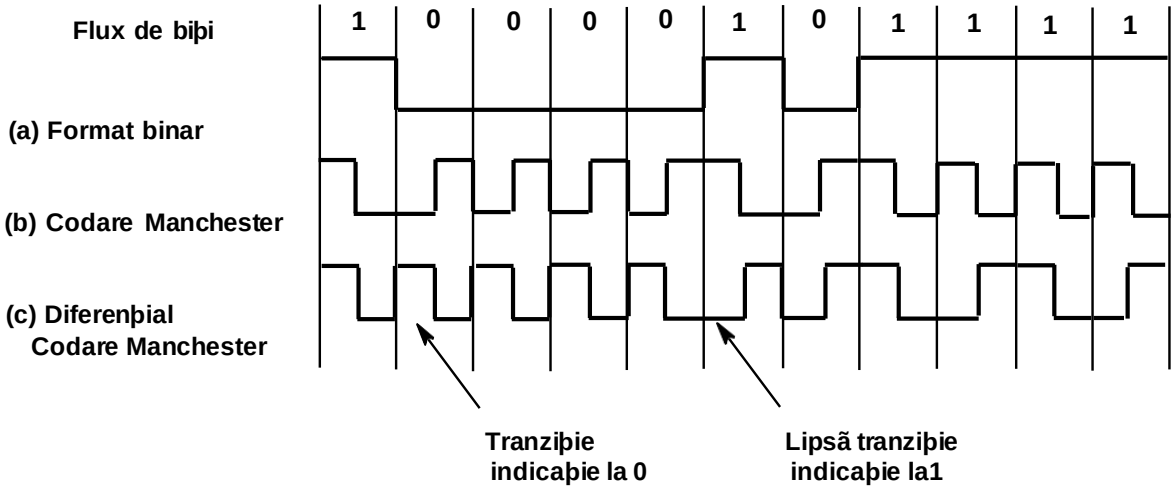


Fig. 18. Modalităţi de codificare

La nivelul legătură fizică al destinaţiei, undele radio/impulsurile electrice/impulsurile optice sunt retransformate în biţi/grupuri de biţi.

La destinaţie, toată succesiunea de transformări efectuate asupra datelor petrecute la sursă este reluată în sens invers:

- Antetul cadrelor de date este îndepărtat, rămânând pachetele de date;
- Pachetele de date îşi vor pierde antetul, iar datele din ele vor fi reasamblate în ordinea corectă pentru a forma secvenţe de date;
- Secvenţele de date îşi vor pierde antentul şi vor fi reasamblate în ordinea corectă pentru a forma datele originale.

1.2.2.2. Modelul TCP/IP

Modelul TCP/IP este cel mai utilizat model pentru comunicarea dintre calculatoare. Denumirea sa provine de la cele două protocoale fundamentale utilizate: TCP (Transmission Control Protocol) şi IP (Internet Protocol).

Ca şi alte modele, modelul TCP/IP este construit pe niveluri. Fiecare nivel se bazează pe cel aflat sub acesta şi oferă servicii pentru nivelul superior. Nivelul *n* de pe o maşină comunică cu nivelul *n* de pe cealaltă maşină. Regulile şi convenţiile utilizate în comunicarea între niveluri constituie un protocol pentru respectivul nivel.

Comunicarea dintre nivelul *k* dintre două maşini nu are loc direct, ci prin intermediul straturilor inferioare. Dacă se doreşte comunicarea pentru nivelul *k* între maşinile A şi B, în maşina A, nivelul k foloseşte serviciile nivelului *k-1*; practic, mesajul este trimis acestuia, apoi circulă mai departe până la nivelul 1, fiind trimis fizic prin cablu sau eter, ajunge la maşina B, la nivelul 1 şi ajunge la nivelurile superioare până ajunge la nivelul *k*. Când mesajul este trimis la un nivel inferior, acesta poate adăuga, la rândul lui, informaţii specifice nivelului (protocolului), eventual să împartă mesajul în pachete mai mici pentru a putea fi trimis nivelului inferior sau prin reţea.

Între două niveluri adiacente există o interfaţă. Aceasta defineşte operaţiile şi serviciile primitive care sunt oferite de nivelul de jos celui aflat deasupra.

O mulţime de niveluri şi protocoale alcătuiesc o arhitectură de reţea.

O listă de protocoale utilizate de către un anumit sistem, măcar câte un protocol pentru fiecare nivel, se numeşte stivă de protocoale.

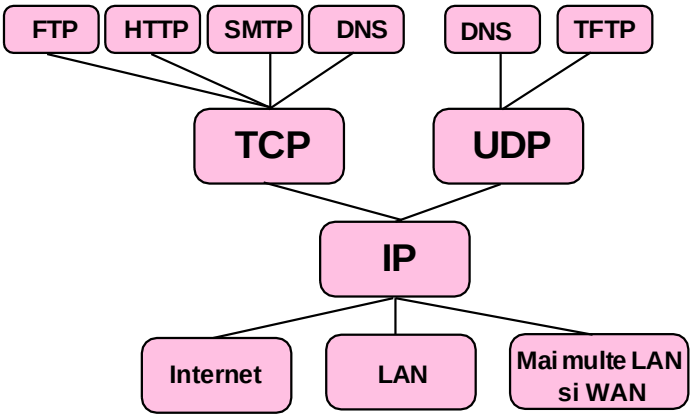


Fig. 19. Protocoale TCP/IP

Nivelurile modelului TCP/IP sunt:

1. Nivelul acces la mediu (host to network)
2. Nivelul reţea
3. Nivelul transport
4. Nivelul aplicaţie

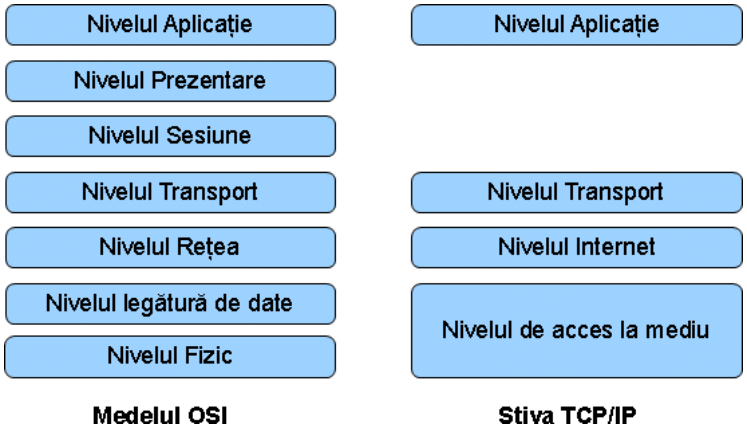


Fig. 20. Nivele OSI si TCP/IP

În continuare, voi prezenta succint fiecare nivel şi protocoalele aferente fiecăruia.

Nivelul de acces la mediu

Acest nivel face legătura dintre canalul fizic de transmitere a informaţiilor şi nivelul reţea. Modelul de referinţă TCP/IP nu prezintă în documentaţiile sale prea multe informaţii despre acest nivel şi protocolul utilizat pentru transmiterea efectivă a pachetelor în reţea.

La acest nivel, identificăm două protocoale pentru rezoluţia adresei IP: ARP (Address Resolution Protocol) şi RARP (Reverse Address Resolution Protocol). Acestea au rolul de a obţine adresa fizică (adresa Ethernet sau adresa MAC) corespunzătoare unei adrese IP şi invers, pornind de la adresa Ethernet să obţinem adresa IP.

Cu adresele IP lucrează operatorii umani, iar la nivel de maşină, se lucrează cu adrese Ethernet. Fiecare placă de reţea are o adresă unică.

Nivelul internet

Rolul acestui nivel este de a emite pachete în orice reţea şi de a face ca pachetele să circule independent până la destinaţie. Eventual, pot exista mai multe rute între două gazde. Acest nivel defineşte clar un format de pachet şi un protocol denumit IP (Internet Protocol). Acesta a fost de la început proiectată pentru interconectarea reţelelor de calculatoare şi s-a ţinut cont de cerinţele iniţiale (de a nu întrerupe conexiunile existente şi de a găsi rute alternative pentru pachete în cazul defectării anumitor linii de transmisie sau noduri de legătură), care a condus la apariţia ARPANET-ului.

Nivelul reţea se ocupă de dirijarea pachetelor în reţea, utilizând diverşi algoritmi pentru realizarea acestui lucru, precum şi de controlul congestiei.

Nivelul transport

Nivelul de transport permite realizarea de comunicări de la un capăt la altul (end-to-end). Acest nivel cuprinde două protocoale: TCP (Transmission Control Protocol) şi UDP (User Datagram Protocol), pe care le prezentăm în continuare separat.

TCP (Transmission Control Protocol)

Protocolul TCP este un protocol sigur, orientat, a cărui conexiune permite trimiterea unui flux de octeţi de la o maşină la alta, fără erori.

Protocolul este sigur întrucât nu au loc pierderi de informaţii. Pentru fiecare pachet IP trimis se aşteaptă o confirmare de primire. Dacă aceasta nu apare într-un anumit interval de timp, pachetul este retransmis.

Mai mult, la recepţionare se verifică integritatea datelor; în cazul în care datele au fost alterate se cere retransmiterea mesajului.

Pachetele IP pot ajunge la destinaţie urmând drumuri diferite, iar unele pachete pot ajunge înaintea altora. De exemplu, ultimul pachet emis poate fi antepenultimul recepţionat. Protocolul TCP reface secvenţa în care au fost emise pachetele.

Protocolul TCP permite conexiunea, acest lucru înseamnând că ambii parteneri care efectuează comunicarea trebuie să fie disponibili simultan. Este similar cu convorbirile telefonice, când ambele persoane implicate trebuie să fie disponibile pentru comunicare.

UDP (User Datagram Protocol)

Faţă de TCP, protocolul UDP este nesigur, fără conexiune şi nu asigură secvenţierea pachetelor.

Protocolul UDP nu mai gestionează numărul de secvenţă pentru pachete, de aceea, nu poate reface secvenţa în care au fost emise pachetele. Tot din aceeaşi cauză, nu poate determina dacă un pachet s-a pierdut. În schimb, rata de pierdere a pachetelor este foarte mică.

Fiind un protocol neorientat conexiune, nu trebuie ca cei doi parteneri de comunicare să fie disponibili simultan. Sistemul este similar cu cel de poştă obişnuită. Pentru ca A să comunice cu B, A scrie un mesaj (scrisoare) şi îl expediază, timp în care nu este solicitat B, apoi mesajul ajunge la B (B nu este obligat ca în acel moment să citească pachetul), la un moment dat B citeşte mesajul şi trimite înapoi răspunsul, timp în care nu este solicitat A.

Întrucât nu mai are loc verificarea secvenţelor şi reordonarea pachetelor, se câştigă timp, transmisia de date prin intermediul protocolului UDP fiind mai rapidă, însă cu riscul de a pierde pachete şi de a primi în altă ordine decât cea în care au fost emise.

Nivelul aplicaţie

Aflat peste nivelul de transport, nivelul aplicaţie conţine toate protocoalele de nivel înalt. Iniţial, acesta includea protocoalele: Telnet (pentru termial virtual), FTP (pentru transfer de fişiere) şi SMTP (pentru poşta electronică). Apoi au fost adăugate şi alte protocoale, cum ar fi protocolul DNS pentru stabilirea corespondenţei dintre numele unei gazde şi adresa IP, NNTP, utilizat pentru servicii de ştiri şi HTTP, pentru pagini Web.

Circulaţia informaţiei

Cum circulă informaţia? Odată ce a fost creată (spre exemplu, după ce am scris un email), informaţia trebuie să treacă prin toate cele 7 straturi, unde va fi procesată pentru trimitere. Această procesare presupune desfacerea şi asamblarea ei în nişte pachete de date, procesul purtând numele *încapsulare*. Acest proces constă, pe lângă crearea pachetelor, şi într-un fenomen prin care se adaugă la fiecare pachet headere şi trailere, care definesc un anumit protocol ce va procesa la destinaţie acel pachet. Pentru o mai simplă înţelegere a fenomenului, se poate lua exemplul cu email-ul. Aşadar, paşii vor fi următorii:

Construirea datelor. Utilizatorul scrie email-ul, al cărui text şi, eventual, imagine vor fi procesate în straturile superioare pentru a avea un format care să poată fi trimis în reţea.

Segmentare datelor. Se face la stratul 4; în felul acesta, se garantează că datele vor ajunge în siguranţă de la o maşină la alta.

Adăugarea adreselor de reţea. Se face la nivelul stratului 3, prin adăugarea unui header la segmentul stratului 3, rezultând ceea ce numim pachet. Acest header vine cu informaţii deosebit de preţioase: adresa logică către care va fi expediat pachetul, adresa logică a sursei. Tot la acest nivel, se decide care va fi următoarea maşină careia i se va livra pachetul (next hop).

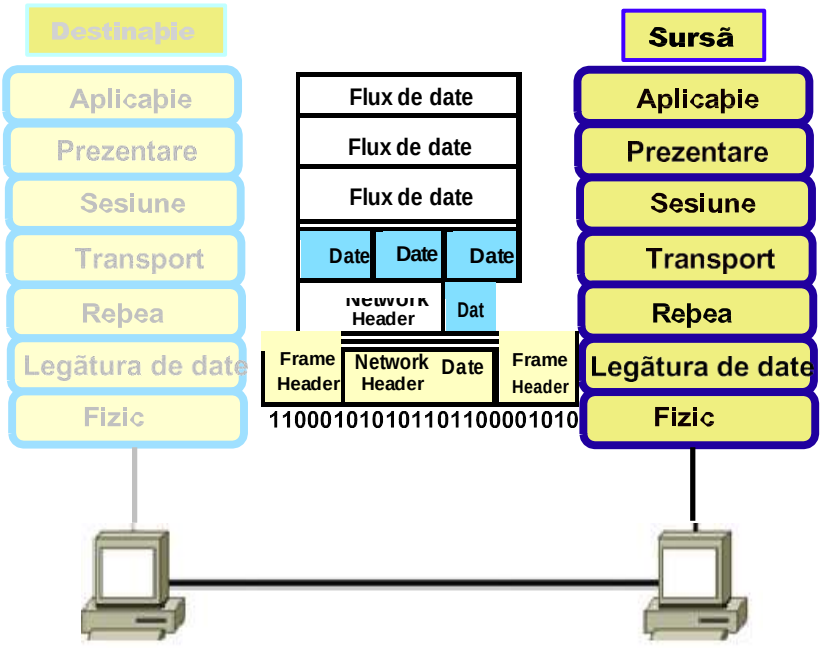


Fig. 21. Încapsularea datelor

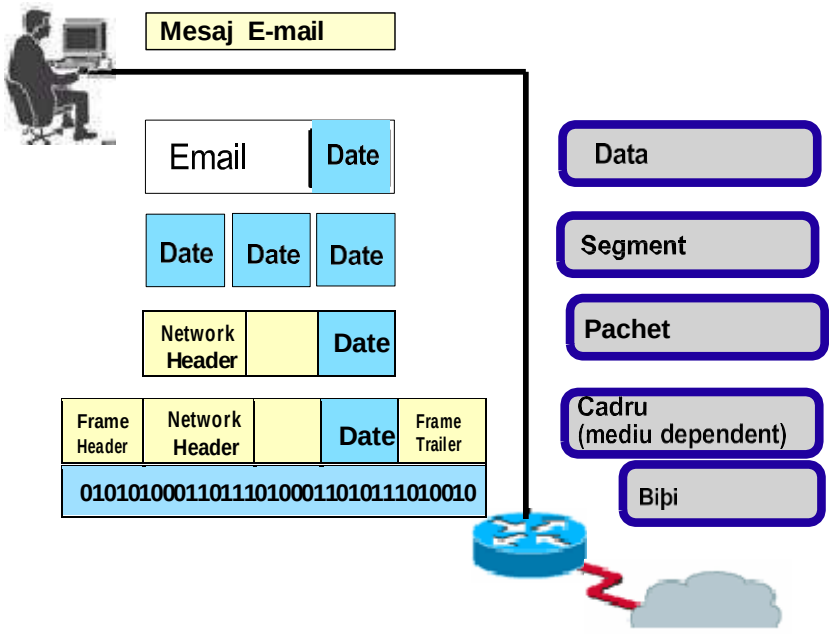


Fig. 22. Încapsularea în modelul OSI

Adăugarea headerului de strat 2. Aici se adaugă un header care conţine informaţii cu privire la următoarea maşină care va primi acea informaţie, rezultatul acestei asamblări fiind ceea ce numim un frame. Trebuie deosebită această adresare de cea de la layer 3: spre exemplu, daca sunt într-o reţea R şi trimit informaţia în aceeaşi reţea, IP-ul destinaţiei va fi al maşinii către care trimit, iar MAC-ul, de asemenea; pe când, dacă trimit într-o altă reţea, IP-ul va fi al destinaţiei, iar MAC-ul va fi al default gateway-ul din reţeaua R în care mă aflu eu.

Convertirea frame-ului într-o secvenţă de biţi (0 şi 1). Aşa circulă informaţia în mediul de propagare. Aici se mai află şi un ceas care permite celor două maşini care comunică să se poată sincroniza.

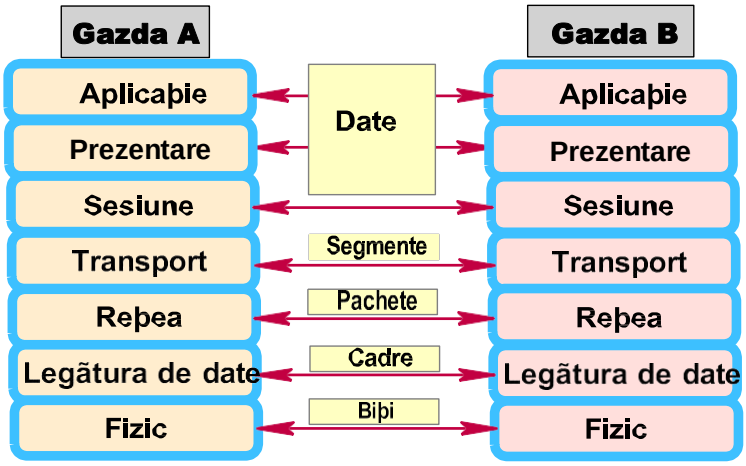


Fig. 23. Comunicarea peer-to-peer

Acelaşi parcurs îl are informaţia odată ce a atins destinaţia, dar în sens contrar: de la stratul 1 până la 7. În sensul acesta, trebuie precizat că fiecare strat comunică cu echivalentul său din maşina cu care s-a stabilit o conexiune. Acest tip de comunicare se numeşte comunicare peer-to-peer şi implică folosirea unor PDU-uri (Prodocol Data Units). Pentru layer 4, PDU-ul este segmentul, pentru layer 3, packet-ul, iar pentru layer 2, frame-ul.

Câteva porturi asignate sunt prezentate in urmatorul tabel:

Tabel 1

Port	Protocol	Unitate
21	FTP	Transfer de fişiere
23	TELNET	Login la distanţă
25	SMTP	E-mail
69	TFTP	Protocol de transfer de fişiere trivial
79	FINGER	Căutare de informaţii despre un utilizator
80	HTTP	World Wide Web
110	POP-3	Acces prin e-mail la distanta
119	NNTP	Ştiri USENET

1.3. APLICAŢIE

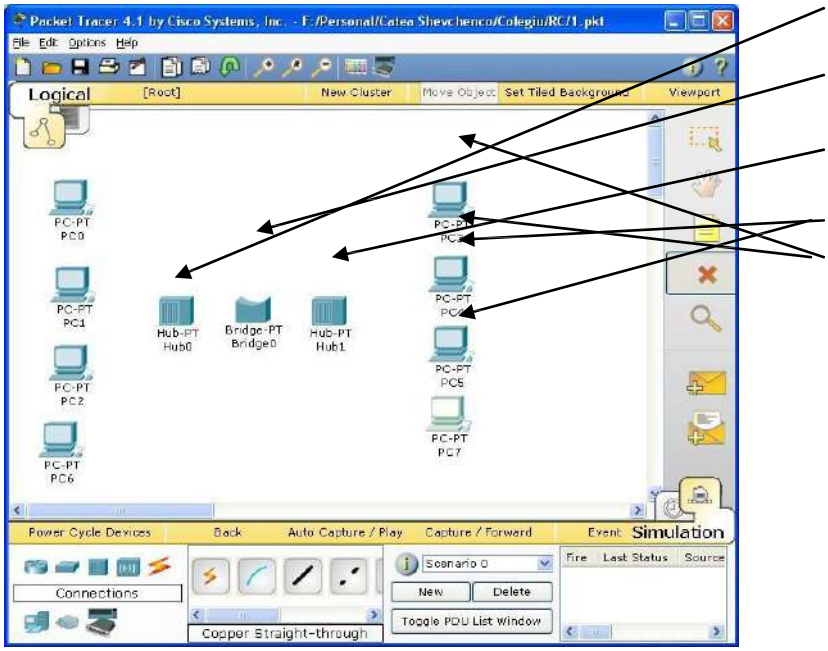
Simularea unei reţele de calculatoare cu ajutorul aplicaţiei Packet Tracer de la CISCO.

Obiective:

- aplicarea aspectelor teoretice;
- crearea unui proiect care va avea la bază 2 cazuri:
    - în primul caz, de descris o reţea de calculatoare compusă din 8 calculatoare - câte 4 calculatoare la câte un hub (hub-urile între ele se vor conecta printr-un bridge);
    - în cel de-al doilea caz, de descris o reţea de calculatoare formată din 6 calculatoare, câte 3 calculatoare la fiecare switch;
- evidenţierea etapelor de creare a reţelei;
- identificarea diferenţelor şi asemănărilor dintre etapele parcurse;
- analiza concluziilor.

Primul Caz:

Am creat o reţea de calculatoare formată din 8 calculatoare, 2 hub-uri şi un bridge.



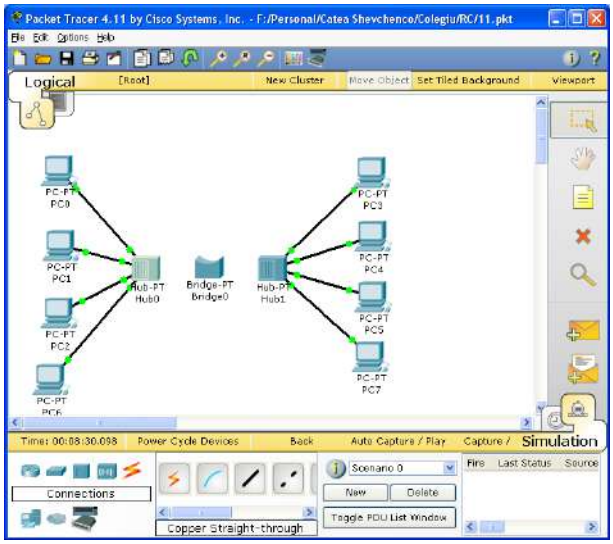
Hub – pentru conectarea celor 4 calculatoare
Bridge – pentru conectare hub-urilor
Hub – pentru conectare celorlalte 4 calculatoare

PC(calculatoarele)

Conectăm calculatoarele la hub-uri utilizând conexiunea ***Straigth-through***.

Conexiunea ***Straigth-through***

Obţinem:

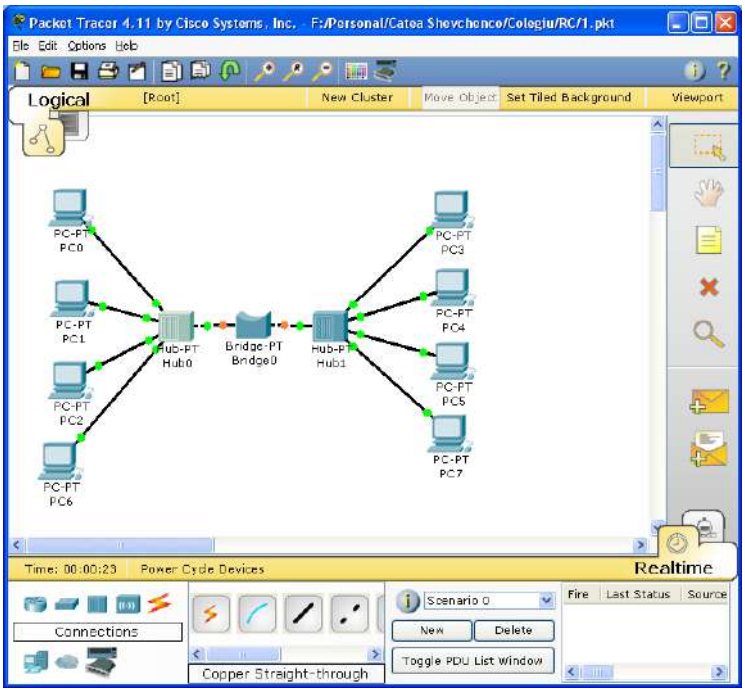


Conectăm Primul Hub şi cel de-al doilea printr-un bridge.

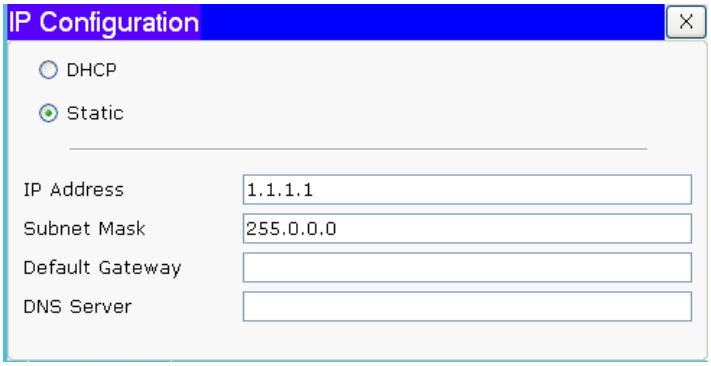
Se utilizează conexiunea ***Cross-Over***

Conexiunea ***Cross-Over***

Obţinem:



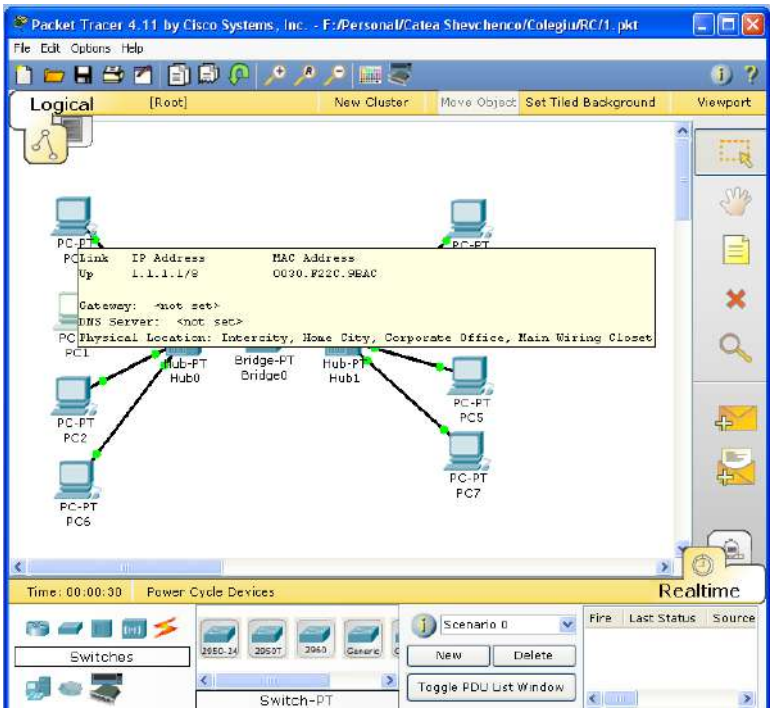
Configurăm calculatoarele:



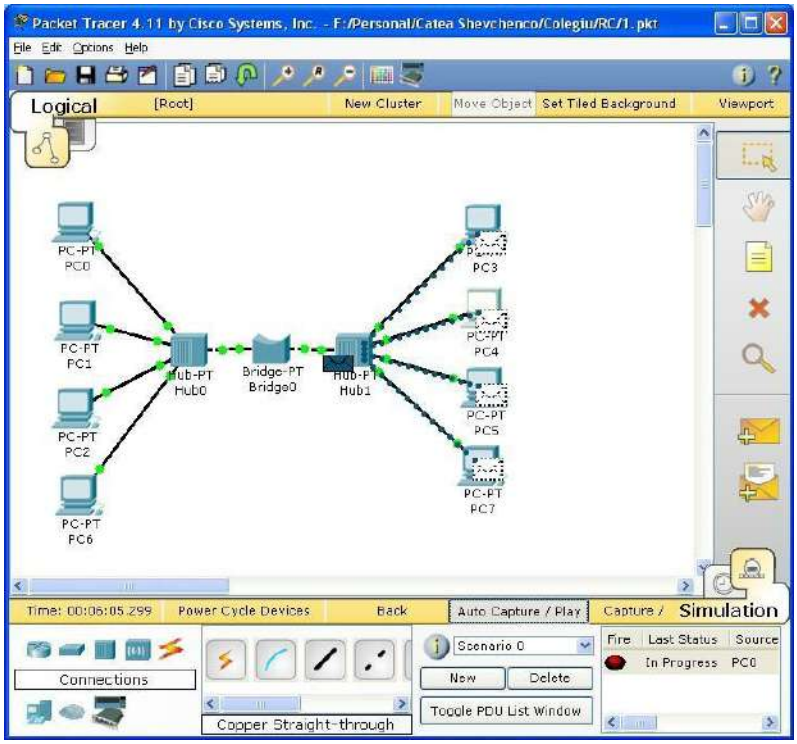
Configurăm toate calculatoarele; IP-urile trebuie să fie diferite.

Obs: Utilizăm un bridge pentru a nu supraîncărca reţeaua. Acesta este la fel ca un switch, şi anume dispune de un tabel de adrese care permite înregistrarea adresei fiecăruia dintre calculatoare, astfel permiţând transmiterea informaţiei doar calculatorului-destinaţie.

Astfel, obţinem:



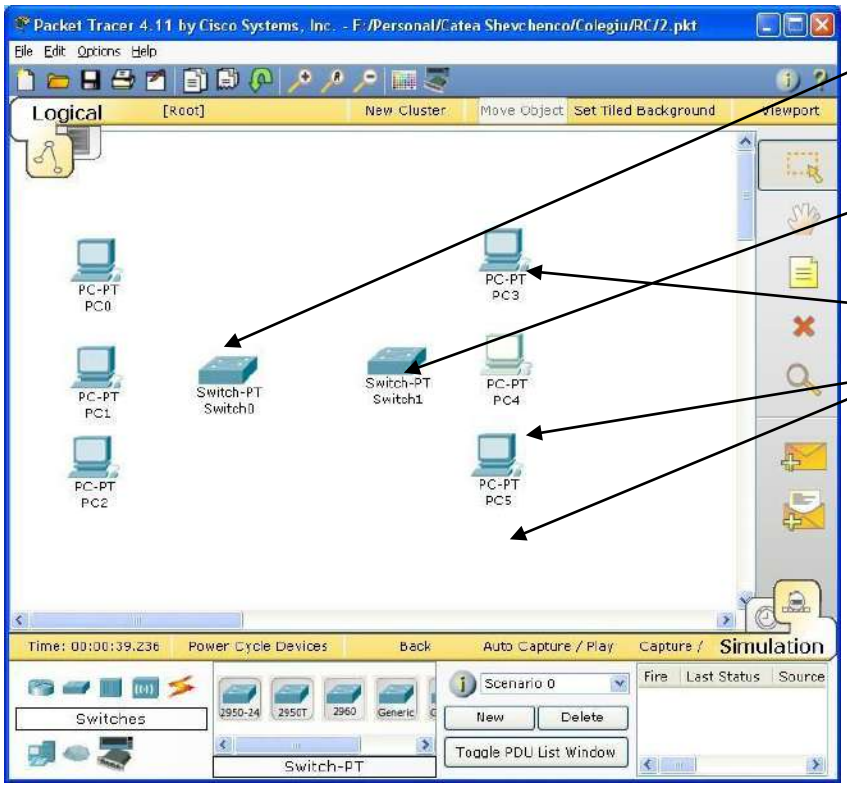
Transmiterea informaţiei:



Obs: Observăm că la comunicarea informaţiei dintre calculatoare, hub-ul, când primeşte semnalul/datele, îl/le transmite automat către toate porturile conectate, mesajul este primit de staţia-destinaţie, iar celelalte staţii închid semnalul.

Al doilea caz:

Am creat o reţea de calculatoare formată din 6 calculatoare şi 2 switch-uri.

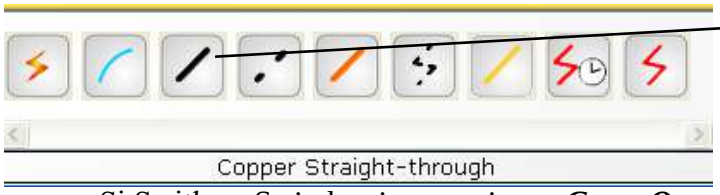


Switch - pentru conectarea celor 3 calculatoare

Switch - pentru conectarea celorlalte 3 calculatoare

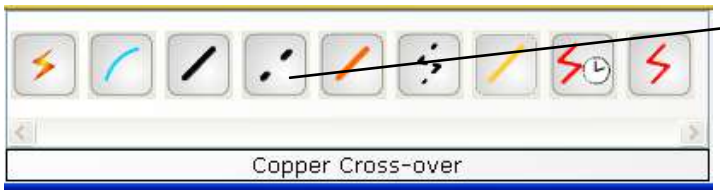
PC(calculatoare)

La fel, conectăm calculatoarele la Switch, prin conexiunea ***Straigth-through***.



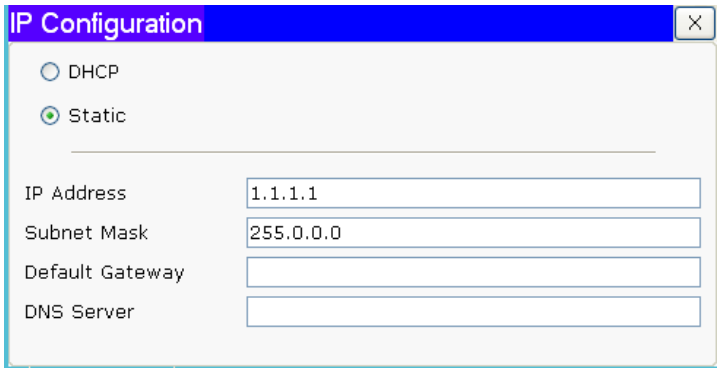
Straigth-through

Și Swith cu Switch prin conexiunea ***Cross-Over***



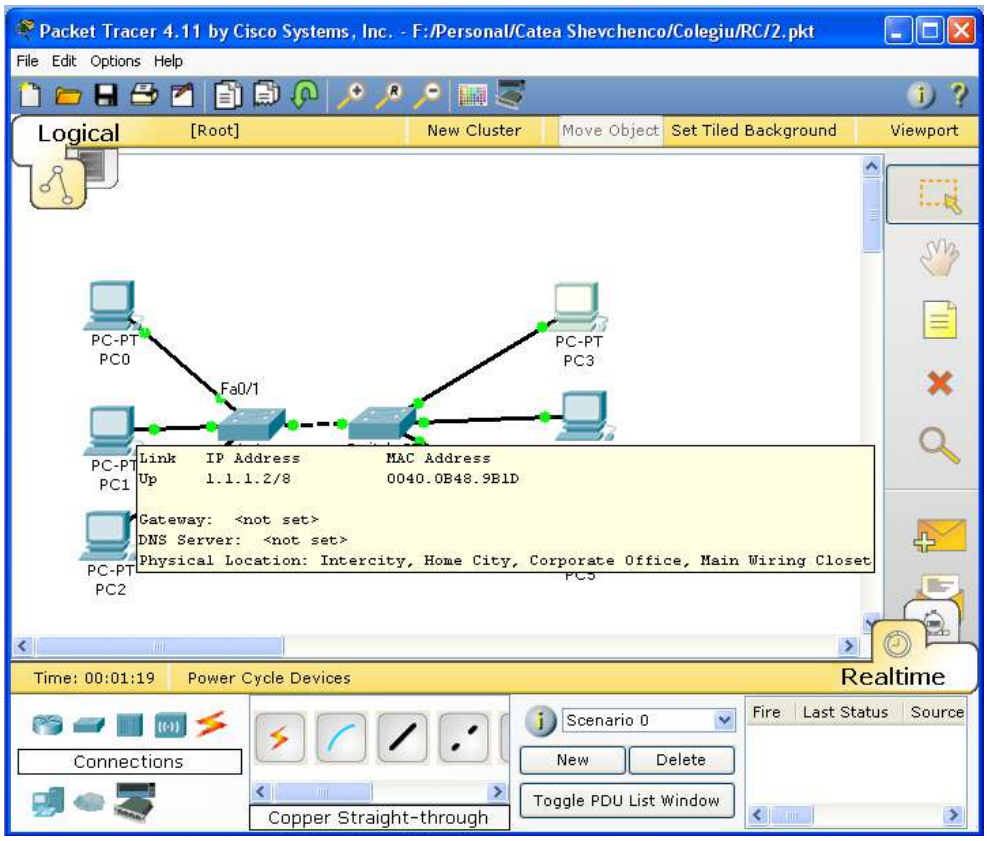
Cross-Over

Configurăm calculatoarele:

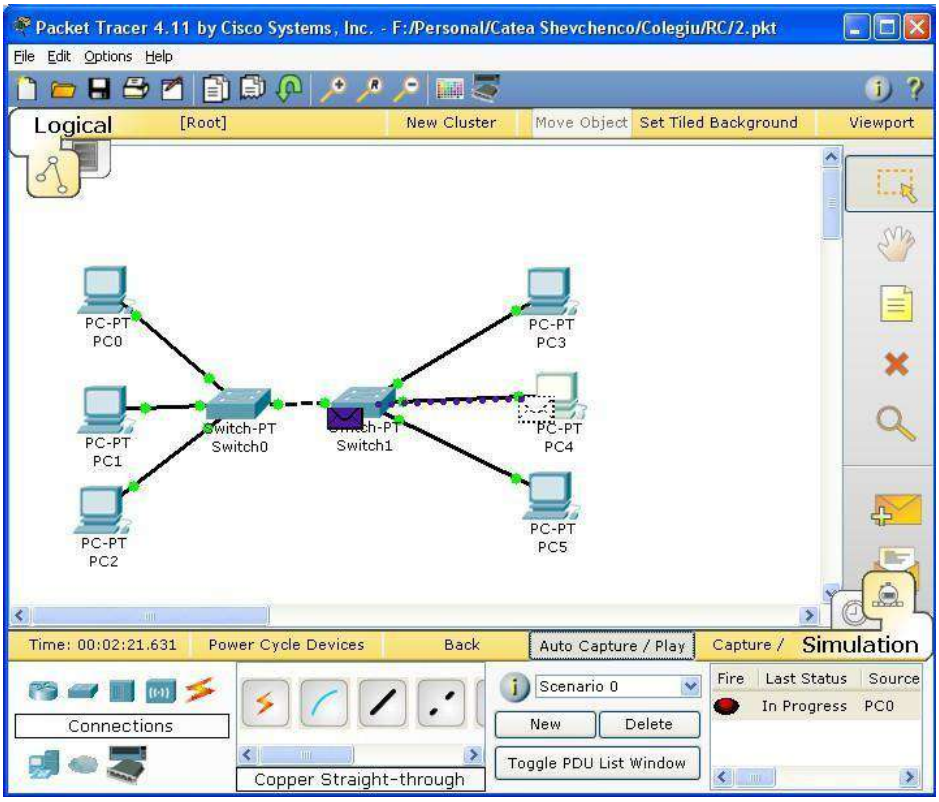


Pentru toate calculatoarele, facem acelaşi lucru.

Obţinem:



Transmiterea informaţiei:



Obs: Switch-ul, când primeşte semnalul, verifică adresa destinatarului şi o transmite pe portul căruia îi corespunde, fapt care îl şi deosebeşte de hub, care, fizic se aseamănă cu un switch, switch-ul dispunând de o tabelă de adrese.

Concluzie:

În acest proiect, am avut drept ţintă crearea unei reţele de calculatoare şi utilizarea echipamentelor de reţea studiate, pentru a observa asemănările şi diferenţele dintre acestea.

La crearea reţelei, în primul caz, am utilizat 2 hub-uri şi un bridge.

Definiţie: *Hub-ul reprezintă un dispozitiv cu cel puţin 2 porturi, care asigură conexiunea dintre mai multe calculatoare punct la punct*.

În cazul nostru, am utilizat un hub cu 6 porturi; 4 porturi le-am utilizat pentru conectarea acestuia la calculatoare, unul pentru conectarea la bridge, iar pe celălalt l-am lăsat liber.

Bridge-ul (pod) – se foloseşte pentru a conecta două reţele între ele, asigurând astfel ca reţeaua să nu fie supraîncărcată.

La crearea reţelei, în cel de-al doilea caz, am utilizat 2 switch-uri.

Definiţie: *Switch-ul reprezintă un dispozitiv care asigură conexiunea dintre mai multe calculatoare.* Acesta, fizic, nu se deosebeşte de hub cu nimic, însă este mai inteligent, având o memorie în care se înmagazinează tabela de adrese MAC, care prezintă modul în care vor fi filtrate pachetele prin porturile switch-ului.

Tabela conţine:

- *Adresa MAC a device-ului*
- *VLAN-ul care-l conţine*
- *Portul asociat*
- *Modul în care e adăugată areas MAC:* ***dynamic*** *(programare automată a switch-ului) sau* ***static*** *(introdusă manual în STATIC MAC FORWARDING)*

Astfel, la transferul informaţiei de la un calculator la altul, am observat, în cazul hub-ului, că acesta, atunci când primeşte semnalul de la un calculator, îl transmite automat către toate porturile conectateș mesajul este primit de staţia-destinaţie, celelalte staţii nimicesc semnalul, pe când switch-ul verifică adresa destinatarului şi o transmite pe portul căruia îi corespunde.

La conectarea calculatorului la hub (switch), am utilizat conexiunea: ***Straith-through.***

La conectarea switch-ului (hub-ului) la alt switch (bridge), am utilizat conexiunea: ***Cross-over.***

În concluzie, putem nota faptul că, pentru crearea unei reţele de calculatoare, este necesară, în primul rând, folosirea echipamentelor de reţea şi cunoaşterea rolului acestora.

CAPITOLUL 2. APLICAŢII ÎN REŢELE

2.1. POŞTA ELECTRONICĂ

Poşta electronică serveşte la transferul de mesaje electronice între utilizatorii aflaţi pe sisteme diferite. Protocolul a fost creat iniţial pentru mesaje text şi a fost modificat ulterior pentru a permite transferul de fişiere cu conţinut arbitrar. Sistemul este conceput în ideea că este acceptabil ca transferul mesajului să dureze câteva ore, pentru a putea funcţiona pe sisteme ce nu dispun de o legătură permanentă în reţea.

Poşta electronică este una dintre primele aplicaţii ale reţelelor de calculatoare şi a fost dezvoltată în aceeaşi perioadă cu Internet-ul. Ca urmare, protocolul cuprinde prevederi create în ideea transferului prin alte mijloace decât o legătură prin protocol Internet.

Arhitectura sistemului cuprinde următoarele elemente (fig. 24):

• Un proces de tip *mail user agent* (*MUA*), controlat de utilizatorul ce doreşte expedierea mesajului. Acesta interacţionează cu utilizatorul pentru a-l asista în compunerea mesajului şi stabilirea adresei destinatarului. La final, *mail user agent*-ul trimite mesajul unui proces de tip *mail transfer agent* (*MTA*). Transferul este iniţiat de *MUA*-ul utilizatorului expeditor şi utilizează protocolul SMTP .

• O serie de procese de tip *mail transfer agent* care trimit fiecare următorului mesajul. Transferul este iniţiat de *MTA*-ul emiţator şi utilizează tot protocolul SMTP.

• De fiecare adresă destinaţie este răspunzător un *mail transfer agent* care memorează mesajele destinate acelei adrese.

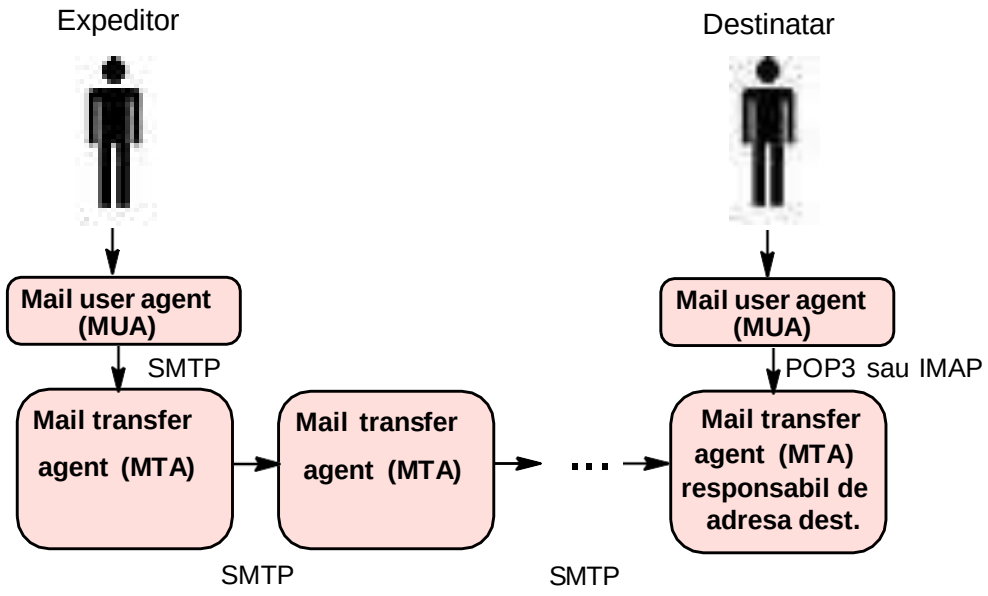


Fig. 24. Elementele sistemului de transmitere a poştei electronice.

Săgeţile arată sensul în care se iniţiază comunicaţia (de la client spre server), nu sensul în care se transferă mesajul de poştă electronică (*mail transfer agent*-ul răspunzător de adresa destinaţie, el este memorat local)

• Utilizatorul destinaţie citeşte mesajul cu ajutorul unui proces de tip *mail user agent*. Acesta contactează *mail transfer agent*-ul responsabil de adresa utilizatorului destinaţie şi recuperează mesajul de la el. Transferul este iniţiat de *MUA* (adică de receptor). Există două protocoale utilizate pentru transfer: POP3 şi IMAP.

2.1.1. Formatul mesajelor

Formatul mesajelor este definit în RFC 2822, 2001 (care înlocuieşte „clasicul" RFC 822, 1982). Fiecare mesaj începe cu un antet cuprinzând adresa expeditorului, adresa destinatarului, data şi alte câteva informaţii. După antet este notat corpul mesajului, care conţine mesajul propriu-zis.

Întreg mesajul este de tip text ASCII. Rândurile sunt delimitate prin secvenţe formate din caracterul *carriage return* (cod 13), urmat de *line feed* (cod 10). Rândurile nu au voie să aibă lungime mai mare de 998 caractere şi se recomandă să nu aibă mai mult de 78 de caractere. Caracterele de control (cu codul între 0 şi 31 sau egal cu 127) nu sunt permise, cu excepţia secvenţelor *carriage retur, line feed* care separă rândurile. În particular, caractere *carriage return* izolate sau caractere *line feed* izolate nu sunt permise.

Pentru a permite transmiterea mesajelor prin linii seriale incapabile să transmită caractere de 8 biţi (ci doar caractere de 7 biţi), este recomandabil ca mesajele să nu conţină caractere cu codul între 128 şi 255.

Datorită unor protocoale folosite pentru transmiterea şi pentru stocarea mesajelor, se impun următoarele restricţii suplimentare asupra conţinutului mesajelor:

• niciun rând să nu constea doar într-un caracter punct;

• un rând ce urmează după un rând vid să nu înceapă cu termenul From.

De menţionat că, în protocolul de comunicaţie dintre două *mail transfer agent*-uri, sunt transferate informaţii privind adresa expeditorului şi adresa destinatarului, independente de cele plasate în antetul mesajului. Aceste informaţii formează aşa-numitul *plic* (engl. *envelope*) al mesajului. Expeditorul şi destinatarul daţi în antetul mesajului sunt informaţii pentru utilizatorul uman; informaţiile de pe *plic* sunt cele utilizate efectiv în transmiterea mesajului.

2.1.1.1. Antetul mesajelor

Antetul mesajelor este constituit dintr-un număr de *câmpuri*, fiecare câmp având un *nume* şi o *valoare*. În principiu, fiecare câmp este un rând separat conţinând numele, caracterul *două puncte* (:) şi valoarea; secvenţa *carriage return* urmată de *line feed*, acţionează ca separator între câmpuri. Antetul se termină cu două secvenţe *carriage return, line feed* consecutive.

Dacă un câmp este prea lung pentru a încăpea într-un rând (standardul recomandă ca rândurile să nu depăşească 78 de caractere şi interzice rândurile mai lungi de 998 caractere), câmpul poate fi continuat pe rândurile următoare, care trebuie să înceapă cu spaţiu sau *tab*; spaţiile şi *tab*-urile de la începutul rândurilor următoare se consideră ca făcând parte din câmp.
Exemplul 2.1: Un posibil document (vezi mai jos explicaţiile privind semnificaţiile diverselor câmpuri):

From: Radu Lupsa <rlupsa@cs.upgploiesti.ro> To: Test User <test@test.com>
Date: Sat, 1 Sep 2007 10:12:20 +0300
Message-ID: my-emailer.20070901101220.53462@nessie.cs.upgploiesti.ro
Subject: Un mesaj dat ca exemplu
Salut,
Mesajul acesta este doar un exemplu.

Principalele câmpuri ce pot apărea într-un mesaj sunt:

• *To, Cc* şi *Bcc* reprezintă adresele la care trebuie livrat mesajul.

Adresele din câmpul *To* reprezintă persoanele cărora le este adresat mesajul.

Adresele din câmpul *Cc* reprezintă persoane ce trebuie informate de trimiterea mesajului; de exemplu, în corespondenţa oficială dintre un angajat al unei firme şi un client al firmei, angajatul va pune adresa clientului în câmpul *To* (deoarece acestuia îi este destinat mesajul), iar în câmpul *Cc* va pune adresa şefului (care trebuie informat cu privire la comunicaţie).

Adresele din câmpul *Bcc* reprezintă persoane cărora le va fi livrat mesajul, fără ca ceilalţi destinatari să fie informaţi despre aceasta. Câmpul *Bcc* este completat de către *mail user agent* şi este eliminat de către primul *mail transfer agent* de pe traseu.

• *From, Sender* şi *Reply-to* reprezintă adresa expeditorului şi adresa la care trebuie răspuns.

În condiţii obişnuite, un mesaj conţine doar câmpul *From*, reprezentând adresa expeditorului mesajului şi, totodată, adresa la care trebuie trimis un eventual răspuns.

Câmpul *Sender* este utilizat atunci când o persoană trimite un mesaj în numele altei persoane sau în numele unei organizaţii pe care o reprezintă. Există două situaţii practice care conduc la această situaţie: dacă mesajul provine de la şef, dar este scris şi trimis efectiv de secretara şefului, atunci adresa şefului este pusă în câmpul *From,* iar adresa secretarei în câmpul *Sender*. Dacă o persoană trimite un mesaj în numele unei organizaţii, atunci adresa organizaţiei va fi trecută în câmpul *From* şi adresa persoanei ce scrie mesajul va fi plasată în câmpul *Sender*.

Câmpul *Reply-to* reprezintă adresa la care trebuie trimis un eventual răspuns la mesaj, dacă această adresă este diferită de adresa din câmpul *From*. Dacă destinatarul răspunde la un mesaj primit, utilizând funcţionalitatea de *reply* a *mail user agent*-ului său, *MUA*-ul oferă implicit, ca adresă destinaţie a mesajului de răspuns, adresa preluată din campul Reply-to a mesajului original. În lipsa unui câmp Reply-to, *MUA*-ul oferă adresa din câmpul *From*. De asemenea, chiar în prezenţa unui câmp Reply-to, este bine ca *MUA*-ul să ofere posibilitatea de a trimite răspunsul la adresa din câmpul *From*.

Un exemplu de utilizare a câmpului *Reply-to* este următorul: un mesaj adresat unei liste de discuţii are ca *From* adresa autorului mesajului şi ca *To* adresa listei. Mesajul se transmite centrului de distribuţie al listei (un *mail transfer agent*), care retransmite mesajul către abonaţii listei. Mesajul retransmis către abonaţi va avea adăugat un câmp *Reply-to* indicând adresa listei. Astfel, mesajul primit de abonat are *From* autorul mesajului, *To* adresa listei şi *Reply-to* tot adresa listei. Abonatul listei va răspunde foarte uşor pe adresa listei deoarece, *mail user agent*-ul său va prelua adresa listei din *Reply-to* şi o va pune ca adresă destinaţie în mesajul de răspuns. Totuşi, este bine ca utilizatorul să nu folosească orbeşte această facilitate, întrucât, uneori, răspunsul este bine să ajungă doar la autorul mesajului original, nu la toată lista.

• *Received* şi *Return-path* au ca rol diagnosticarea sistemului de livrare a mesajelor. Fiecare *mail transfer agent* de pe traseul mesajului adaugă în faţa mesajului un câmp *Received* în care scrie numele maşinii sale, numele şi adresa IP a *mail transfer agent*-ului care i-a trimis mesajul, data şi ora curentă şi emiţătorul şi destinatarul mesajului, conform cererii *mail transfer agent*-ului care îi transmite mesajul (cei indicaţi pe „plicul“ mesajului, nu cei din câmpurile *To* sau *From*). Astfel, un mesaj începe cu un şir de antete Received conţinând, în ordine inversă, nodurile prin care a trecut mesajul.

Ultimul *mail transfer agent* adaugă un câmp Return-path, conţinând adresa sursă a mesajului conform *plicului* (datele furnizate de MTA-ul sursă).

Câmpurile *Received* şi *Return-path* sunt utilizate pentru a returna un mesaj la autorul său, în cazul în care apar probleme la livrarea mesajului. De notat diferenţa între *Reply-to*, care reprezintă destinatarul unui eventual răspuns dat de utilizator la mesaj, şi *Return-path*, care reprezintă destinatarul unui eventual mesaj de eroare.

• *Date* reprezintă data generării mesajului. Este, în mod normal, completat de *mail user agent*-ul expeditorului mesajului. Are un format standard, cuprinzând data şi ora locală a expeditorului, precum şi indicativul fusului orar pe care se găseşte expeditorul. Formatul cuprinde: ziua din săptămână (prescurtare de trei litere din limba engleză), un caracter virgulă, ziua din lună, luna (prescurtarea de trei litere din limba engleză), anul, ora, un caracter *două puncte,* minutul, opţional încă un caracter *două puncte* urmat de secundă şi, în final, indicativul fusului orar. Indicativul fusului orar cuprinde diferenţa, în ore şi minute, dintre ora locală şi ora universală coordonată. Faptul că ora locală este oră de vară sau nu, apare în indicativul de fus orar. România are în timpul iernii ora locală egală cu UTC+2h, iar în timpul verii UTC+3h.

• *Subject* reprezintă o scurtă descriere (cât mai sugestivă) a mesajului, dată de autorul mesajului.

• *Message-ID, In-reply-to, Reference* servesc la identificarea mesajelor.

Valoarea câmpului *Message-ID* este un şir de caractere care identifică unic mesajul. El este construit de către *mail user agent*-ul expeditorului pornind de la numele calculatorului, de la ora curentă şi de la nişte numere aleatoare, astfel încât să fie extrem de improbabil ca două mesaje să aibă acelaşi *Message-ID.*

În cazul răspunsului la un mesaj prin funcţia *reply* a *mail user agent*-ului, mesajul de răspuns conţine un câmp *In-reply-to* având ca valoare *Message-ID-ul* mesajului la care se răspunde. Valoarea câmpului *Reference* se creează din câmpurile *Reference* şi *Message-ID* ale mesajului la care se răspunde. Câmpurile Reference şi Message-ID pot fi folosite, de exemplu pentru afişarea, de către *mail user agent*-ul destinaţie, a succesiunilor de mesaje legate de o anumită problemă şi date fiecare ca replică la precedentul.

• *Resent-from, Resent-sender, Resent-to, Resent-cc, Resent-bcc, Resent-date* şi *Resent-msg-id* sunt utilizate dacă destinatarul unui mesaj doreşte să retrimită mesajul, fără modificări, către altcineva. O astfel de retrimitere se poate efectua printr-o comandă *bounce* sau *resend* a *MUA*-ului. În acest caz, câmpurile din antetul mesajului original (inclusiv *From, To* sau *Date*) sunt păstrate, reflectând expeditorul, destinatarii şi data trimiterii mesajului original. Pentru informaţiile privind transmiterea (adresa utilizatorului ce efectuează retransmiterea, destinatarii mesajului retransmis, data retransmiterii etc.), se utilizează câmpurile *Resent….* Semnificaţia fiecăruia dintre aceste câmpuri *Resend….* este identică cu cea a câmpului fără *Resent-* corespunzător, dar se referă la retransmitere, nu la mesajul original.

2.1.1.2. Extensii MIME

Standardul original pentru mesaje de poştă electronică (RFC 822) a suferit o serie de extensii. Acestea sunt cunoscute sub numele *Multipur-pose Internet Mail Extension* (*MIME*) şi sunt descrise în [RFC 2045, 1996], [RFC 2046, 1996] şi [RFC 2047, 1996].
Extensiile MIME servesc în principal pentru a putea transmite fişiere ataşate unui mesaj de poştă electronică.

Un mesaj conform standardului *MIME* trebuie să aibă un câmp în antet cu numele Mime-version. Valoarea câmpului este versiunea standardu-lui *MIME* în conformitate cu care a fost creat mesajul.

2.1.1.3. Ataşarea fişierelor şi a mesajelor

Standardul *MIME* oferă posibilitatea ataşării unor fişiere la un mesaj de poştă electronică, mecanismul permitând formarea unui mesaj din mai multe părţi. Un mesaj cu ataşamente este dat în exemplul 2.2.

Fiecare mesaj are în antet un câmp, Content-type, care arată ce tip de date conţine şi în ce format sunt ele reprezentate. Exemple de tipuri sunt: text/plain (text normal), text/html (document HTML), image/jpeg (imagine în format *JPEG*) etc.

De regulă, partea din faţa caracterului *slash* (/) arată tipul de document, iar partea a doua arată formatul (codificarea) utilizată. Astfel, o imagine va avea Content-type de forma image/*format* ; de exemplu, image/gif, image/jpeg etc.

Mesajele ce conţin doar text obişnuit trebuie să aibă Content-type: text/plain. Acesta este, de altfel, tipul implicit în cazul absenţei câmpului Content-type.

Mesajele cu ataşamente au Content-type: multipart/mixed. În general, un mesaj de tip multipart/*subtip* este format, de fapt, din mai multe „fişiere“ (oarecum, ca un fişier arhivă *zip*). Într-un mesaj multipart/mixed, de obicei, una dintre părţi este mesajul propriu-zis, iar fiecare dintre celelalte părţi este câte un fişier ataşat.

Fiecare parte a unui mesaj multipart are un antet şi un corp, similar cu un mesaj de sine stătător. Antetul părţii poate conţine doar câmpuri specifice *MIME*. Astfel, fiecare parte are propriul tip, care poate fi, în particular, chiar un multipart.

Cele mai importante subtipuri ale tipului multipart sunt:

• multipart/mixed înseamnă, pur şi simplu, mai multe componente puse împreună, ca un fişier arhivă.

• multipart/alternative arată că părţile sunt variante echivalente ale aceluiaşi document, în formate diferite.

Separarea părţilor unui mesaj de tip multipart se face printr-un rând ce conţine un anumit text, ce nu apare în nici una dintre părţile mesajului. Textul utilizat ca separator este plasat în câmpul Content-type după multipart/*subtip*. Este scris sub forma (utilizabilă şi în alte câmpuri şi pentru alte informaţii) unui şir boundary=*şir* situat după şirul multipart/*subtip* şi separat prin punct şi virgulă faţă de aceasta. Exemplu:

Content-type: multipart/mixed; boundary="abcdxxxx"

Corpul mesajului multipart este separat după cum urmează:

• în faţa primei părţi. precum şi între fiecare două părţi consecutive, se găseşte un rând format doar din textul de după boundary= precedat de două caractere minus (--);

• după ultima parte, se găseşte un rând format doar din textul de după boundary=. Fiecare parte a unui mesaj de tip multipart/mixed are un câmp Content-disposition [RFC 2183, 1997]. Valoarea acestui câmp arată ce trebuie să facă *mail user agent*-ul destinaţie cu partea de mesaj în care se găseşte acest antet. Valori posibile sunt:

• *inline* arată că mesajul sau partea de mesaj trebuie să fie afişată utilizatorului;

• *attachment* arată că mesajul sau partea de mesaj nu trebuie afişată decât la cerere. Un astfel de câmp poate avea în continuare o informaţie suplimentară: filename = *nume*, arată numele sugerat pentru salvarea părţii respective. De notat că, din motive de securitate, *mail user agent*-ul destinaţie trebuie să nu salveze orbeşte partea de mesaj sub numele extras din mesaj, ci să ceară mai întâi permisiunea utilizatorului. În caz contrar, un adversar poate să trimită un fişier având ataşat un anumit fişier, cu care să suprascrie un fişier al destinatarului.

2.1.1.4. Codificarea corpului mesajului şi a ataşamentelor

Standardul original al formatului mesajelor prevede că mesajele conţin doar text ASCII, cu utilizare restricţionată a caracterelor de control. Singurele caractere de control (cele cu codurile între 0 şi 31) admise sunt *carriage return* (cod 13) şi *line feed* (cod 10), utilizate pentru separarea rândurilor din mesaj. De asemenea, se recomandă să nu se utilizeze caracterele cu coduri între 128 şi 255, din cauza imposibilităţii transmisiei lor în unele sisteme. Ca urmare, transmiterea unui fişier arbitrar (inclusiv a unui text ISO-8859) nu este posibilă direct.

Transmiterea unui conţinut arbitrar se face printr-o recodificare a acestuia, utilizând doar caracterele permise în corpul mesajului. Ca urmare, mesajele apar, faţă de *mail transfer agent*-uri, identice cu cele conforme formatului original. Un *mail user agent* vechi, conform

standardului original, nu va fi capabil să transmită un mesaj cu extensii MIME, iar în cazul primirii unui astfel de mesaj, îl va afişa într-un mod mai puţin inteligibil pentru utilizator.

Recodificarea este aplicată doar asupra corpului mesajului, nu şi asupra antetului. Antetul conţine un câmp, Content-transfer-encoding, a cărui valoare arată dacă şi ce recodificare s-a aplicat asupra conţinutului. Codificările definite sunt:

• 7bit – înseamnă, de fapt, lipsa oricărei recodificări. În plus, arată că mesajul nu conţine decât caractere ASCII (cu codurile cuprinse între 0şi 127).

• 8bit - arată că mesajul nu a fost recodificat, dar poate conţine orice caractere (cu coduri între 0 şi 255).

• quoted-printables - arată că fiecare caracter de control şi fiecare caracter egal (=) a fost recodificat ca o secvenţă de trei caractere, formată dintr-un caracter egal (=), urmat de două cifre hexa; acestea din urmă reprezintă codul caracterului original. De exemplu, caracterul egal se recodifică =3D, iar caracterul *escape* (cod 27) se recodifică =1B. Restul caracterelor pot fi scrise direct sau pot fi recodificate ca şi caracterele speciale; de exemplu, litera *a* poate fi scrisă a sau =61.

• base64 - corpul mesajului a fost recodificat în baza 64.

În lipsa vreunui antet Content-transfer-encoding, se presupune codificarea 7bit.

Pentru un mesaj (sau o parte de mesaj) de tip multipart, mesajul (respectiv partea) nu este permis să fie codificat decât 7bit sau 8bit, însă fiecare parte a unui multipart poate fi codificată independent de restul. În mod curent, un mesaj cu ataşamente are corpul mesajului codificat 7bit, partea corespunzătoare mesajului propriu-zis este codificată 7bit sau quoted- printables, iar părţile corespunzătoare ataşamentelor sunt codificate base64.

Exemplul 2.2: Un mesaj cu fişiere ataşate este dat în continuare:

From: Radu Lupsa <rlupsa@cs.upgploiesti.ro> To: Test User <test@cs.upgploiesti.ro>
Date: Sat, 1 Sep 2007 10:12:20 +0300
Message-ID: my-emailer.20070901101220.53462@nessie.cs.upgploiesti.ro
Subject: Un mesaj cu fişiere ataşate
MIME-Version: 1.0
Content-transfer-encoding: 7bit
Content-type: multipart/mixed; boundary="qwertyuiop"
--qwertyuiop
Content-type: text/plain
Content-transfer-encoding: 7bit
Content-disposition: inline

Acesta este mesajul propriu-zis.

--qwertyuiop

Content-type: application/octet-stream

Content-disposition: attachment; filename="test.dat" Content-transfer-encoding: base64 eAAXLRQ= --qwertyuiop

Content-type: text/plain; charset=utf-8

Content-disposition: attachment; filename="test.txt" Content-transfer-encoding: quoted-printables

=C8=98i =C3=AEnc=C4=83 un text. qwertyuiop

2.1.2. Transmiterea mesajelor

Aşa cum am văzut, mesajele sunt transmise din aproape în aproape, fiecare mesaj parcurgând un lanţ de *MTA*-uri. Fiecare *MTA*, cu excepţia ultimului, determină următorul *MTA* şi-i pasează mesajul.

2.1.2.1. Protocolul SMTP

Protocolul utilizat pentru transmiterea mesajelor de la un *MTA* la următorul este protocolul *Simple Mail Transfer Protocol - SMTP*. Este un protocol de tip text, construit de obicei peste o conexiune TCP.

Rolul de client SMTP îl are *MTA*-ul ce are mesajul de trimis mai departe; rolul de server aparţine *MTA*-ului ce primeşte mesajul. Clientul deschide o conexiune TCP către portul 25 al serverului.

După deschiderea conexiunii, serverul trimite un mesaj conţinând un cod de răspuns urmat de numele serverului. În continuare, clientul trimite câte o cerere, la care serverul răspunde cu un cod ce arată dacă cererea a fost executată cu succes sau nu, urmat de un text explicativ. Cererile sunt formate, de regulă, dintr-un cuvânt-cheie, urmat de eventuali parametri şi se încheie printr-o secvenţă *carriage return-line feed*. Răspunsurile sunt formate dintr-un număr, transmis ca secvenţă de cifre, urmat de un text explicativ. Numărul arată, într-o formă uşor de procesat de către calculator, dacă cererea s-a executat cu succes sau nu şi cauza erorii. Textul cuprinde informaţii suplimentare pentru diagnosticarea manuală a transmiterii mesajelor. Clientul trebuie să înceapă printr-o cerere HELO care are ca parametru numele maşinii clientului. Prin aceasta, clientul se identifică faţă de server. De notat că nu există posibilitatea

autentificării clientului de către server; serverul este nevoit să „ia de bun“ numele transmis de client.

După identificarea prin cererea HELO, clientul poate transmite serverului mai multe mesaje de poştă electronică.

Transmiterea fiecărui mesaj va începe printr-o cerere MAIL FROM:, având ca parametru adresa expeditorului mesajului. După acceptarea de către server a comenzii MAIL FROM:, clientul va trimite una sau mai multe cereri RCPT TO:; fiecare cerere are ca parametru o adresă destinaţie. Fiecare cerere RCPT TO: poate fi acceptată sau refuzată de către server, independent de celelalte. Serverul va transmite mesajul fiecăreia dintre adresele destinaţie acceptate. În final, clientul trimite o cerere DATA fără parametrii. Serverul răspunde, în mod normal, cu un cod de succes. În caz de succes, clientul trimite corpul mesajului, încheiat cu un rând pe care se găseşte doar caracterul punct. După primirea corpului mesajului, serverul trimite încă un răspuns.

Mesajele primite de MTA sunt plasate într-o coadă de aşteptare, stocată, de obicei, în fişiere pe disc. MTA-ul receptor încearcă imediat să trimită mai departe fiecare mesaj primit. Dacă trimiterea mai departe nu este posibilă imediat, mesajul este păstrat în coadă şi MTA-ul reîncearcă, periodic, să-l trimită. După un număr de încercări eşuate sau în cazul unei erori netemporare (de exemplu, dacă nu există adresa destinaţie), MTA-ul abandonează şi încearcă trimiterea unui mesaj (e-mail) de eroare înapoi către expeditor.

Dacă adresa destinaţie este locală, MTA-ul plasează mesajul în fişierul corespunzător cutiei poştale a destinatarului.

De notat că, în trimiterea mai departe, livrarea locală sau trimiterea unui mesaj de eroare, MTA-ul utilizează doar informaţiile de pe *plic*, adică parametrii comenzilor MAIL FROM: şi RCPT TO:, şi nu valorile câmpurilor From sau To din antetul mesajului.

Exemplul 2.3: Fie mesajul din exemplul 2.1. Transmiterea lui de la MTA-ul de pe cs.upgploiesti.ro catre test.com decurge astfel (rândurile transmise de la cs.upgploiesti.ro la test.com sunt precedate de o săgeată la dreapta, iar rândurile transmise în sens invers, de o săgeată la stânga):

← 220 test.com
→ HELO nessie.cs.upgploiesti.ro
← 250 test.com
→ MAIL FROM: <rlupsa@cs.upgploiesti.ro>
← 250 Ok
→ RCPT TO: <test@test.com>

← 250 Ok

→ DATA

← 354 End data with <CR><LF>.<CR><LF>

→ From: Radu Lupsa <rlupsa@cs.upgploiesti.ro>

→ To: Test User <test@test.com>

→ Date: Sat, 1 Sep 2007 10:12:20 +0300

→ Message-ID: my-emailer.20070901101220.53462@nessie.cs.upgploiesti.ro

→ Subject: Un mesaj dat ca

→ exemplu

→ Salut,

→ Mesajul acesta este doar un exemplu.

→ .

← 250 Ok: queued

→ QUIT

← 221 Bye

Exemplul 2.4: Următorul mesaj ilustrează utilizarea câmpurilor în cazul unui mesaj transmis unei liste de utilizatori. Mesajul este reprodus aşa cum ajunge la un abonat al listei, având adresa test@test.com. Mesajul ilustrează, de asemenea, utilizarea câmpului Sender de către cineva care transmite un mesaj în numele altcuiva şi a câmpului Cc.

Return-path: errors-26345@comunitati-online.test
Received: from server27.comunitati-online.test
by test.com for test@test.com; 31 Aug 2007 22:09:23 -0700
Reply-to: Forumul OZN<ozn@comunitati-online.test> Received: from roswell.greenmen.test
by server27.comunitati-online.test
for ozn@comunitati-online.test; 1 Sep 2007 05:09:21 +0000
Received: from localhost by roswell.greenmen.test
for ozn@comunitati-online.test; 1 Sep 2007 08:09:20 +0300
From: Organizatia omuletilor verzi <office@greenmen.test>Sender: Ion Ionescu <ion@greenmen.test>
To: Forumul OZN <ozn@comunitati-online.test>
Cc: Organizatia omuletilor verzi <office@greenmen.test> Date: Sat, 1 Sep 2007 10:12:20 +0300
Message-ID: my-emailer.20070901101220.534@roswell.greenmen.test

In-reply-to: my-emailer.20070830222222.321@ufo.test

Subject: Re: Incident
Organizatiei omuletilor verzi anunta ca nu a avut niciun amestec în incidentul de la balul anual E.T.

Ion Ionescu,
Presedintele Organizatiei omuletilor verzi

2.1.3. Securitatea poştei electronice

Principalele probleme privind securitatea sunt:

• *spoofing*-ul - falsificarea adresei sursă (From sau Sender);

• *spam*-urile - mesaje, de obicei, publicitare, trimise unui număr mare de persoane şi fără a fi legate de informaţii pe care destinatarii le-ar dori;

• viruşii - programe executabile sau documente, ataşate unui mesaj electronic, a căror execuţie sau respectiv vizualizare duce la trimiterea de mesaje electronice către alţi destinatari.

Falsificarea adresei sursă este extrem de simplă, deoarece, în transmiterea obişnuită a mesajelor, nu este luată nicio măsură de autentificare a expeditorului.

Falsificarea adresei sursă (*spoofing*) poate fi depistată în anumite cazuri examinând câmpurile Received din antetul mesajului şi verificând dacă există neconcordanţe între numele declarat prin HELO a unui client SMTP şi adresa sa IP sau neconcordanţe între numele primului MTA şi partea de domeniu din adresa expeditorului. De notat că anumite neconcordanţe pot fi legitime, în cazul în care căsuţele poştale dintr-un domeniu sunt ţinute de un calculator al cărui nume nu face parte din acel domeniu .

O metodă mai eficientă pentru depistarea falsurilor este utilizarea semnăturii electronice. Există două standarde de semnătură electronică utilizate, OpenPGP [RFC 2440, 2007, RFC 3156, 2001] şi S-MIME [S/MIME,].

Verificarea semnăturii necesită însă ca destinatarul să dispună de cheia publică autentică a expeditorului. Până la generalizarea utilizării mesajelor semnate, sistemul de poştă electronică trebuie să asigure livrarea mesajelor nesemnate şi, în consecinţă, cu risc de a fi falsificate.

Uşurinţa falsificării adresei sursă şi uşurinţa păstrării anonimatului autorului unui mesaj a dus la proliferarea escrocheriilor. Adesea escrocheriile constau în trimiterea de mesaje unui număr mare de utilizatori (acest fapt, în sine, este *spam*) în speranţa de a găsi printre aceştia unii care să se lase păcăliţi.

Spam-urile dăunează deoarece consumă în mod inutil timpul destinatarului. În plus, există riscul ca un mesaj legitim, „îngropat“ între multe spam-uri, să fie şters din greşeală.

Există detectoare automate de spam-uri, bazate pe diferite metode din domeniul inteligenţei artificiale. Astfel de detectoare se instalează pe MTA-uri şi resping sau marchează prin antete speciale mesajele detectate ca spam-uri. Un MTA care detectează şi respinge sau marchează spam-urile se numeşte *filtru anti-spam* .

De menţionat că filtrele anti-spam nu pot fi făcute 100% sigure, deoarece nu există un criteriu clar de diferenţiere. Ca urmare, orice filtru anti-spam va lăsa să treacă un anumit număr de spam-uri şi există şi riscul de a respinge mesaje legitime.

Majoritatea furnizorilor de servicii Internet nu permit, prin contract, clienţilor să trimită spam-uri şi depun eforturi pentru depistarea şi penalizarea autorilor. În acest scop, ei primesc sesizări şi întreţin liste cu adresele de la care provin spam-urile (*liste negre-blacklist*).

Trimiterea spam-urilor necesită recoltarea, de către autorul spam-urilor, a unui număr mare de adrese valide de poştă electronică. Acest lucru se realizează cel mai uşor prin căutarea, în paginile web, a tot ceea ce arată a adresă de poştă electronică. Contramăsură la această recoltare este scrierea adreselor, din paginile web, doar în forme dificil de procesat automat - de exemplu, ca imagine (într-un fişier *jpeg*, *gif* sau *png*).

Termenul *virus* poate desemna mai multe lucruri, înrudite, dar distincte. În general, un virus informatic este un fragment de program a cărui execuţie duce la inserarea unor copii ale sale în alte programe de pe calculatorul pe care se execută virusul. Impropriu, prin *virus* se mai desemnează un fragment, inserat într-un program util, care execută acţiuni nocive utilizatorului în contul căruia se execută acel program. Denumirea corectă pentru un astfel de program este aceea de *cal troian*. Denumirea de *virus* poate fi dată, corect, doar fragmentelor de program capabile să se reproducă (să-şi insereze copii în alte programe).

În contextul poştei electronice, un *virus* este un fragment dintr-un program plasat ca fişier ataşat la un mesaj electronic. Virusul se poate reproduce fie prin mijloace independente de poşta electronică, fie prin expedierea, către alţi utilizatori, a unor copii ale mesajului. În acest al doilea caz, virusul utilizează, de obicei, adrese extrase din lista, ţinută de MUA, a adreselor partenerilor de corespondenţă ai utilizatorului care primeşte mesajul virusat.

Indiferent de forma de propagare (infectarea fişierelor locale sau transmiterea de mesaje spre alţi utilizatori), pentru a-şi realiza scopul, un virus trebuie să ajungă să determine execuţia, cu drepturile utilizatorului victimă, a unei secvenţe de instrucţiuni aleasă de autorul virusului.

Acest lucru se poate întâmpla în două moduri:

• Virusul se găseşte într-un program executabil, pe care utilizatorul îl execută.

• Virusul este un document astfel construit încât, exploatând o eroare din programul utilizat pentru vizualizarea documentului, să determine programul de vizualizare să execute acţiunea dorită de autorul virusului.

Pentru a păcăli destinatarul şi a-l determina să execute sau să vizualizeze fişierul ataşat, corpul mesajului este construit astfel încât să câştige încrederea utilizatorului. Astfel, mesajul este adesea construit ca şi când ar proveni de la administratorul de sistem sau de la un prieten al destinatarului.

Metodele de prevenire a viruşilor de poştă electronică sunt aceleaşi ca şi metodele de prevenire a viruşilor, în general. Pentru programele executabile, dacă utilizatorul are încredere în autorul declarat al programului (de exemplu, autorul este o firmă de soft de încredere), atunci programul poate fi semnat electronic, iar utilizatorul poate verifica această semnătură pentru a se convinge de autenticitatea programului. Pentru programe provenite din surse ce nu sunt de încredere, execuţia lor se poate face într-un mediu controlat, de exemplu, dintr-un cont separat, cu drepturi minime, sau prin intermediul unui interpretor care să nu execute instrucţiunile potenţial nocive. Această din urmă abordare este utilizată de *apple* -urile Java.

Pe lângă aceste metode de prevenire, există câteva acţiuni care micşorează riscul sau consecinţele execuţiei unui virus. Una dintre ele este reducerea la minimul necesar a lucrului din contul de administrator. Altă măsură preventivă este aceea de a nu vizualiza sau executa fişierele ataşate unui mesaj suspect; pentru aceasta, este bine ca expeditorul unui mesaj să nu trimită niciodată un mesaj numai cu fişiere ataşate, ci să scrie un mic text explicativ, care să-i permită destinatarului să identifice autorul şi scopul fişierelor ataşate.

2.2. TRANSFERUL FIŞIERELOR ÎN REŢEA

Cerinţa de a transfera fişiere în reţea poate avea diferite particularităţi. Există mai multe protocoale şi mai multe aplicaţii pentru transferul fişierelor în reţea, adaptate pentru diferite necesităţi.

O primă categorie de protocoale şi aplicaţii priveşte, în principal, transferul fişierelor unui utilizator de pe o maşină pe alta, în condiţiile în care utilizatorul are cont pe ambele maşini. Protocoalele construite pentru aceasta sunt *ftp* şi *ssh*. De notat că şi poşta electronică poate servi ca mecanism de transfer de fişiere.

O a doua categorie priveşte transferul fişierelor publice de la un calculator ce stochează astfel de fişiere la calculatorul unui utilizator ce doreşte să citească fişierele respective. Iniţial se utiliza protocolul *ftp* în acest scop. Protocolul utilizat în mod curent este însă *http*.

O a treia categorie priveşte accesul proceselor de pe un calculator la fişiere stocate pe alt calculator ca şi când fişierele ar fi locale. De principiu fişierele respective sunt private, ca şi pentru prima categorie de protocoale. Protocoalele din această categorie trebuie să satisfacă două cerinţe specifice (faţă de prima categorie): să permită transferul doar a unei părţi mici dintr-un fişier şi să permită controlul partajării fişierului între procese. Protocoalele utilizate aici sunt SMB (utilizat în reţelele Windows) şi NFS.

2.2.1. Protocolul *ftp*

Descriem pe scurt conceptele de bază ale protocolului *ftp*. Pentru detalii, a se vedea [RFC 765, 1985].

Clientul deschide o conexiune TCP către portul 21 al serverului; această conexiune se numeşte *conexiune de control*. Prin conexiunea de control, clientul transmite comenzi în format text, câte o comandă pe o linie. Fiecare comandă începe cu numele comenzii urmat de eventuali parametrii. Parametrii sunt separaţi prin spaţii, atât faţă de numele comenzii cât şi între ei. Serverul răspunde tot în format text, fiecare răspuns începând cu un cod care arată dacă comanda s-a executat cu succes sau ce erori s-au produs, după care urmează un text ce descrie, în limbaj natural, rezultatul execuţiei comenzii. Cu o singură excepţie (în cazul comenzii PASV, descrisă mai jos), textul din răspuns nu este interpretat de către aplicaţia client. El este însă afişat, de obicei, pe ecran utilizatorului aplicaţiei client.

Autentificarea se face la solicitarea clientului. Clientul trimite succesiv două comenzi, USER şi PASS, având ca parametrii numele utilizatorului şi parola acestuia. Serverul refuză execuţia majorităţii comenzilor clientului înainte de autentificarea cu succes a acestuia. După autentificare, serverul acceptă să efectueze operaţiile cerute de client doar dacă utilizatorul în contul căruia s-a făcut autentificarea are dreptul la operaţiile respective. Pe sistemele de tip UNIX, reglementarea drepturilor de acces se face, de obicei, astfel: la lansare, serverul rulează din contul root; la conectarea unui client, serverul creează (prin apelul sistem fork()) un proces fiu care se ocupă de acel client; după autentificare, procesul fiu trece în contul utilizatorului autentificat (prin apelul setuid()); în continuare, serverul acceptă orice comenzi de la client şi încearcă să le execute, iar verificarea drepturilor de acces este făcută de nucleul sistemului de operare în momentul în care procesul server fiu încearcă să acceseze sistemul de fişiere.

Pentru transferul de fişiere publice, serverul este configurat să accepte autentificare cu numele de utilizator *ftp* sau *anonymous* fără să solicite parolă sau acceptând orice şir de caractere pe post de parolă. În vremurile de început ale Internet-ului, se obişnuia ca un utilizator

care dorea acces la fişiere publice să-şi dea, pe post de parolă, adresa sa de poştă electronică. O dată cu răspândirea *spam*-urilor, s-a renunţat la acest obicei.

Transferul fişierelor se cere prin comenzile SEND (dinspre client spre server) şi RETR (dinspre server spre client). Comenzile au ca argument numele de pe server al fişierului de transferat. Transferul propriu-zis are loc printr-o conexiune separată, numită *conexiune de date*.

Pentru fiecare fişier se deschide o nouă conexiune de date, care se închide la finalul transferului fişierului. Dimensiunea fişierului nu este specificată explicit nicăieri, receptorul fişierului obţinând lungimea din faptul că emiţătorul închide conexiunea de date la finalul fişierului.

Există două moduri de deschidere a conexiunii de date:

• Modul *activ* prevede că serverul deschide conexiunea de date ca o conexiune TCP dinspre portul 20 al serverului către un port specificat de client. Clientul specifică portul pe care aşteaptă conexiunea prin comanda PORT. Conexiunea se deschide ca urmare a comenzii de transfer (SEND sau RETR), nu imediat după primirea comenzii PORT.

• Modul *pasiv* prevede deschiderea conexiunii de date de către client, dinspre un port oarecare al său, către un port specificat de server. Portul specificat de server se obţine ca răspuns al comenzii PASV date de client. Acesta este singurul caz în care clientul interpretează din răspunsul serverului şi altceva decât codul returnat.

Listarea fişierelor de pe server este solicitată de client prin comanda LIST. Transferul listei de fişiere se face tot printr-o conexiune de date, ca şi în cazul comenzii RETR.

2.2.2. Protocolul HTTP

HyperText Transmission Protocol (HTTP) este un protocol elaborat pentru transferul dinspre server spre client a fişierelor cu informaţii disponibile public. El înlocuieşte protocolul *ftp* utilizat cu conectare ca utilizator *anonymous*. Deşi numele protocolului face referire la hipertext, el poate fi utilizat pentru a transfera orice fel de conţinut.

Protocolul de bază constă în trimiterea de către client a unei cereri, în care informaţia principală este numele fişierului cerut. Răspunsul serverului conţine nişte informaţii despre fişier şi conţinutul efectiv al fişierului. Implicit, conexiunea se închide după transferul unui fişier. Dacă clientul doreşte mai multe fişiere de pe acelaşi server, va trebui să deschidă fiecare fişier. Protocolul a fost însă extins, ajungând să fie folosit ca protocol de transfer de date pentru aplicaţii de orice tip.

2.2.2.1. Structura cererilor şi a răspunsurilor

Formatul comunicaţiei este mixt, atât la cereri, cât şi la răspunsuri. Partea de început este text, iar conţinutul fişierului este binar.

Cererea cuprinde, pe prima linie, un cuvânt reprezentând numele operaţiei cerute. Pentru solicitarea unui fişier public de pe server, numele este GET. După numele operaţiei urmează numele fişierului şi apoi identificarea versiunii de protocol în conformitate cu care este formată cererea. Cele trei elemente sunt separate prin câte un spaţiu.

Următoarele linii sunt de forma *nume*:*valoare*, similar cu antetul unui mesaj de poştă electronică. După ultima linie de antet urmează o linie vidă. Pentru unele tipuri de cereri, după linia goală, se găseşte un conţinut. În acest caz, una dintre liniile din antet are numele Content-length şi are ca valoare lungimea conţinutului, dată ca şir de cifre zecimale.

Răspunsul este structurat similar cu cererea. Pe prima linie se află identificatorul versiunii HTTP, număr de trei cifre şi un text. Numărul arată dacă cererea a fost satisfăcută cu succes sau nu, iar textul, neinterpretat de client, este o descriere în cuvinte a semnificaţiei codului de trei cifre. Următoarele linii sunt de forma *nume*:*valoare* şi dau informaţii despre fişierul solicitat. După ultima linie de antet urmează o linie vidă şi apoi conţinutul (binar) al fişierului. În antet se găseşte o linie cu numele Content-length, având ca valoare lungimea fişierului. Determinarea sfârşitului conţinutului propriu-zis de către client trebuie făcută prin numărarea octeţilor din partea de conţinut.

Adesea, mai multe servere *HTTP* sunt găzduite fizic pe acelaşi calculator. În acest caz, fie numele serverelor corespund, prin DNS, unor adrese IP diferite, dar aparţinând aceluiaşi calculator, caz în care serverul este configurat să răspundă în funcţie de IP-ul către care a fost deschisă conexiunea, fie numele serverelor corespund aceleiaşi adrese IP, caz în care este necesar ca în cererea *HTTP* să fie specificat serverul dorit. Acest lucru se realizează prin aceea că, în cererea clientului, se plasează un antet cu numele Host şi având ca valoare numele de server dorit.

2.2.2.2. URL-urile

O pagină web este în general un fişier scris în *HyperText Markup Language* (HTTP) şi oferit în acces public prin protocolul HTTP.

O pagină web constă, de obicei, din mai multe fişiere. Există un fişier de bază, scris în limbajul *HTML*, şi alte fişiere, conţinând anumite elemente ale paginii: imagini (în fişiere separate în formate specifice-JPEG, PNG, GIF), applet-uri (Java), specificări de formatare a

paginii (fişiere *Cascading Style Sheet* - CSS). De asemenea, o pagină conţine, în general, legături (*link*-uri) spre alte pagini. Toate acestea necesită referiri dintr-un fişier HTML către alte fişiere disponibile în acces public. Referirea acestor fişiere se face prin nume care să permită regăsirea lor uşor.

Un *Universal Resource Locator* (URL) este un nume prin care se poate identifica şi cu ajutorul cărora se poate regăsi o resursă disponibile în Internet. URL-urile au apărut ca un format standard de scriere a numelor fişierelor referite din paginile web; ele permit însă utilizări mult mai vaste.

Un URL este alcătuit, în general, din trei componente:

• *Tipul* identifică protocolul utilizat. Exemple mai cunoscute sunt: http, ftp, https, mailto.

• *Numele maşinii* este numele de domeniu sau adresa IP a maşinii pe care se găseşte resursa (fişierul).

Pe lângă numele maşinii, în cadrul acestei componente, se poate adăuga numele de utilizator în contul căruia trebuie să se autentifice un client pentru a obţine accesul dorit la resursă. Numele de utilizator se dă în faţa numelui sau adresei maşinii, separat de acesta prin caracterul @. Standardul original prevedea şi posibilitatea de a scrie în URL parola necesară conectării. Această utilizare este nerecomandată.

• *Calea* identifică resursa (fişierul) în cadrul serverului care o găzduieşte. În principiu, ea este calea completă a fişierului cerut, relativă la un director de bază, fixat, al documentelor publice.

URL-urile se pot utiliza şi se utilizează efectiv în multe alte scopuri decât identificarea paginilor web. De exemplu, sistemul *SubVersion* (SVN) utilizează URL-uri de forma svn://*maşină* /*cale* pentru a referi fişierele dintr- un *repository*.

2.2.2.3. Alte facilităţi HTTP

Antetul răspunsului HTTP oferă mai multe informaţii despre fişierul returnat:

• Tipul conţinutului fişierului este specificat de către server prin intermediul unui antet cu numele Content-type şi cu valoarea construită ca şi în cazul antetului Mime-type de la poşta electronică. Tot ca şi în cazul lui Mime-type, tipul conţinutului poate fi urmat de specificarea codificării utilizate pentru text; de exemplu, *Content-type: text/html; charset=utf-8* înseamnă că fişierul este de tip HTML, iar codificarea utilizată pentru caractere este UTF-8.

• Data ultimei modificări a fişierului este specificată prin valoarea antetului cu numele Date.

• Tipul de compresie utilizat (dacă fişierul returnat este comprimat) este dat ca valoare a antetului Content-transfer-encoding.

• Limba în care este scris textul din fişier (dacă este cazul) este returnată ca valoare a antetului Language.

Este posibil ca unui URL să-i corespundă mai multe fişiere pe server, având conţinut echivalent, dar în diverse formate, limbi sau codificări. Pentru a selecţiona varianta dorită, clientul poate anunţa posibilităţile şi preferinţele sale cu privire la tipul de fişier, limbă şi codificare. Antetele corespunzătoare, din cererea clientului, sunt: Accept, Accept-language şi Accept-encoding. Fiecare dintre acestea are ca valoare o listă de variante, în ordinea preferinţei. De exemplu: *Accept-language: ro, en, fr* solicită serverului, de preferinţă, varianta în limba română a textului. Dacă o variantă în limba română nu este disponibilă, se solicită una în limba engleză, iar în lipsa acesteia una în limba franceză.

Protocolul HTTP permite formularea de cereri condiţionate sau parţiale. O cerere parţială este utilă dacă fişierul cerut este mare şi clientul doreşte să-l aducă din bucăţi sau dacă, la o cerere precedentă, a căzut legătura după transferul unei părţi din fişier. O cerere condiţionată determină serverul să transmită clientului fişierul numai dacă este îndeplinită o anumită condiţie, cel mai adesea dacă a fost modificat mai recent decât o anumită dată specificată de client. Dacă nu este îndeplinită condiţia, serverul dă un răspuns format doar din antet, fără conţinutul propriu-zis. Această facilitate este utilă dacă clientul deţine o copie a unui fişier şi doreşte împrospătarea acesteia. Cererea parţială se specifică de către client prin intermediul antetului Range; cererea condiţionată se specifică prin antetul If-modified-since.

Pentru optimizarea traficului, în cazul în care un client doreşte mai multe fişiere de pe acelaşi server (aceasta se întâmplă adesea în cazul în care clientul aduce un fişier *html*, iar apoi are de adus imaginile şi, eventual, alte obiecte din document), este prevăzută posibilitatea de a păstra conexiunea deschisă pe durata mai multor cereri. În acest scop, clientul cere păstrarea conexiunii deschise, plasând în cerere antetul *Connection: keep-alive* ; pentru a nu permite unor clienţi să deschidă o conexiune şi apoi să o lase deschis la nesfârşit, ţinând ocupate resurse pe server, serverul trebuie configurat să închidă automat conexiunea dacă nu se transferă date o perioadă de timp.

Este util, uneori, ca un utilizator care accesează un URL să fie redirecţionat automat către alt URL. Aceasta se întâmplă dacă administratorul site-ului decide o reorganizare a paginilor şi doreşte ca utilizatorii care au reţinut URL-uri desfiinţate în urma reorganizării să fie redirecţionaţi către paginile corespunzătoare din noua organizare. Această redirecţionare se face prin trimiterea de către server a unui antet cu numele Location având drept conţinut URL-ul spre care se doreşte redirecţionarea clientului. În acest caz, programul client nu afişează

conţinutul returnat de server (conţinut care în mod normal lipseşte complet), ci cere şi afişează conţinutul de la URL-ul indicat în antetul Location.

2.2.2.4. Proxy HTTP

Un *proxy http* este un proces care, faţă de un client HTTP, acţionează aproape ca un server HTTP, iar pentru satisfacerea cererii contactează serverul solicitat de client şi acţionează, faţă de acest server, ca un client.

Un *proxy HTTP* este utilizat, în mod normal, pentru următoarele funcţii:

• dacă dintr-o reţea locală există şanse mari ca mai mulţi utilizatori să acceseze aceleaşi pagini web: în acest caz, clienţii *HTTP* ai calculatoarelor din reţea se configurează să utilizeze un acelaşi *proxy http* din reţeaua locală. Dacă există mai multe cereri pentru aceeaşi pagină, la prima cerere *proxy*-ul memorează pagina adusă, iar la următoarele cereri o serveşte clienţilor din memoria locală.

• dacă într-o reţea se utilizează adrese IP locale şi nu se doreşte configurarea unui mecanism de translaţie de adrese, în acest caz, se instalează un *proxy HTTP* pe un calculator din reţeaua locală. dar având şi adresă IP publică. Clientul accesează *proxy*-ul prin reţeaua locală, iar *proxy*-ul, având adresă publică, poate accesa fără restricţii serverul dorit.

• dacă este de dorit un control fin asupra paginilor ce pot fi accesate dintr-o reţea locală (de exemplu, pentru a restricţiona accesul angajaţilor la site-uri nelegate de activitatea lor normală), în acest caz, se configurează un *proxy* în care se configurează toate restricţiile de acces dorite. Apoi, pentru a împiedica accesul direct, prin evitarea *proxy*-ului, pe router-ul ce leagă reţeaua internă la Internet se configurează un filtru de pachete care să blocheze pachetele adresate portului TCP 80 al serverelor exterioare.

O diferenţă între protocolul de comunicaţie dintre un client şi un *proxy* faţă de protocolul client-server este că, după o cerere (de exemplu, GET), la protocolul client-server urmează calea locală din URL, iar la protocolul client-*proxy* urmează URL-ul solicitat (inclusiv numele protocolului şi numele serverului).

O a doua diferenţă este dată de existenţa unei cereri, CONNECT, ce poate fi adresată doar unui *proxy*. La primirea unei cereri CONNECT, *proxy*-ul deschide o conexiune către serverul specificat în comanda CONNECT şi apoi retrimite datele dinspre client direct spre server şi, reciproc, dinspre server înapoi spre client. În cazul unei cereri CONNECT, *proxy*-ul nu se implică mai departe în comunicaţia dintre client şi server. Ca urmare, în acest caz *proxy*- ul nu mai memorează paginile aduse şi nu permite filtrarea cererilor decât după serverul şi

portul la care se conectează, în schimb permite tunelarea oricărui protocol (nu numai a protocolului HTTP) între un client dintr-o reţea cu adrese interne şi un server din Internet.

2.2.2.5. Conexiuni securizate: SSL/TLS

SSL - *Secure Sockets Layer*, rom. *nivelul conexiunilor securizate* este un protocol pentru securizarea conexiunilor. A fost creat de firma Netscape în vederea securizării comunicaţiei între clientul şi serverul HTTP. Protocolul este însă suficient de flexibil pentru a permite oricărei aplicaţii ce comunică prin conexiuni să-l folosească. TLS [RFC 4346, 2006] – *Transport Layer Security*, rom. *securitate la nivel transport* - este derivat din SSL versiunea 3, dar dezvoltat de IETF (Internet Engineering Task Force). În cele ce urmează, vom discuta doar despre TLS, însă toate chestiunile prezentate sunt valabile şi pentru SSL.

Protocolul TLS presupune existenţa unei legături nesecurizate între un client (entitatea care iniţiază activ comunicaţia) şi un server (entitatea care aşteaptă să fie contactat de către client). Legătura nesecurizată este, în mod obişnuit, o conexiune TCP. Protocolul TLS oferă un serviciu de tip conexiune. TLS asigură confidenţialitatea şi autenticitatea datelor utile transportate. Datele utile transportate de TLS pot aparţine oricărui protocol. Protocolul ale cărui date sunt transportate ca date utile de către TLS este numit *tunelat* prin TLS.

În cadrul iniţierii unei conexiuni TLS, se realizează stabilirea unei chei de sesiune care este utilizată în continuare pentru securizarea transportului datelor utile. Autentificarea stabilirii cheii poate fi unilaterală, doar clientul autentificând serverul cu care a realizat negocierea cheii de sesiune, sau bilaterală. În cazul autentificării unilaterale, se poate utiliza o autentificare a clientului faţă de server în cadrul protocolului tunelat. În acest caz, deoarece serverul este deja autentificat, autentificarea clientului poate fi făcută prin parolă, fără riscul ca parola să fie transmisă unui adversar.

Autentificarea stabilirii cheii de sesiune se face printr-un mecanism de semnătură digitală. Pentru distribuirea sigură a cheilor publice, utilizate în cadrul autentificării, se utilizează certificate .

Serverul trebuie să aibă o pereche de chei pentru semnătură digitală şi un certificat, semnat de o autoritate în care clientul are încredere, pentru cheia publică. La iniţierea conexiunii TLS, serverul transmite clientului certificatul său. Clientul verifică faptul că numele din certificat coincide cu numele serverului la care dorea conectarea, că semnătura autorităţii de certificare asupra certificatului este validă, că autoritatea de certificare este de încredere şi, în final, utilizează cheia publică din certificatul clientului pentru a autentifica stabilirea cheii de

sesiune. Dacă se doreşte şi autentificarea clientului faţă de server tot prin TLS, atunci clientul trebuie să aibă, la rândul său, o pereche de chei şi un certificat.

În vederea verificării semnăturii din certificat şi a faptului că semnatarul (autoritatea de certificare) este de încredere, fiecare dintre parteneri trebuie să aibă o listă cu cheile autorităţilor de certificare de încredere. Cheia unei autorităţi de certificare este, în mod obişnuit, plasată tot într-un certificat.

Certificatul unei autorităţi de certificare poate fi semnat de către o altă autoritate de certificare sau chiar de către autoritatea posesoare a certificatului. În acest ultim caz, certificatul se numeşte *certificat autosemnat* (engl. *self-signed certificate*) sau *certificat rădăcină* (engl. *root certificate*). În majoritatea cazurilor, clientul are o mulţime de certificate autosemnate ale autorităţilor în care are încredere.

Există mai multe produse soft pentru crearea perechilor de chei şi pentru crearea şi semnarea certificatelor. Un astfel de produs este *OpenSSL*, disponibil pe sistemele de tip UNIX.

2.2.2.6. Utilizarea TLS pentru web

Transferul securizat al paginilor web se realizează prin tunelarea protocolului HTTP peste SSL sau TLS. Construcţia realizată se numeşte HTTPS. URL-urile resurselor accesibile prin conexiuni securizate au, ca nume al protocolului, şirul de caractere https (în loc de http).

Un navigator web care are de adus o pagină a cărei URL are, în partea de protocol, https, execută următoarele:

• Afară de cazul în care URL-ul specifică explicit un număr de port, clientul deschide conexiunea către portul 443 al serverului (în loc de portul 80, implicit pentru HTTP).

• Dacă în locul contactării directe a serverului web se utilizează un *proxy*, clientul trimite serverului *proxy* o cerere CONNECT pentru stabilirea conexiunii spre server. Prin această conexiune, stabilită prin intermediul *proxy*-ului, se transmit mesajele SSL sau TLS, în cadrul cărora se transmit datele HTTP.

• La deschiderea conexiunii (fie conexiune TCP directă, fie un lanţ de conexiuni TCP prin intermediul *proxy*-ului), are loc mai întâi schimbul de mesaje legat de stabilirea cheii SSL sau TLS. După iniţializarea conexiunii securizate, prin canalul securizat, are loc un dialog conform protocolului HTTP. Cu alte cuvinte, cererile şi răspunsurile HTTP constituie date utile pentru nivelul SSL sau TLS.

Autentificarea serverului, în cadrul protocolului TLS, necesită, aşa cum am văzut, ca navigatorul web să dispună de certificatele autorităţilor de certificare de încredere. În general, producătorii de navigatoare încorporează în acestea nişte certificate ale unor autorităţi de

atestate larg recunoscute. Utilizatorul poate însă să dezactiveze oricare dintre aceste certificate, precum şi să adauge alte certificate.

Atragem atenţia asupra unor particularităţi legate de utilizarea HTTPS:

• Deoarece autentificarea serverului, prin mecanismul TLS, se face înaintea trimiterii cererii HTTP, certificatul trimis de server nu poate depinde de antetul Host transmis de către client.

Ca urmare, dacă mai multe site-uri web securizate sunt găzduite de un acelaşi calculator, este necesar ca aceste site-uri să fie distinse prin adresa IP sau prin numărul de port. În cazul în care s-ar utiliza doar antetul Host pentru ca serverul să determine site-ul cerut de client, serverul ar trimite acelaşi certificat, indiferent de site-ul dorit de client. Ca urmare, ar exista o nepotrivire între numele din certificat şi numele site-ul solicitat de client. În consecinţă, clientul ar declara site-ul ca fiind unul fals.

• O pagină web este formată, în mod obişnuit, din mai multe obiecte, cu URL-uri diferite (pagina HTML propriu-zisă şi imaginile din pagină).

În aceste condiţii, este posibil ca, într-o aceeaşi pagină, unele dintre elemente să fie securizate şi celelalte elemente să fie nesecurizate. De asemenea, este posibil ca diferite elemente să provină de pe site-uri diferite, autentificate prin certificate diferite. Într-un astfel de caz, navigatorul web trebuie să avertizeze utilizatorul.

2.3. APLICAŢIE

Tema: *Modelarea traficului pe o placă de reţea a unui PC cu ajutorul proceselor stocastice autosimilare.*

SARCINA LUCRĂRII:

Modelarea traficului observat pe o placă de reţea a unui calculator personal cu ajutorul proceselor stocastice autosimilare;

De creat modelul conceptual şi modelul funcţional-structural al traficului pe o placă de reţea;

Descrierea analitică a principalelor probleme;

Crearea modelului experimental şi vizualizarea rezultatelor.

2.3.1. Modelul Conceptual

Noţiunea de model

Ştiinţific, noţiunea *model* se defineşte ca un sistem care reproduce funcţiile şi proprietăţile obiectului /fenomenului studiat, din realitate (ale originalului) în mod simplificat şi generalizat, cu scopul de a pătrunde mai adânc esenţa acestuia.

Modelarea are o mare valoare euristică colaterală, prin utilizarea ei putându-se dezvolta spiritul de observaţie, capacitatea de analiză, sinteză, creativitatea.

A modela însă înseamnă a construi modele, adică a construi un atare sistem simplificat şi generalizat al obiectului (fenomenului) ce urmează să fie studiat. Acest sistem reproduce anumite particularităţi şi funcţii esenţiale ale originalului.

Clasificarea modelelor

Există câteva modalităţi de clasificare a modelelor. După forma prin care se descrie obiectul/fenomenul studiat ele pot fi clasificate în: modele verbale, grafice şi matematice.

Modelul verbal **este o descriere verbală a fenomenului studiat sau a unor aspecte ale mediului înconjurător**. Exemplu de model verbal: subiectul unei nuvele, scenariul.

Pentru descrierea modelelor verbale se foloseşte limbajul natural care este mai puţin precis, dar mai flexibil şi subtil decât limbajul utilizat pentru calculator sau cel utilizat matematicieni. Astfel de modele se folosesc în cazul când proprietăţile esenţiale ale fenomenului nu pot fi uşor măsurate.

Încercând să construim un model formalizat, putem obţine rezultate neadecvate fenomenului real. Exemplu: fenomene ca patriotismul sau morala nu pot fi măsurate cu ajutorul unor indici numerici. Pentru aceste noţiuni mult mai potrivite sunt modelele verbale.

Modelul grafic **este o descriere grafică a obiectului sau a fenomenului studiat**. Exemplu de model grafic: diagrama, planul în arhitectură şi construcţie, harta (în navigaţie). Modelele grafice sunt flexibile şi uşor de construit. Avantajul lor este că pot fi mai lesne urmărite, mai înţelese decât cele descrise verbal şi pot fi uşor modificate, în caz de necesitate.

Un desen bun valorează cât o mie de cuvinte, deoarece conţine mai multă informaţie decât o descriere verbală pe câteva pagini.

Modelul matematic **reflectă percepţia lumii reale cu ajutorul simbolurilor matematice**.

La aceste modele ne vom referi mai amănunţit în paragraful următor. După natura lor, modelele mai pot fi clasificate în ***modele ideale şi materiale*** (substanţiale).Ca exemple de ***modele ideale*** pot servi: schema unui circuit electric, modelul grafic, logic sau cel matematic, reprezentarea undelor prin curbe sinusoidale.

Modelele materiale sunt concretizări ale modelelor ideale. Ele pot fi realizate atât în forma de existenţă a originalului, cât şi în alte forme. Modelul material nu poate fi construit decât după elaborarea modelului ideal, care exprimă elementele esenţiale ce trebuie concretizate în modelul substanţial (material). Avantajul modelelor materiale faţă de cele ideale e posibilitatea verificării ipotezelor emise prin experimentarea directă asupra lor. Modelele mai pot fi clasificate în modele statice şi dinamice.

Modelul static descrie un moment al stării fenomenului (obiectului) cercetat şi nu ţine seama de modificarea ei în timp.

Modelul dinamic include şi legătura dintre diferitele momente ale stării obiectului (fenomenului), în diferite intervale de timp.

Traficul

Traficul este o noţiune abstractă care identifică obiectul procesărilor dintr-o reţea de telecomunicaţii. De exemplu, pentru o reţea telefonică traficul reprezintă convorbirile, într-o reţea de date, traficul reprezintă datele transferate: fişiere, email-uri, filme, pagini web etc.

Traficul este un fenomen probabilist. Să luam exemplul unei reţele telefonice. Pentru a modela determinist traficul, ar trebui să avem un mijloc prin care să determinăm ce apeluri va efectua fiecare client al reţelei. Sigur, aşa ceva se poate face, de exemplu, dacă este vorba de o reţea privată de telefonie asupra căreia se impun reguli stricte de utilizare. O astfel de reţea ar putea fi folosită, de exemplu, de o companie petrolieră. Această reţea ar avea câte un telefon lângă fiecare puţ petrolier şi unu la centru. Regula strictă de folosire pentru fiecare telefon de la un puţ petrolier ar fi „la ora h se sună centrul de la acest telefon pentru a anunţa cantitatea de petrol extrasă în ultimele 24 de ore; în rest, telefonul nu se foloseşte“. Exemplul poate fi considerat exagerat, dar, totuşi, nu este departe de realitate.

Ei bine, într-o astfel de reţea, traficul ar putea fi privit ca fiind determinist şi, făcând astfel, se pot construi modele mult mai apropiate de realitate decât modelele stocastice. Dar astfel de modele ar avea o arie de aplicabilitate foarte redusă pentru că numărul de reţele în care se pot impune astfel de restricţii dure este extrem de mic. În plus, o astfel de restricţie are şi un efect psihologic neplăcut: „cum? n-am voie să folosesc telefonul atunci când am nevoie?“ Şi iată, am ajuns la o justificare intuitivă a folosirii teoriei probabilităţilor (sau stocastice): dumneavoastră ştiţi, în general, când veţi avea nevoie să daţi un telefon? Ei bine, dacă nici măcar dumneavoastră nu ştiţi, atunci cum ar putea să ştie un model matematic al comportării dumneavoastră? În principiu, nu este imposibil, dar puteţi vedea cât de nepractică ar fi o astfel de abordare. Cum traficul are o evoluţie în timp şi este probabilist modelul matematic, cel mai potrivit este procesul stocastic.

Orice reţea de telecomunicaţii poate fi privită ca o reţea complicată de conducte prin care circula nişte jetoane: unităţile de trafic. Conductele se intersectează uneori, iar acolo unde o fac există mecanisme care dirijează jetoanele ce intră în intersecţie. Terminaţiile conductelor sunt nişte cutiuţe care primesc şi emit jetoane. Unele sunt ca nişte fabrici de jetoane, altele doar mănâncă jetoane, iar altele primesc jetoane, le prelucrează (le rup, le lipesc, le ataşează alte jetoane etc.) şi apoi le retrimit pe o altă conductă.

Reprezentarea aceasta este una abstractă. Jetonul poate fi un octet de date, un pachet de date sau o secundă de convorbire telefonică. Conductele pot fi cabluri telefonice, reţele Ethernet sau reţele ATM. Intersecţiile dintre conducte pot fi comutatoare, rutere sau centrale telefonice.

PLACA DE REŢEA

Dispozitivele de acces la reţea

În cadrul oricărei reţele, în afară de calculatoare (noduri), există cel puţin două componente hardware care sunt obligatorii. Una dintre aceste componente este constituită din placa de interfaţă cu reţeaua, placa de reţea sau adaptor de reţea (plăci Ethernet, ARCnet, Token Ring, sau modem-uri), iar cealaltă, sistemul de cablare.

Interfaţa fizică dintre calculator şi mediul de transmitere este placa de reţea (*Network Interface Card-NIC*). Pentru reţeaua cu cablu, placa se conectează, prin portul ei, cu cablul de reţea, pentru a realiza legătura fizică între calculator şi restul reţelei.

Rolul plăcii de reţea:

a) pregăteşte datele din calculator pentru a fi transmise prin cablul de reţea, formatarea corectă a datelor, astfel încât să fie acceptate de reţea;

b) transmite datele către reţea;

c) controlează fluxul de date intre calculator si cablul de reţea;

d) recepţionează datele sosite prin cablu şi transformarea lor în octeţi, pe care unitatea centrală a calculatorului îi poate înţelege.

Fizic, NIC-ul, este o placă de circuite instalată în calculator într-un slot de intrare/ieşire de pe placa de bază a acestuia, având un port prin care se realizează conectarea în reţea a calculatorului. Fiecare placă de reţea este identificată printr-un cod unic, numit controlul accesului la mediu (Media Access Control - MAC).

Plăcile adaptoare pentru reţea au o mică memorie folosită ca memorie-tampon, numită buffer. Similar altor dispozitive hardware, placa de reţea are nevoie de un driver prin care să poată fi controlată. În sistemele Plug-and-Play (PnP), plăcile de reţea sunt configurate automat, fără intervenţia utilizatorului, în timp ce pe sisteme non-PnP, configurarea se face manual, prin programul de setare a comutatoarelor DIP. Testul de eroare în calitatea semnalului SQE (Signal

Quality Error) este folosit pentru a testa dacă circuitul dintre transmiţător şi interfaţa de reţea (NIC) prezintă coliziuni. În majoritatea reţelelor moderne Ethernet, testul SQE nu mai este folosit. Cele mai multe plăci de reţea (NIC) au un transmiţător integrat, iar testul pentru erori si coliziuni nu mai este necesar.

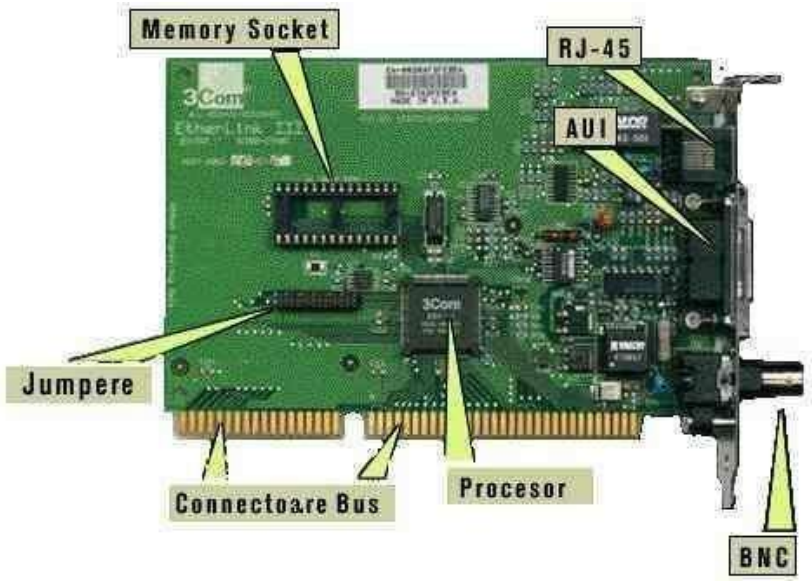


Fig. 25. O placă de reţea

MODELAREA TRAFICULUI

Traficul este un sistem complex. Acest sistem ca şi altele, mai mult sau mai puţin complexe, are nevoie de a fi analizat şi, ulterior, modelat. Există mai mulţi parametri ce caracterizează traficul, de exemplu:

- Viteza maximă de transmisie;
- Capacitatea maximă de transmisie;
- Rata efectivă de transmisie;
- Capacitatea maximă de intrare;
- Stabilitatea traficului;
- Securitatea traficului;
- Altele.

Toţi parametrii enumeraţi mai sus se află într-o inderdependenţă strânsă. În funcţie de caracteristica dorită a traficului, de cerinţele utilizatorilor sistemelor informaţionale şi telecomunicaţionale, se modelează aceşti parametrii pentru a obţine o formulă optimală pentru

fiecare dintre ei sau pentru a mări unu-doi parametri, în detrimentul celorlalţi, pe care nu se pune aşa accent mare.

Deoarece traficul este un fenomen probabilistic, caracterizarea şi modelarea lui, la fel, se fac prin modele, procese şi variabile probabilistice. Aceste procese mai sunt numite aleatorii sau socastice. Astfel de modele clasice sunt Procese Poisson, procese Bernoulli, modele Markov, traficul ca fluid etc. În ultimul timp, mai ales în cazul reţelelor de calculatoare, traficul se modelează prin metoda proceselor stocastice autosimilare, deoarece datele elementele-componente ale traficului se aseamănă una cu alta şi sunt interdependente una faţă de alta.

Procese stocastice autosimilare

Procesele stocastice autosimilare sunt modele matematice utilizate în multe domenii: hidrologie, economie şi finanţe, fizică.

O realizare particulară a unui proces stocastic real este o funcţie reală. Fiecare realizare particulară are asociată o probabilitate de apariţie, care poate fi infinitezimală. Un proces stocastic este o mulţime de realizări particulare ale căror probabilităţi însumate (sau integrate) fac 1. Prin eşantionarea unui proces stocastic („la un moment de timp"), se obţine o variabilă aleatoare. O realizare particulară a unei variabile aleatoare este un număr.

Procesele stocastice autosimilare reprezintă un tip aparte de procese stocastice.

Definiţie: Un proces stocastic {Y (t), t ≥0} se numeşte autosimilar daca pentru orice a > 0 există b > 0 astfel încât

$$Y(at) \doteq bY(t)$$

Notaţia $\doteq$ semnifică egalitatea tuturor distribuţiilor de probabilitate finite. Semnul = între două variabile aleatoare sau între doua procese stocastice înseamnă că pentru orice eveniment realizările particulare ale celor două entităţi sunt egale.

Definitie: Un proces stocastic {Y (t), t ≥0} este continuu stocastic în t dacă pentru orice ε> 0 avem ca

$$\lim_{h \to 0} P\{|Y(t+h) - Y(t)|\} = 0$$

Definitie: Un proces stocastic se numeşte trivial dacă are o singură realizare particulară.

Unicitatea exponentului Hurst. Dacă {Y (t), t ≥0} nu este trivial, este continuu stocastic în t = 0 şi este autosimilar, atunci există şi este unic exponentul H ≥ 0, astfel încât b = a^H. În plus, H > 0 dacă şi numai dacă Y (0) = 0.

S-a observat că traficul cumulat dintr-o reţea de telecomunicaţii poate fi bine modelat de procese autosimilare. În general, se consideră că intensitatea traficului corespunde unui proces stocastic staţionar. Se spune despre un proces stocastic autosimilar că are incremente

staţionare dacă procesul stocastic {Y (h+t)−Y (t)} este staţionar pentru orice valoare a parametrului h.

Pentru modelarea intensităţii traficului se foloseşte procesul stocastic:

$X(n) = Y(n + 1) - Y(n), n \in N$

Parametrul Hurst descrie autosimilaritatea unui proces stocastic. Modelarea traficului cu ajutorul proceselor stocastice autosimilare este motivată de dorinţa de a explica intermitenţa traficului real observată la multe scări de timp. Ca urmare, parametrul Hurst oferă şi o cuantificare a noţiunii de intermitenţă. Până nu de mult, toate studiile de trafic privind modele autosimilare se făceau prin captarea pe o durată mare şi analiză ulterioară. Un avantaj al analizei în timp real (sau on-line) este acela că rezultatul poate fi utilizat în ajustarea comportamentului protocoalelor de telecomunicaţii. Un alt avantaj este că ajută un cercetător să observe rapid ce efect au diverse aplicaţii, topologii de reţea etc. asupra traficului generat în reţea. Un dezavantaj este că estimarea în timp real, fie consumă mult din resursele de calcul, fie este mai inexactă ca urmare a metodei de estimare folosită. O problemă ridicată de estimarea în timp real este adaptarea metodelor clasice de estimare.

Estimarea parametrului Hurst

Parametrul Hurst este prezentat, în general, două etape:

1. Estimarea prin metode clasice a unei funcţii ce descrie procesul stocastic. Exemple de astfel de funcţii sunt autocorelaţia, densitatea spectrală de putere şi varianta ca funcţie de gradul de agregare.
2. Prin alegerea parametrului Hurst se potriveşte forma tipică a funcţiei respective pentru procese autosimilare pe rezultatul estimării anterioare.

Metodele:

- Metoda R/S;
- Analiza varianta-timp;
- Metoda Higuchi;
- Metoda corelogramei;
- Metoda periodogramei;
- Estimatorul Whittle.

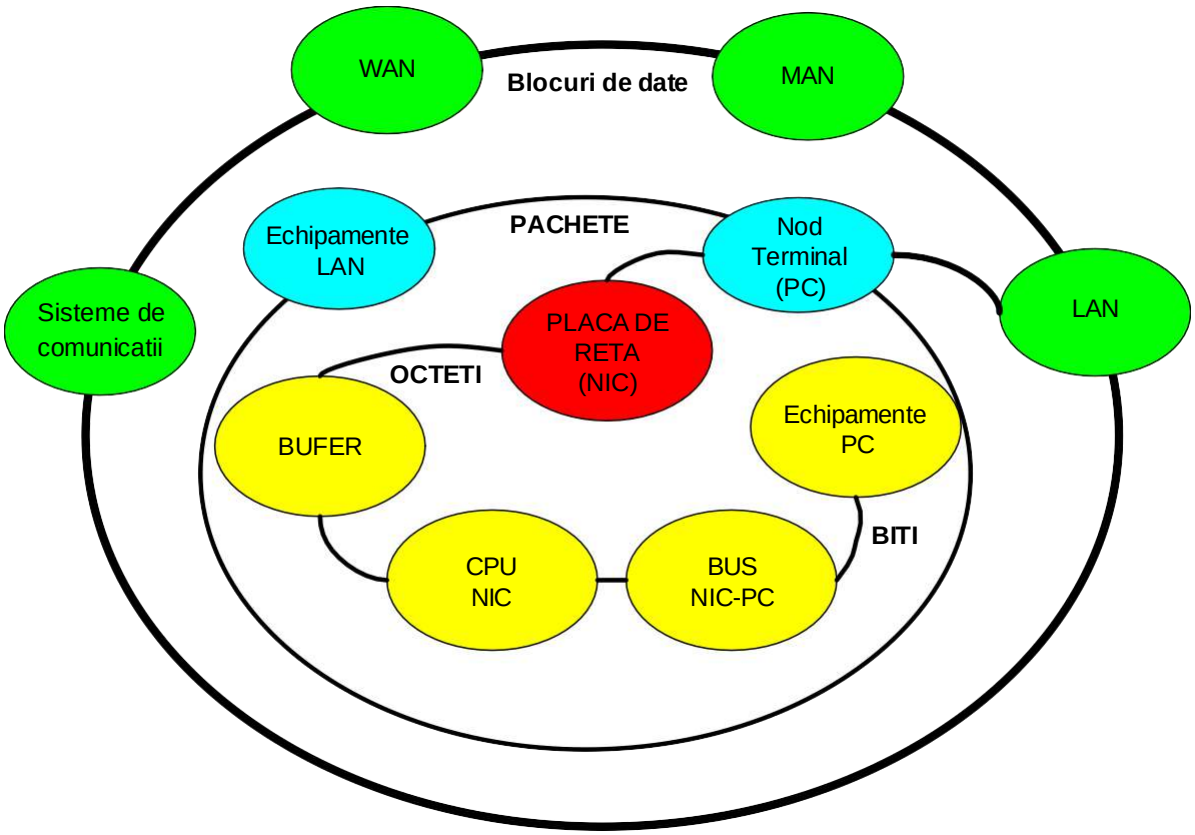


Fig. 26. Modelul conceptual al traficului (în raport cu placa de reţea)

DIAGRAMA MODELULUI CONCEPTUAL AL TRAFICULUI

În figura 26 este reprezentat sistemul complex de circulaţie a ***traficului de date în raport cu placa de reţea a calculatorului***. Pe prima treaptă sunt situate în mod ierarhic macrosistemele de intercomunicaţii prin care circulă traficul de date, începând cu sisteme ca telecomunicaţii, radiocomunicaţii, internet, apoi trecând la reţele foarte mari, terminând cu cele locale (LAN).

Traficul este prezent în fiecare bloc din această treaptă. Pe a doua treaptă se situează componentele principale din care este constituită un LAN – echipamentele de reţea, ce direcţionează, concentrează s.a.m.d. traficul şi nodurile terminale, emiţătorii şi receptorii, dar şi depozitele de informaţii. În centru se află placa de reţea – interfaţa între nivelele macro şi micro, blocul pe care are loc intrarea şi ieşirea cadrelor sau pachetelor datelor din trafic. În placă, datele se stochează în memoria buffer, după care sunt transmise procesorului sau controllerului NIC-ului pentru prelucrare ulterioară. Apoi sunt stocate pe magistrala PC-ului, după care ajung la procesorul central al calculatorului şi alte echipamente, şi invers.

2.3.2. Modelul Structural-Funcţional

SISTEME COMUNICAŢIONALE. REŢELE DE DATE

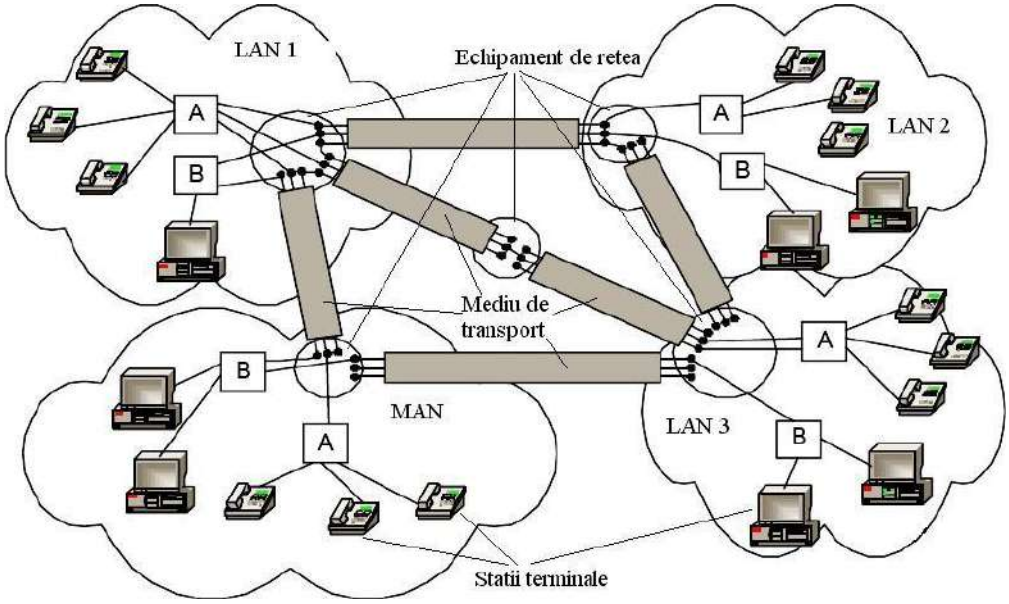


Fig. 27. Un sistem telecomunicaţional compus din diverse reţele de date

Sistemele comunicaţionale sau telecomunicaţionale sunt sistemele ce cuprind un număr mare de noduri emiţătoare şi receptoare de date (semnale informaţionale) si un număr mare de staţii şi echipamente intermediare ce asigură o legătură sigură între nodurile terminale. Ca exemplu de sisteme telecomunicaţionale pot servi reţelele de televiziune, radiotransmisiune, telefonie fixă, telefonie mobilă, reţele de calculatoare etc., iar semnalele informaţionale transmise – imagini video, voce, unde sonore, octeţi de informaţii etc.

FUNCŢIONAREA REŢELELOR DE DATE

Datele transmise prin reţea reprezintă date informaţionale din diverse domenii şi de diferite tipuri. Prin reţea pot circula date ce reprezintă informaţii grafice, video, sunete, alfanumerice, textuală etc. Într-o reţea de calculatoare aceste date se păstrează sub aceeaşi formă – în blocuri de semnale digitale (0 şi 1 logice) – biţi, reunite apoi în octeţi şi formatate pentru a le percepe în forma caracteristică lor. La transmisiunea acestor date prin reţea, ele sunt triate şi formatate în concordanţă cu caracteristicele reţelei: protocoalele de reţea, capacitatea de transmisie, rata de transmisie, viteza etc.

Pornind de la o staţie (calculator) terminală datele (fişiere) sunt fragmentate în blocuri de mărimea capacităţii maxime sau efective de transfer, sunt formatate în concordanţă cu parametrii, ca destinaţia finală, timpul de transmisie, nivelul de securitate etc. Astfel de blocuri se numesc cadre sau pachete de date, ce sunt de un anumit tip. Acest tip este regula şi modul de

transmisiune sau, cum îl mai numim, protocolul. Până a ajunge la staţia finală, cadrele pot trece printr-un şir de staţii intermediare. Aceste puncte intermediare sunt echipamentele de reţea, care pot fi repetoare (dispozitive de corectare şi/sau amplificare şi retransmisiunii) a cadrelor de date, hub-uri, switch-uri (dispozitive de concentrare și/sau comutare a datelor venite si retransmisiunea lor la destinaţia finală) etc.

La venirea la staţia finală, cadrele de date se supun unui proces invers: ele sunt deformatate sau despachetate de informaţiile auxiliare (destinaţia finală, timpul de viaţă etc.); formatate după standardele sistemului informaţional (staţiei finale), stocate şi reunite în fişiere video, audio, text etc.

Astfel, în rezultat, avem transferul de date de la o staţie la altă staţie, aflate la diferite distanţe. Toate etapele intermediare de transformare şi transmisiune a informaţiilor reprezintă traficul de date.

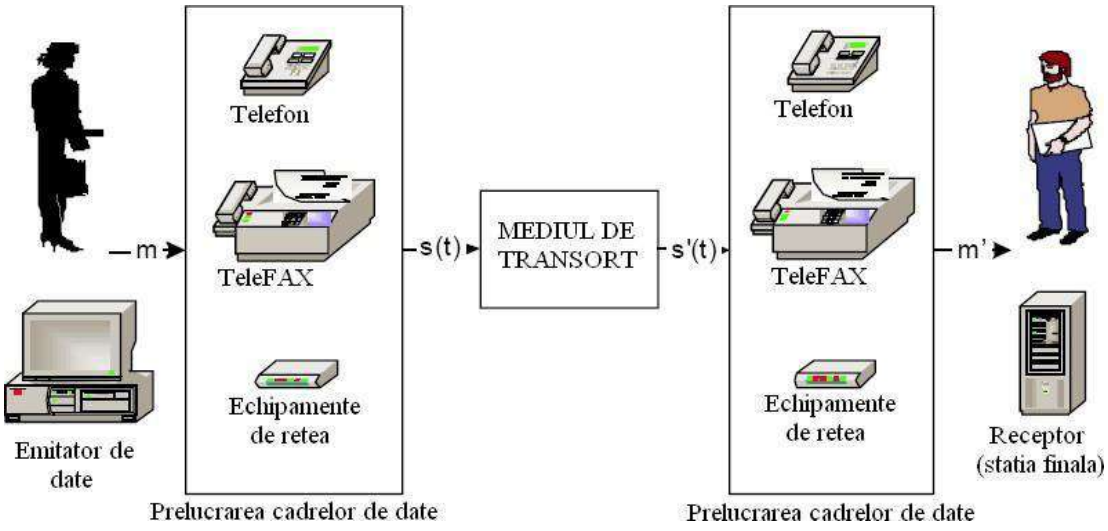


Fig. 28. Model schematic al parcurgerii traficului de date între 2 noduri terminale

Un exemplu mai detaliat este traficul de date pe Placa de reţea:

Pregătirea datelor

Înainte ca datele să fie transmise în reţea, placa de reţea trebuie să le convertească din forma în care ele sunt înţelese de calculator, într-o formă sub care acestea pot circula prin cablul de reţea. Datele circulă în calculator de-a lungul unor circuite, numite magistrale. Acestea constă din mai multe căi alăturate, pe care datele pot circula în paralel, grupate. Pe cablul de reţea, datele trebuie să circule într-un singur şir de biţi (transmisie serială). Placa de reţea preia datele care circulă în paralel, sub formă de grup, şi le restructurează astfel încât să devină un flux serial de biţi ce va fi transportat prin cablul de reţea. Acest lucru se realizează prin transformarea semnalelor digitale din calculator în semnale electrice sau optice care pot

parcurge cablurile de reţea. Componenta care realizează aceasta funcţie este transceiverul. În afară de transformarea datelor, placa de reţea trebuie să îşi notifice poziţia sau adresa către restul reţelei, pentru a putea fi diferenţiată de celelalte plăci din reţea. Adresele de reţea sunt unice, fiind codificate în cipurile plăcii de reţea.

Transmiterea datelor

Înainte ca placa de reţea emiţătoare să transmită date în reţea, ea poartă un dialog electronic cu placa de reţea receptoare, pentru a se pune de acord asupra următorilor parametri:

a) dimensiunea maxima a grupurilor de date ce vor fi transmise;

b) volumul de date transmise fără a se aştepta confirmarea;

c) intervalul de timp dintre blocurile de date;

d) intervalul de timp până la transmiterea confirmării;

e) capacitatea memoriei tampon, pentru a se evita depăşirea acesteia;

f) viteza transmisiei de date.

Fiecare placă semnalează celeilalte propriii parametrii, precum şi acceptarea sau adaptarea la parametrii celeilalte placi. Când toate detaliile comunicării sunt puse la punct, cele două plăci încep sa transmită şi să recepţioneze date.

Putem împărţi o placă de reţea în două mari componente: o parte care se ocupă de traficul pe cablul de reţea şi pe care o vom numi tranceiver şi o parte care asigură interfaţa cu bus-ul calculatorului şi care este interfaţa cu calculatorul.

Tranceiverul primeşte de la interfaţă pachetele de date codate pe care le amplifică şi verifică dacă apar sau nu coliziuni pe cablu în timpul transmisiei, conform CSMA/CD sau CSMA/CA. CSMA este un protocol de transmisie al nivelului legăturii de date implementat aici pentru a asigura o viteză mai mare de rejectare a pachetelor incomplete. Amplificarea semnalului trebuie să fie suficient de puternică, astfel încât, chiar şi în cel mai defavorabil caz, când avem un segment întreg (500m) ocupat (100 de staţii), toate staţiile să primească un semnal suficient de puternic şi, în acelaşi timp, să nu fie atât de puternic, încât staţiile apropiate să sesizeze că a apărut o coliziune (se consideră coliziune când nivelul semnalului în cablu depăşeşte o referinţă care este reglabilă cu componente externe). O altă problemă a tranceiverelor este impedanţa pe care o prezintă conectorului şi care, dacă depăşesc limitele standardului va afecta forma semnalului şi, deci, vor apărea recepţii eronate, putându-se ajunge la deteriorarea întregului trafic pe reţea.

Interfaţa este realizată şi ea, ca şi tranceiverul, pe un singur chip, care are nevoie de un număr foarte mic de componente suplimentare pentru a completa interfaţa. De exemplu, interfaţa firmei AMD 7990 are integrat controller-ul de bus pentru calculatoare IBM PC, codorul-decodorul Manchester, logica şi memoria necesară controlului CRC necesitând

suplimentar un PAL de adaptare, în cazul unei alte CPU şi un amplificator de interfaţă serială, în cazul în care se foloseşte cablu de tranceiver. Acest chip va realiza, deci, codarea şi adăugarea preambulului de sincronizare şi a codului de verificare a CRC la pachetul de date MAC pe care îl primeşte din memorie. La recepţie, va transfera în memorie pachetul pe care îl primeşte fără preambul, dar cu cei 4 octeţi de verificare a CRC. AMD 7990 realizează o testare a CRC pe măsură ce pachetul soseşte, astfel că, la sfârşit, va semnala dacă pachetul are CRC eronat. Această funcţie a nivelului de date este implementată aici pentru a micşora timpul de lucru asupra unui pachet care, oricum, este eronat. Totuşi, renunţarea la pachete nu se face la acest nivel decât în cazul în care pachetul este mai mic de 64 de octeţi (acesta apare doar în cazul unei coliziuni). Restul erorilor sunt raportate astfel încât să se poată renunţa la pachetele incomplete sau să se retransmită cele afectate de coliziune. Aceasta se face şi din cauza faptului că memoria internă a chip-ului este destul de mică, el lucrând prin DMA cu memoria RAM, prin care şi dialoghează cu CPU (rezultatul este că mare parte a pachetelor sunt deja în memorie când se constată o eroare a CRC sau o coliziune). O facilitate a lui AMD 7990 este un reflectometru care poate detecta locul unde este defect cablul de legătură.

Comunicarea între interfaţă şi tranceiver este interactivă, astfel tranceiver-ul ascultă permanent cablul de legătură şi semnalează recepţia şi începe să transmită datele către interfaţă. Aceasta calculează CRC dacă în acest timp tranceiver-ul raportează o coliziune, interfaţa va ignora restul pachetului şi va transmite mai departe eroare de coliziune; altfel, în momentul când s-a umplut buffer-ul, cere accesul la bus şi depune acest bloc în memorie începând cu locaţia care i-a fost comunicată anterior. Daca la sfârşit va constata eroare de CRC, va raporta această eroare.

La transmisie, primeşte pachetul MAC şi începe transmisia, în cazul în care tranceiver-ul nu raportează recepţie cu semnalul de sincronizare continuând cu restul mesajului. Dacă apare o coliziune înainte de a termina transmisia primilor 64 de octeţi, atunci chip-ul va repeta transmisia conform algoritmului nivelului MAC după un timp aleator calculat, conform algoritmului cu sloturi binare exponenţiale trunchiate. Va încerca retransmisia de 15 ori, a 16-a oară va semnaliza eroare de transmisie şi va trece la trimiterea următorului mesaj. Dacă eroarea apare după transmisia primilor 64 octeţi, va semnaliza eroarea, fără a încerca retransmisia.

DIAGRAMA MODELULUI STRUCTURAL-FUNCŢIONAL

Această diagramă trebuie să fie uşor citită şi înţeleasă de personalul uman: inginerii care vor proiecta acest model, tehnicienii care îl vor monta, experţii care îl vor testa şi analiza etc.

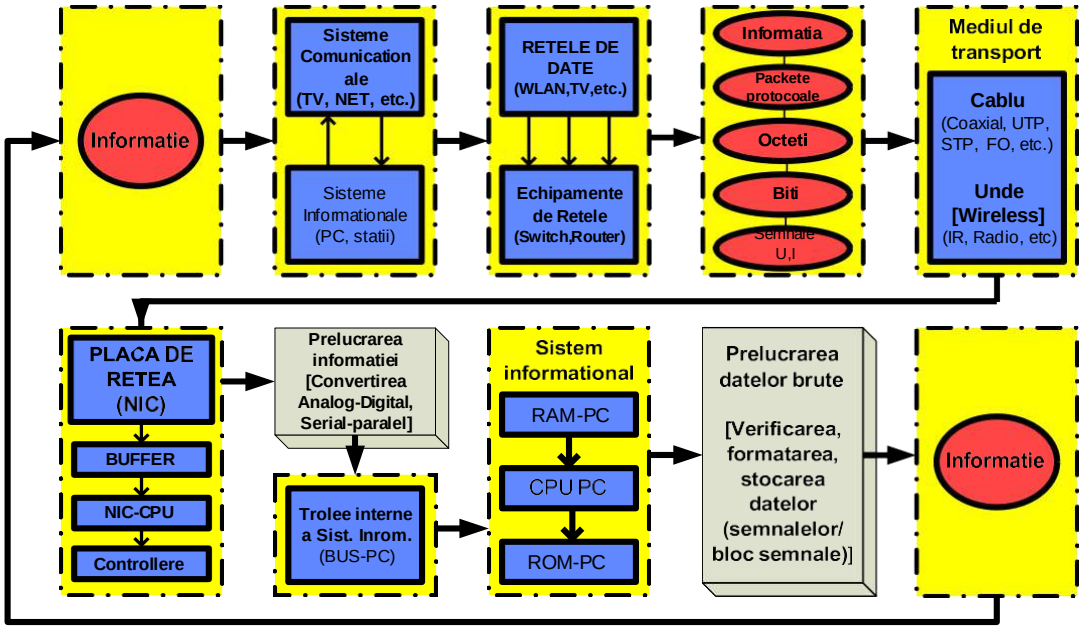


Fig. 29. Modelul structural-funcţional al circulaţiei traficului de date

În modelul structural-funcţional (fig.29.), circulaţia şi transformarea traficului sunt divizate pe blocuri structural-funcţionale. De la informaţia stocată sau culeasă de la diverşi emiţători, traficul sub formă de blocuri informaţionale trece în cele mai complexe sisteme de comunicaţii şi celule principale ale acestora–sistemele informaţionale. De aici, se direcţionează deja în reţele de date şi echipamentele acestora, care o redirecţionează şi o ajustează. Aici, blocurile de informaţii se transformă în pachetele diferitor protocoale de comunicaţii, care, la rândul lor se compun din octeţi, biţi şi, în sfârşit, semnale electrice ca tensiune (U) şi intensitate (I). Aceste semnale trec în mediul de transport, care poate fi cablu sau eter (prin unde). De aici, se ajunge în placa de reţea şi elementele ei (buffer, procesor etc.), unde are loc prelucrarea semnalelor (conversie analog-digitală şi invers, serial-paralel, verificarea erorilor etc.) şi apoi se transpune pe magistrala PC-ului. De aici, semnalele sunt preluate şi procesate de părţile-componente ale PC-ului, unde se verifică integritatea, tipul lor, se formatează şi stochează în memoria fixă, ajungând iar la starea iniţială, dar în altă locaţie.

2.3.3. MODELUL EXPERIMENTAL

Modelul experimental reprezintă simularea modelului conceptual şi structural-funcţional construit cu scopul de a demonstra şi a arăta funcţionalitatea modelelor teoretice anterioare sau a unor anumite aspecte ale lor.

Descrierea modelului experimental

Majoritatea programelor şi aplicaţiilor ce analizează traficul real în reţele de date sunt dotate cu dispozitive sau/şi softuri auxiliare speciale care capturează traficul de la echipamentele de reţea (conectoare, comutatoare, rutere, modem-uri, plăci de reţea etc.). Aceste instrumente auxiliare sunt elaborate de către instituţii specializate şi folosite în scopuri profesionale de analizare şi modelare a traficului.

Din motivul lipsei de astfel de instrumente, în modelul dat, datele traficului vor fi simulate, adică generate cu ajutorul variabilelor aleatoare. În cazul dat, am folosit analogia datelor ce reprezintă traficul cu procesele stocastice autosimilare. Aplicând metoda variantă- timp, am realizat algoritmul ce efectuează modelarea datelor de trafic, după următorul principiu: *parametrul Hurst* (H), se determină în funcţie de caracteristicile (încărcarea mediului) traficului – raportul datelor de intrare la datele de ieşire, în cazul dat a pachetelor pe placa de reţea a calculatorului. În funcţie de acest parametru H se modelează caracteristica ulterioară a traficului.

Fig. 30.a. Raportul traficului pe o placă reală, t1

În acest program-exemplu, s-a luat drept capacitate maximă de trecere în dispozitiv date cu mărimea (*size*) de 10 u.c.m. (unităţi convenţionale de măsură), datele generate – pachetele au mărimea cuprinsă între 8÷12 u.c.m. şi sunt fragmentate în caz că mărimea lor depăşeşte

capacitatea maximă de trecere. Pentru aceasta, se folosesc porţiuni a buffer-ului de memorie (buf cu mărimea de 100 u.c.m.) subbufer-ele (*fi_buf* şi *fe_buf*).

Fig. 30 b. Raportul traficului pe o placă reală t2

Analizând traficul real pe o placă de reţea la un calculator, observăm că numărul pachetelor de intrare este mai mare decât cel de ieşire (fig.30 a,b) pentru timpuri diferite (*t1*,*t2*), din motivul că computerul în cauză comunică cu mai multe dispozitive intermediare (switch- uri, hub-uri etc.) şi noduri terminale (alte computere, servere, printere etc.) şi primeşte mai multe date decât emite în exterior. De aceea, am luat iniţial puterea de emisie a plăcii în corelaţie cu parametrul Hurst. Iniţial, H=3 (la 3 pachete primite din exterior, placa emite un pachet) şi, în continuare, în funcţie de încărcarea traficului, H variază.

Programul (*fişierul TRAFIC.CPP*) ce simulează şi modelează traficul de date pe placa de reţea a calculatorului este compus din următoarele blocuri principale:

- Declararea şi iniţializarea elementelor de trafic: pachete, porturi, buffere, contoare şi alţi parametri auxiliari;
- Generarea pachetelor interne (*i_pack*) şi externe (*e_pack*);
- Analizarea pachetelor (determinarea tipului, necesităţii fragmentarii, contorizării etc.);
- Determinarea încărcării reţelei (usage) şi a parametrului Hurst (H);
- Afişarea datelor în timp real şi a totalizărilor la consolă în fişierul *TRAFIC.LOG*.
- Codul-sursa și rezultatele sunt prezentate în anexă.

Descrierea mediului de lucru

Programul dat a fost efectuat în limbajul de programare Turbo C++ 3.0.

Mediul a fost ales ţinând cont de faptul ca Turbo C++ conţine bibliotecile şi instrucţiunile de bază, ce permit simplu crearea de algoritmi de simulare şi modelare a datelor. În Turbo C++ este uşor şi accesibil de a crea structuri (de exemplu, structura packet folosită de noi în program), variabile aleatore (generarea caracteristicilor datelor).

Aceşti algoritmi pot fi uşor rulaţi, atât în mediul MS-DOS, cat şi în mediul Windows OS şi alte medii. Rularea şi executarea codului scris, practic nu depinde de mediul SoftWare în care funcţionează, nu are cerinţe mari faţă de caracteristicile mediului HardWare. La fel mediul Turbo C++ oferă posibilitate ca aceşti algoritmi să fie modificaţi uşor pentru adaptare la alte cerinţe, condiţii şi uşor pot fi modificaţi pentru a crea algoritmi similari în mediile ce se bazează pe C++, cum ar fi C++ Builder, C++ Visio, C#, Java, PHP. La fel unele porţiuni de cod, ce realizează operaţii concrete, pot fi uşor moştenite de alte medii şi aplicaţii, având aceeaşi bază sintactică şi semantică. Toate aceste avantaje oferă o portabilitate înaltă codului- sursă a programelor scrise în Turbo C++.

CONCLUZII

Modelarea traficului cu ajutorul proceselor stocastice autosimilare este promiţătoare. Până acum, s-au făcut studii, mai ales în ceea ce priveşte dimensionarea memoriilor tampon ale elementelor de reţea. Se speră ca noile modele sa se aplice în zone conexe, cum ar fi controlul congestiei. Un prim-pas necesar în acest sens este realizarea în timp real şi prin software a estimării parametrului Hurst.

Anexa 1

Listingul programului:

```
#include <conio.h>
#include <stdlib.h>
#include  <stdio.h>
#include <string.h>
#include <dos.h>

#define IP  101 //
#define TCP  102 //
#define UDP  103 //
#define ICMP 104 //
#define FTP  105 // definirea porturilor pentru tipurile de packete
#define DNS  106 //
#define HTTP 107 //
#define POP3 108 //

int main(void)
{

clrscr();
printf("\nPROGRAM:\n");
printf("\nMODELAREA TRAFICULUI PE PLACA DE REŢEA. METODA PROCESELOR
STOCASTICE AUTOSIMILARE.\n");
printf("\n\n\n\n\n\n");
printf("\nPentru rularea programului secvential, apasati orce tasta\n");
printf("\nPentru finisare şi vizualizarea totalizarilor, tastati <Esc>\n");
printf("\nPentru vizualizarea rezultatelor şi totalizarilor deschideti fisierul TRAFIC.LOG din
directoriul activ\n");
getch();

  struct packet //declarare structura "packet"
  {
    int size;
    int port;
```

```
  char *type;
}
  e_pack, //packet extern
  i_pack;   //packet intern

FILE *f;

int t=0,i=0;
int s_pack=0, r_pack=0;
int buf=0,fi_buf=0,fe_buf=0; //declarare, initializare buffer şi subbufere
int fi_pack,fe_pack; //fragmente de packete
int contorIP=0;
int contorTCP=0;
int contorUDP=0;
int contorICMP=0;
int contorFTP=0;
int contorDNS=0;
int contorHTTP=0;
int contorPOP3=0;
int contorAlteProtocoale=0;
int H=3; //Parametrul Hurst
float usage;

clrscr();
randomize();

f=fopen("TRAFIC.LOG","w+");

while (i!=27)
{
fi_buf=0; //initializare subbuffer pentru fragmentare packete interne

i_pack.size=8+random(5);i_pack.port=101+random(10);//generarea packeteor interne
s_pack++;
```

```
if (i_pack.size>10) {fi_buf=i_pack.size-10; fi_pack=10;}
   else fi_pack=i_pack.size; // fragmentare packete

switch (i_pack.port) // determinarea,numararea tipului packetelor
{
  case IP   : contorIP++;  strcpy(i_pack.type,"IP");   break;
  case TCP  : contorTCP++; strcpy(i_pack.type,"TCP");  break;
  case UDP  : contorUDP++; strcpy(i_pack.type,"UDP");  break;
  case ICMP : contorICMP++;strcpy(i_pack.type,"ICMP"); break;
  case FTP  : contorFTP++; strcpy(i_pack.type,"FTP");  break;
  case DNS  : contorDNS++; strcpy(i_pack.type,"DNS");  break;
  case HTTP : contorHTTP++;strcpy(i_pack.type,"HTTP"); break;
  case POP3 : contorPOP3++;strcpy(i_pack.type,"POP3"); break;
  default   : {contorAlteProtocoale++;strcpy(i_pack.type,"Alt tip");}

}
printf("Parametrul Hurst, H=%d\n",H);
printf("Intern_packet: size=%d port=%d tip=%s\n",i_pack.size,i_pack.port,i_pack.type);
printf("               fi_pack=%d fi_buf=%d\n",fi_pack,fi_buf);
fprintf(f,"Parametrul Hurst, H=%d\n",H);
fprintf(f,"Intern_packet: size=%d port=%d tip=%s\n",i_pack.size,i_pack.port,i_pack.type);
fprintf(f,"               fi_pack=%d fi_buf=%d\n",fi_pack,fi_buf);

usage=0; // initializarea estimatorului incarcarii mediului
for (t=0;t<H;t++)
{
 fe_buf=0; //initializare subbuffer pentru fragmentare packete externe
 e_pack.size=8+random(5);e_pack.port=101+random(10);//generare packete externe
 r_pack++;
 usage+=e_pack.size;
 if (e_pack.size>10) {fe_buf=e_pack.size-10; fe_pack=10;}
    else fe_pack=e_pack.size;

 switch (e_pack.port)
{
```

```
 case IP   : contorIP++;  strcpy(e_pack.type,"IP");   break;
 case TCP  : contorTCP++; strcpy(e_pack.type,"TCP");  break;
 case UDP  : contorUDP++; strcpy(e_pack.type,"UDP");  break;
 case ICMP : contorICMP++;strcpy(e_pack.type,"ICMP"); break;
 case FTP  : contorFTP++; strcpy(e_pack.type,"FTP");  break;
 case DNS  : contorDNS++; strcpy(e_pack.type,"DNS");  break;
 case HTTP : contorHTTP++;strcpy(e_pack.type,"HTTP"); break;
 case POP3 : contorPOP3++;strcpy(e_pack.type,"POP3"); break;
 default   : {contorAlteProtocoale++;strcpy(e_pack.type,"Alt tip");}

}

printf("                                BUFFER=%d\n",buf);
fprintf(f,"                                BUFFER=%d\n",buf);
printf("Extern_packet: size=%d port=%d tip=%s\n",e_pack.size,e_pack.port,e_pack.type);
printf("              fe_pack=%d fe_buf=%d\n",fe_pack,fe_buf);
fprintf(f,"Extern_packet: size=%d port=%d tip=%s\n",e_pack.size,e_pack.port,e_pack.type);
fprintf(f,"              fe_pack=%d fe_buf=%d\n",fe_pack,fe_buf);

 if (buf+fe_pack>=100) //analizare, prelucrare buffer
   {
      printf("\nBUFFER PLIN\n");
      fprintf(f,"\nBUFFER PLIN\n");
      buf=0; //descarcare buffer
      printf("DESCARCARE BUFFER\n\n");
      fprintf(f,"DESCARCARE BUFFER\n\n");
   }
 buf+=fe_pack;
 buf+=fe_buf;
 //i++;
 }

 usage=usage/H;
 //estimarea parametrului H (Hurst)
 if (usage>9.5 && usage<10.5) H=3;//conditia normal
```

```
    else if (usage>=8 && usage<=9.5) H=2;//conditia incarcare mica
          else if (usage>=10.5 && usage<=12) H=4;//conditia incarcare mare

  //delay(1000);
   i=getch();
  }

  //Totalizarea rezultatelor
  //getch();
  printf("\nPackets Sent - %d      Received - %d\n\n",s_pack,r_pack);
  fprintf(f,"\nPackets Sent - %d      Received - %d\n\n",s_pack,r_pack);
  printf("Packete IP    - %d\n",contorIP);
  fprintf(f,"Packete IP    - %d\n",contorIP);
  printf("Packete TCP  - %d\n",contorTCP);
  fprintf(f,"Packete TCP   - %d\n",contorTCP);
  printf("Packete UDP  - %d\n",contorUDP);
  fprintf(f,"Packete UDP  - %d\n",contorUDP);
  printf("Packete ICMP  - %d\n",contorICMP);
  fprintf(f,"Packete ICMP  - %d\n",contorICMP);
  printf("Packete FTP  - %d\n",contorFTP);
  fprintf(f,"Packete FTP  - %d\n",contorFTP);
  printf("Packete DNS  - %d\n",contorDNS);
  fprintf(f,"Packete DNS  - %d\n",contorDNS);
  printf("Packete HTTP  - %d\n",contorHTTP);
  fprintf(f,"Packete HTTP  - %d\n",contorHTTP);
  printf("Packete POP3  - %d\n",contorPOP3);
  fprintf(f,"Packete POP3  - %d\n",contorPOP3);
  printf("Alte protocoale-%d\n",contorAlteProtocoale);
  fprintf(f,"Alte protocoale-%d\n",contorAlteProtocoale);
  getch();
  fclose(f);
  return 0;

}
```

Anexa 2

Rezultatele obţinute:

Exemplu de vizualizare a rezultatelor la consolă:

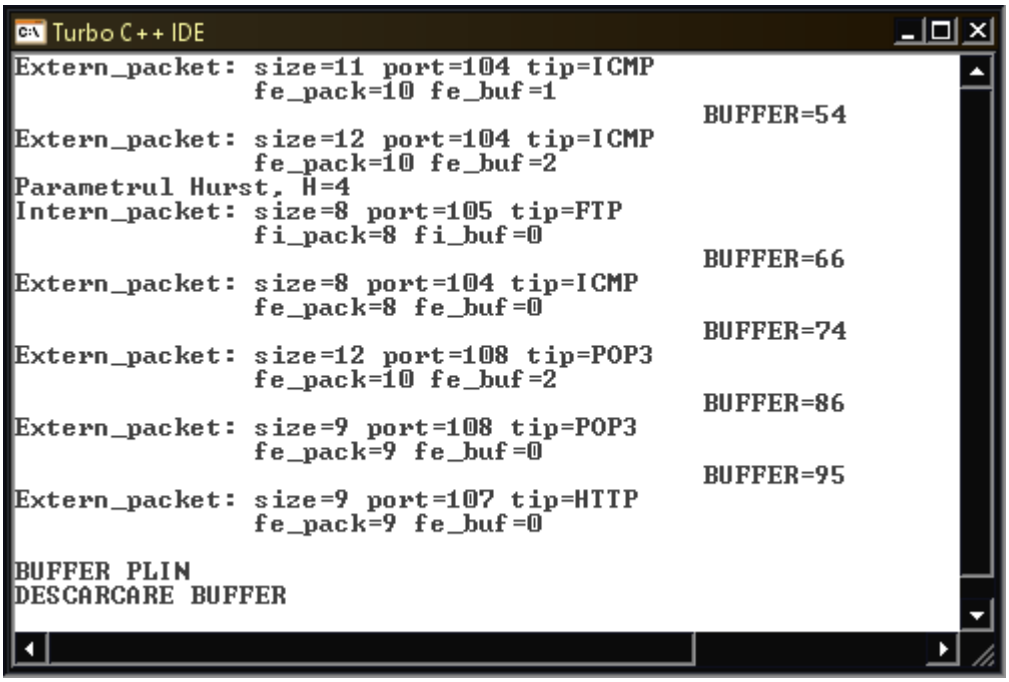


Fig. 31 a. Captarea imaginii de la consolă – traficul la un timp t1

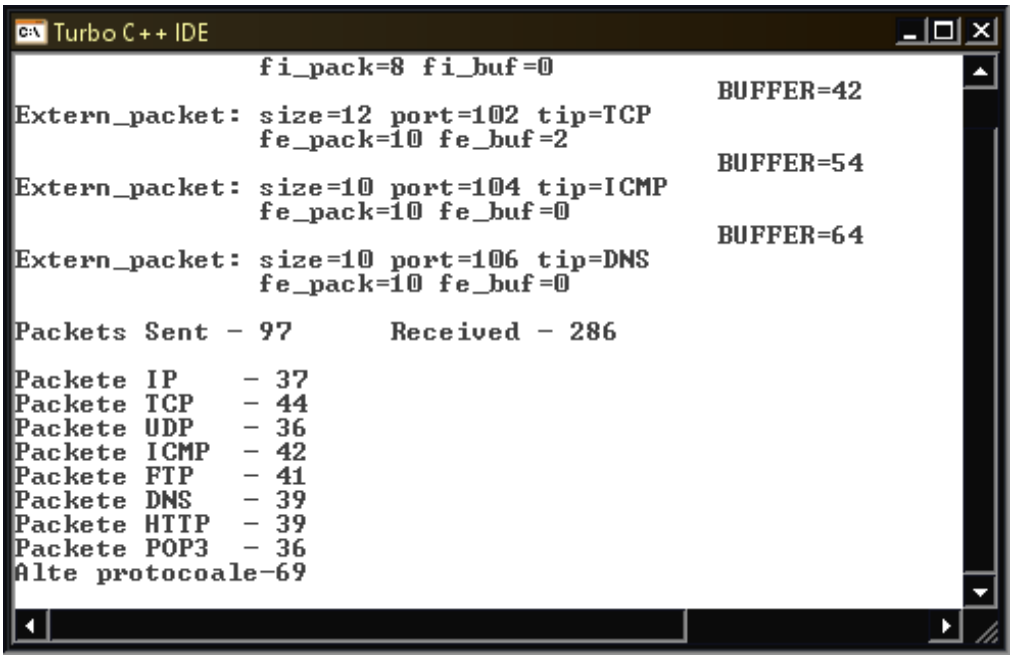


Fig. 31 b. Captarea imaginii de la consolă – traficul la alt timp t2

Fişierul de raport TRAFIC.LOG

Parametrul Hurst, H=3

Intern_packet: size=10 port=104 tip=ICMP

fi_pack=10 fi_buf=0

BUFFER=0

Extern_packet: size=11 port=105 tip=FTP

fe_pack=10 fe_buf=1

BUFFER=11

Extern_packet: size=8 port=102 tip=TCP

fe_pack=8 fe_buf=0

BUFFER=19

Extern_packet: size=9 port=101 tip=IP

fe_pack=9 fe_buf=0

Parametrul Hurst, H=2

Intern_packet: size=8 port=102 tip=TCP

fi_pack=8 fi_buf=0

BUFFER=28

Extern_packet: size=12 port=106 tip=DNS

fe_pack=10 fe_buf=2

BUFFER=40

Extern_packet: size=10 port=104 tip=ICMP

fe_pack=10 fe_buf=0

Parametrul Hurst, H=4

Intern_packet: size=11 port=107 tip=HTTP

fi_pack=10 fi_buf=1

BUFFER=50

Extern_packet: size=10 port=104 tip=ICMP

fe_pack=10 fe_buf=0

BUFFER=60

Extern_packet: size=12 port=103 tip=UDP

fe_pack=10 fe_buf=2

BUFFER=72

Extern_packet: size=9 port=109 tip=Alt tip

fe_pack=9 fe_buf=0

BUFFER=81

Extern_packet: size=8 port=106 tip=DNS

fe_pack=8 fe_buf=0

Parametrul Hurst, H=3

Intern_packet: size=9 port=103 tip=UDP

fi_pack=9 fi_buf=0

BUFFER=89

Extern_packet: size=11 port=110 tip=Alt tip

fe_pack=10 fe_buf=1

BUFFER=100

Extern_packet: size=11 port=103 tip=UDP

fe_pack=10 fe_buf=1

BUFFER PLIN

DESCARCARE BUFFER

… //raportul integral este prezentat pe suportul fizic

BUFFER=8

Extern_packet: size=12 port=103 tip=UDP

fe_pack=10 fe_buf=2

Parametrul Hurst, H=3

Intern_packet: size=9 port=102 tip=TCP

fi_pack=9 fi_buf=0

BUFFER=20

Extern_packet: size=12 port=108 tip=POP3

fe_pack=10 fe_buf=2

BUFFER=32

Extern_packet: size=11 port=103 tip=UDP

fe_pack=10 fe_buf=1

BUFFER=43

Extern_packet: size=8 port=104 tip=ICMP

fe_pack=8 fe_buf=0

Parametrul Hurst, H=3

Intern_packet: size=12 port=104 tip=ICMP

fi_pack=10 fi_buf=2

BUFFER=51

Extern_packet: size=12 port=109 tip=Alt tip

fe_pack=10 fe_buf=2

BUFFER=63

Extern_packet: size=9 port=110 tip=Alt tip

fe_pack=9 fe_buf=0

BUFFER=72

Extern_packet: size=10 port=101 tip=IP

fe_pack=10 fe_buf=0

Parametrul Hurst, H=3

Intern_packet: size=10 port=102 tip=TCP

fi_pack=10 fi_buf=0

BUFFER=82

Extern_packet: size=8 port=105 tip=FTP

fe_pack=8 fe_buf=0

BUFFER=90

Extern_packet: size=8 port=105 tip=FTP

fe_pack=8 fe_buf=0

BUFFER=98

Extern_packet: size=10 port=105 tip=FTP

fe_pack=10 fe_buf=0

BUFFER PLIN

DESCARCARE BUFFER

Parametrul Hurst, H=2

Intern_packet: size=10 port=108 tip=POP3

fi_pack=10 fi_buf=0

BUFFER=10

Extern_packet: size=12 port=106 tip=DNS

fe_pack=10 fe_buf=2

BUFFER=22

Extern_packet: size=10 port=106 tip=DNS

fe_pack=10 fe_buf=0

Parametrul Hurst, H=4

Intern_packet: size=8 port=105 tip=FTP

fi_pack=8 fi_buf=0

BUFFER=32

Extern_packet: size=10 port=106 tip=DNS

fe_pack=10 fe_buf=0

BUFFER=42

Extern_packet: size=10 port=108 tip=POP3

fe_pack=10 fe_buf=0

BUFFER=52

Extern_packet: size=12 port=106 tip=DNS

fe_pack=10 fe_buf=2

BUFFER=64

Extern_packet: size=10 port=109 tip=Alt tip

fe_pack=10 fe_buf=0

Parametrul Hurst, H=4

Intern_packet: size=12 port=106 tip=DNS

fi_pack=10 fi_buf=2

BUFFER=74

Extern_packet: size=8 port=104 tip=ICMP

fe_pack=8 fe_buf=0

BUFFER=82

Extern_packet: size=10 port=108 tip=POP3

fe_pack=10 fe_buf=0

BUFFER=92

Extern_packet: size=8 port=108 tip=POP3

fe_pack=8 fe_buf=0

BUFFER PLIN

DESCARCARE BUFFER

Parametrul Hurst, H=3

Intern_packet: size=8 port=101 tip=IP

fi_pack=8 fi_buf=0

BUFFER=11

Extern_packet: size=10 port=106 tip=DNS

fe_pack=10 fe_buf=0

BUFFER=21

Extern_packet: size=11 port=104 tip=ICMP

fe_pack=10 fe_buf=1

BUFFER=32

Extern_packet: size=10 port=102 tip=TCP

fe_pack=10 fe_buf=0

Parametrul Hurst, H=3

Intern_packet: size=8 port=110 tip=Alt tip

fi_pack=8 fi_buf=0

BUFFER=42

Extern_packet: size=12 port=102 tip=TCP

fe_pack=10 fe_buf=2

BUFFER=54

Extern_packet: size=10 port=104 tip=ICMP

fe_pack=10 fe_buf=0

BUFFER=64

Extern_packet: size=10 port=106 tip=DNS

fe_pack=10 fe_buf=0

Packets Sent - 97 Received - 286

Packete IP - 37

Packete TCP - 44

Packete UDP - 36

Packete ICMP - 42

Packete FTP - 41

Packete DNS - 39

Packete HTTP - 39

Packete POP3 - 36

Alte protocoale-69

3.2. APLICAŢII

3.2.1. Soft Educaţional „Simulare Reţea“

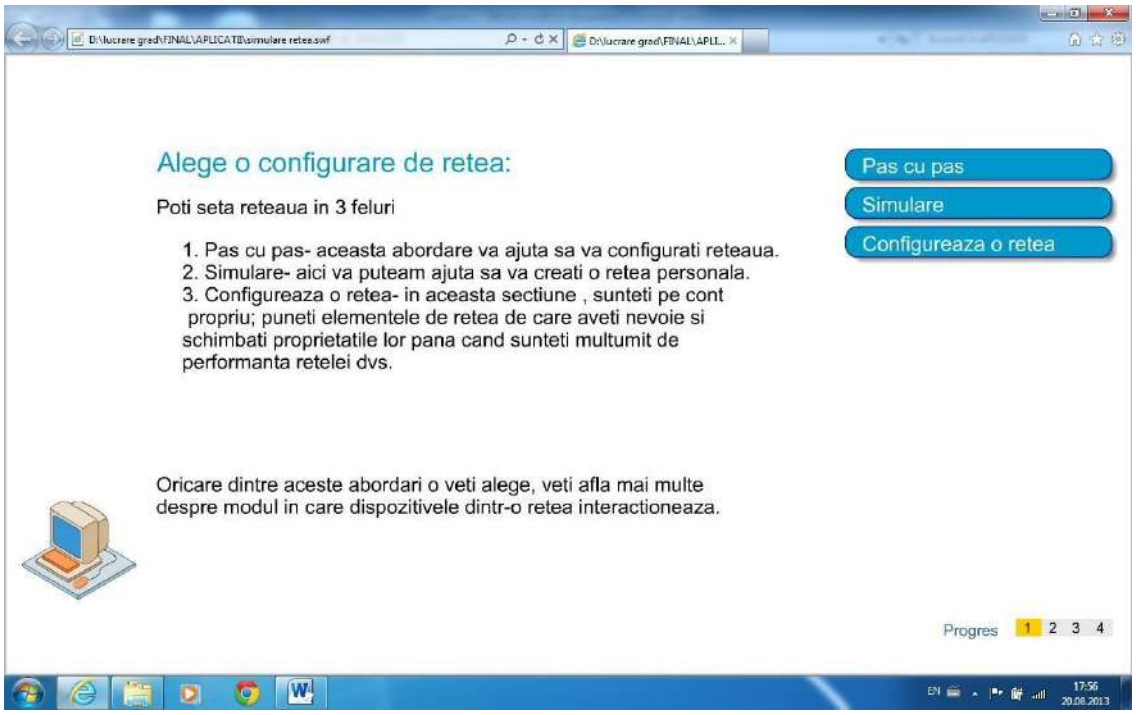
Textul „Software educaţional” sugerează orice produs digital în orice format ce poate fi utilizat pe orice calculator şi reprezintă un subiect, o temă, o problemă, un experiment, o lecţie, un curs etc., fiind o alternativă sau un ajutor faţă de metodele educaţionale tradiţionale (tabla, creta etc.).

La origine, sintagma „soft educaţional” denumea produsul care prin proiectare concretiza, prin parcurgerea softului, care cuprindea teme de lucru îşi regla demersul pe baza feedbackului continuu, iar interacţiunea elevului cu softul producea învăţarea. Acest atribut îi dădea dreptul să se numească soft educaţional.

Proiectarea unui soft educaţional presupune mai multe etape ce diferă prin caracterul activităţii grupurilor de specialişti implicaţi în acest proces. Prima etapă o reprezintă proiectarea pedagogică (design educaţional), moment în care se defineşte şi se concretizează o anumită strategie educaţională. În cea de a doua etapă, cea de realizare informatică/grafică/interfaţă, această strategie este transpusă într-un program de instruire, având toate caracteristicile funcţionale solicitate prin proiectul pedagogic.

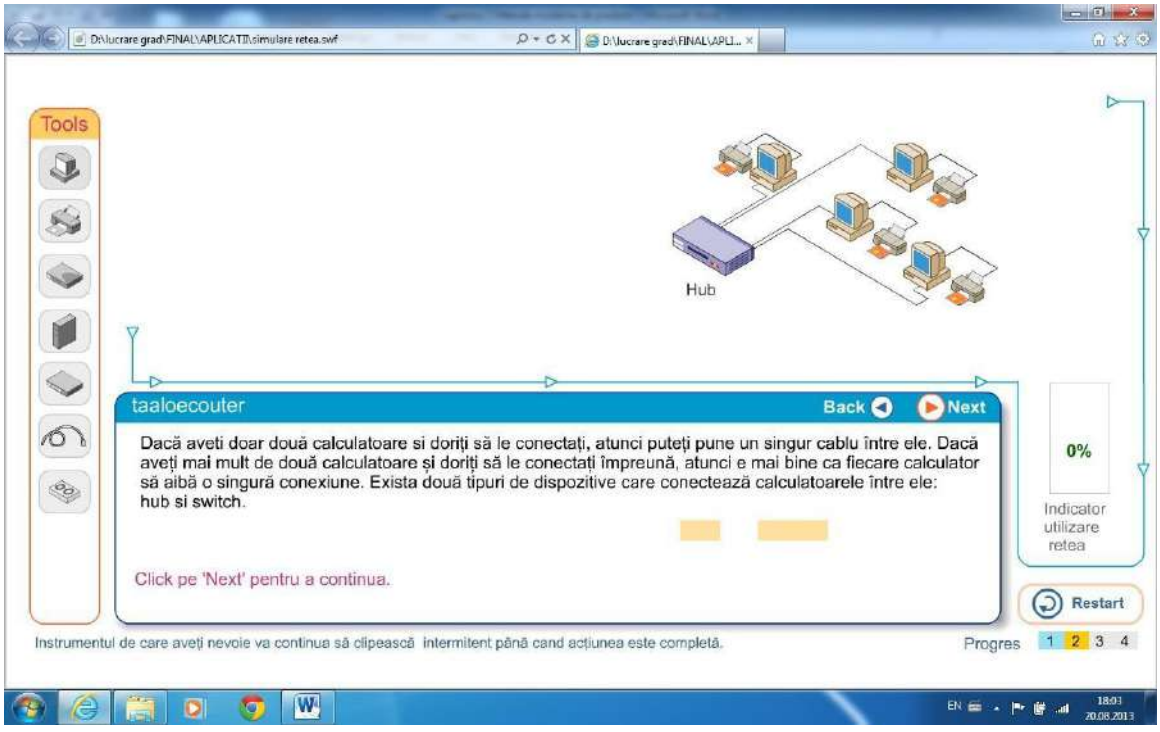
Aplicaţiile de simulare permit reprezentarea controlată a unui fenomen, proces sau sistem real, prin intermediul unui model cu comportament analog, oferind astfel posibilitatea modificării unor parametrii şi observării comportamentului sistemului. Caracteristica principală a aplicaţiilor de acest tip este capacitatea utilizatorului să observe sau să modeleze un fenomen sau acţiune fără o implicare reală în acestea. La această categorie se afiliază şi jocurile educaţionale.

Am creat în Flash un soft educaţional care poate simula traficul într-o reţea de calculatoare. Elevii pot alege să configureze o reţea în 3 feluri, după cum arată imaginea următoare.

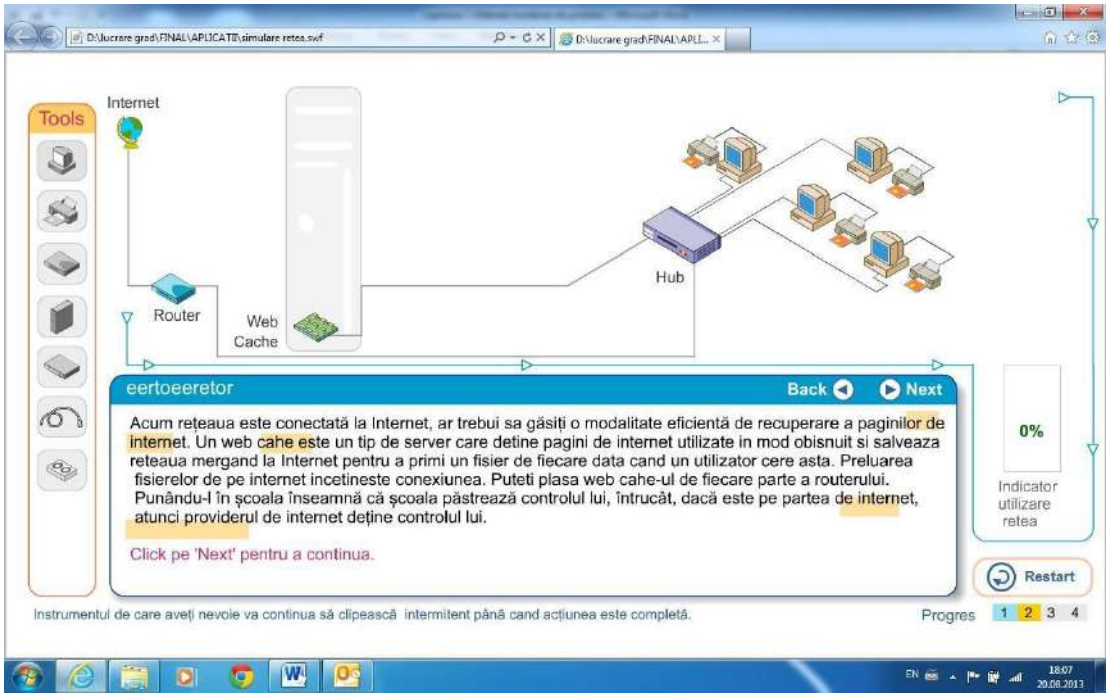


Dacă aleg modalitatea **„Pas cu pas**“, vor fi îndrumaţi pe parcursul creării unei reţele, de la cea mai simplă reţea la o întreagă suită de reţele.

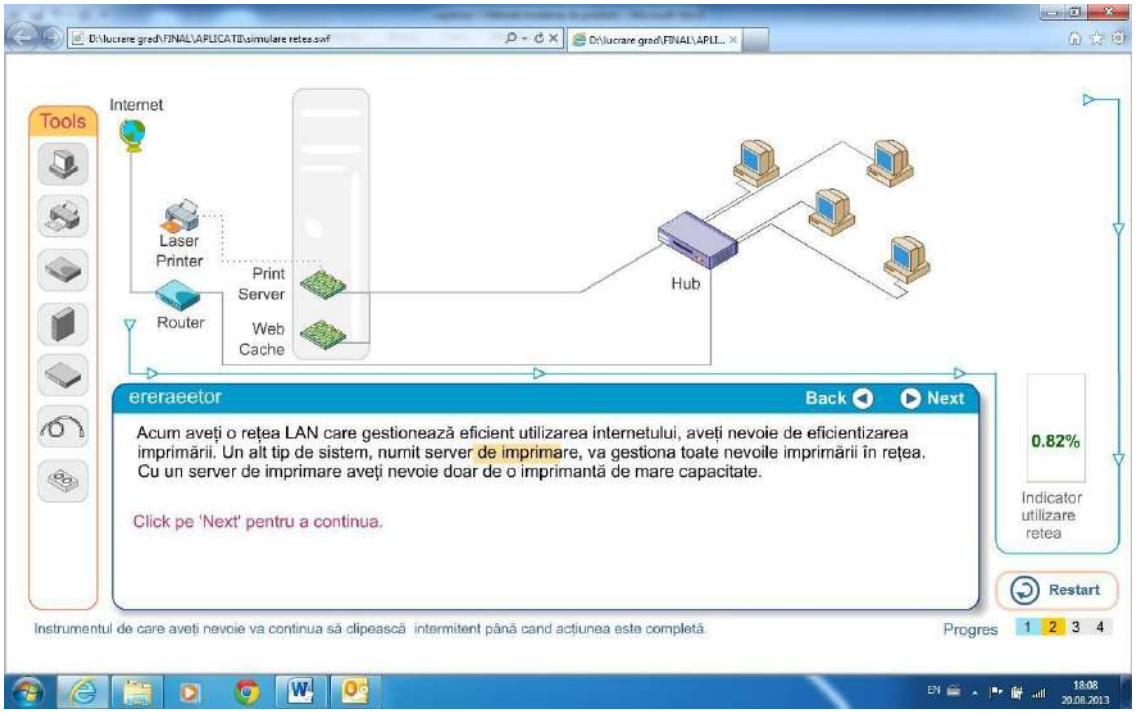
Vor fi motivate toate îndrumările (de ce aleg un switch, de ce un hub, de ce un server de fişiere, un server de mail sau multimedia).

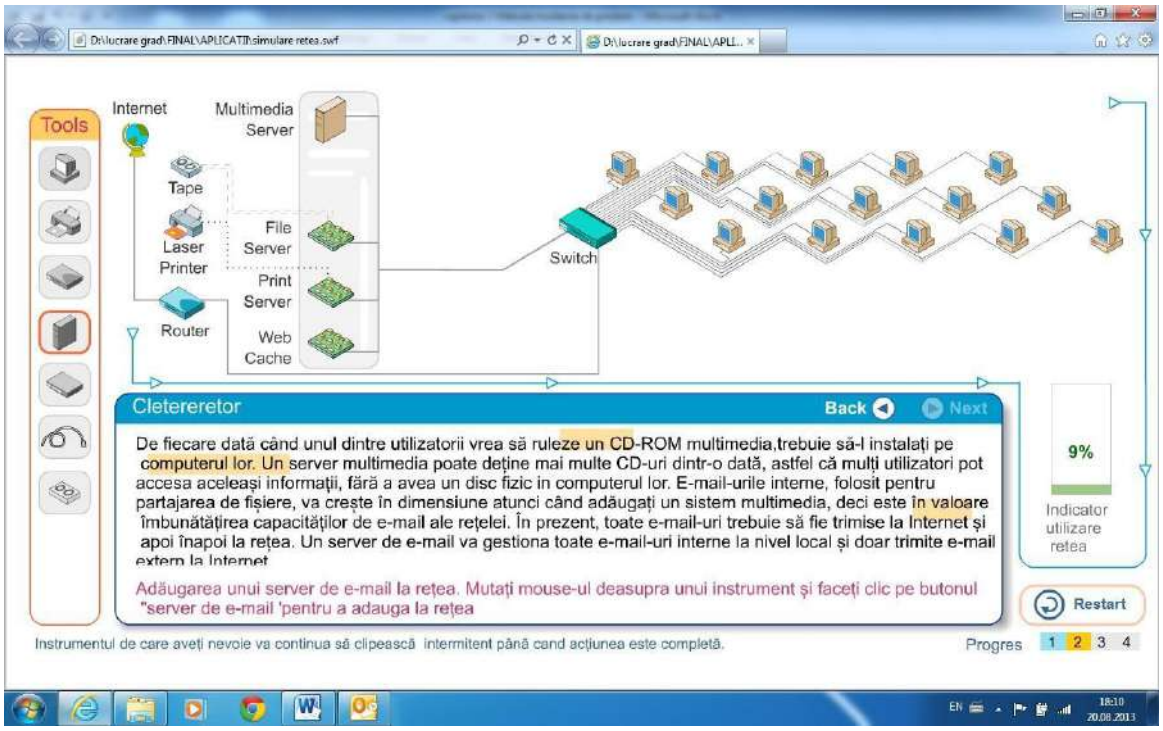


Noţiunile teoretice sunt prezentate în caseta de text din josul ecranului, după cum arată imaginea de mai jos:

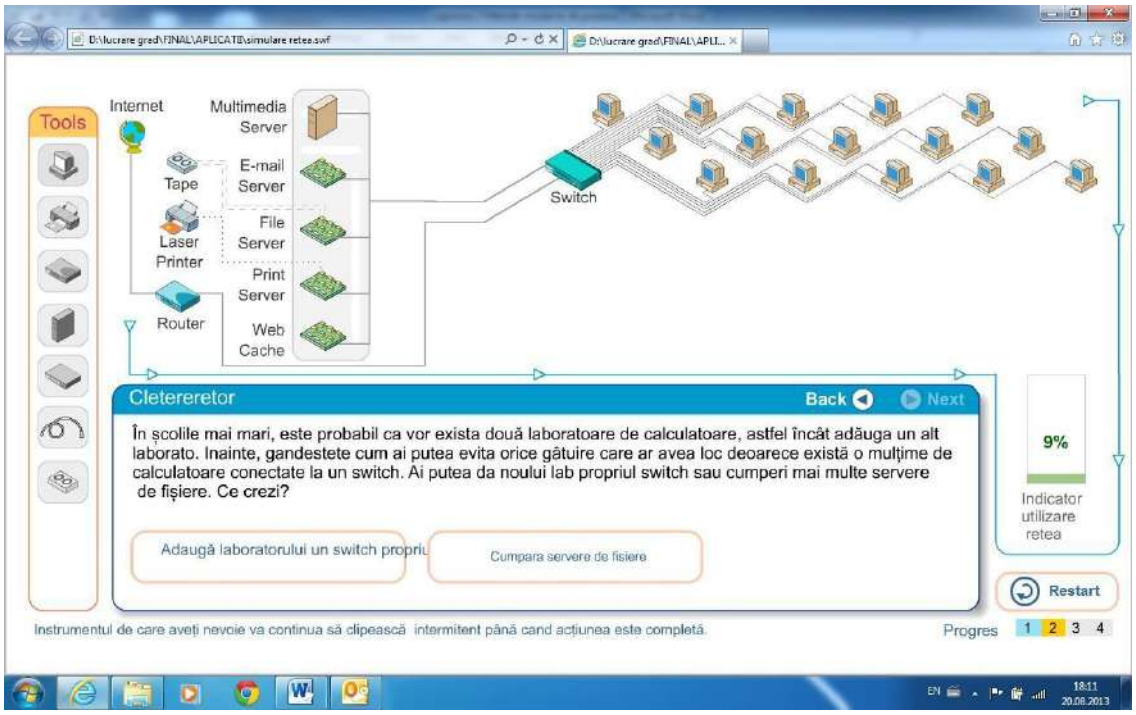


Se porneşte de la reţea simplă de 4 calculatoare şi se va ajunge la 2 laboratoare cu câte 15 calculatoare.

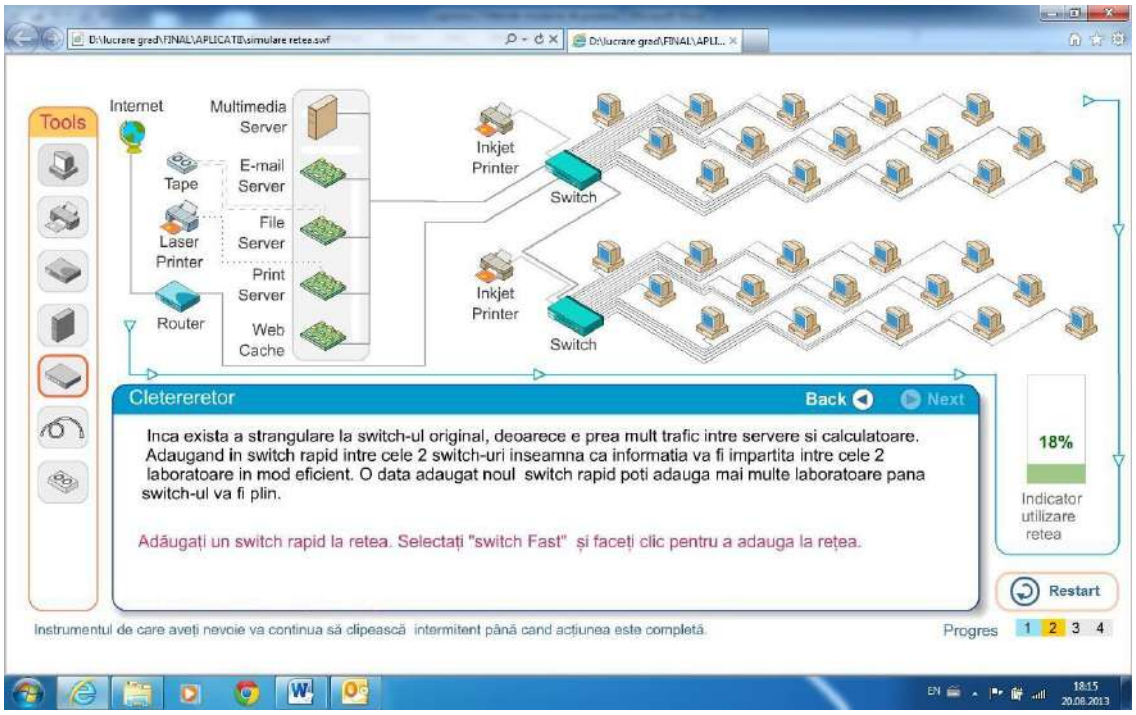




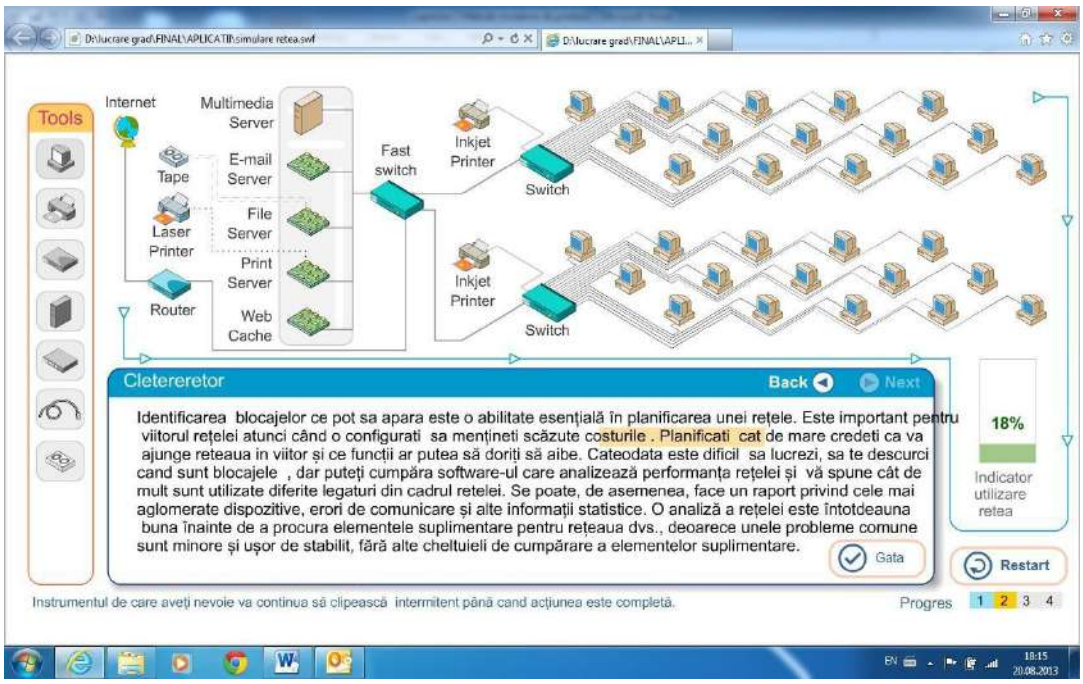
Există pe parcurs şi interacţiunea directă cu utilizatorii, punându-se întrebări despre cum cred ei că ar fi mai bine în implementarea reţelei: achiziţionarea unui switch sau a unui server de fişiere?



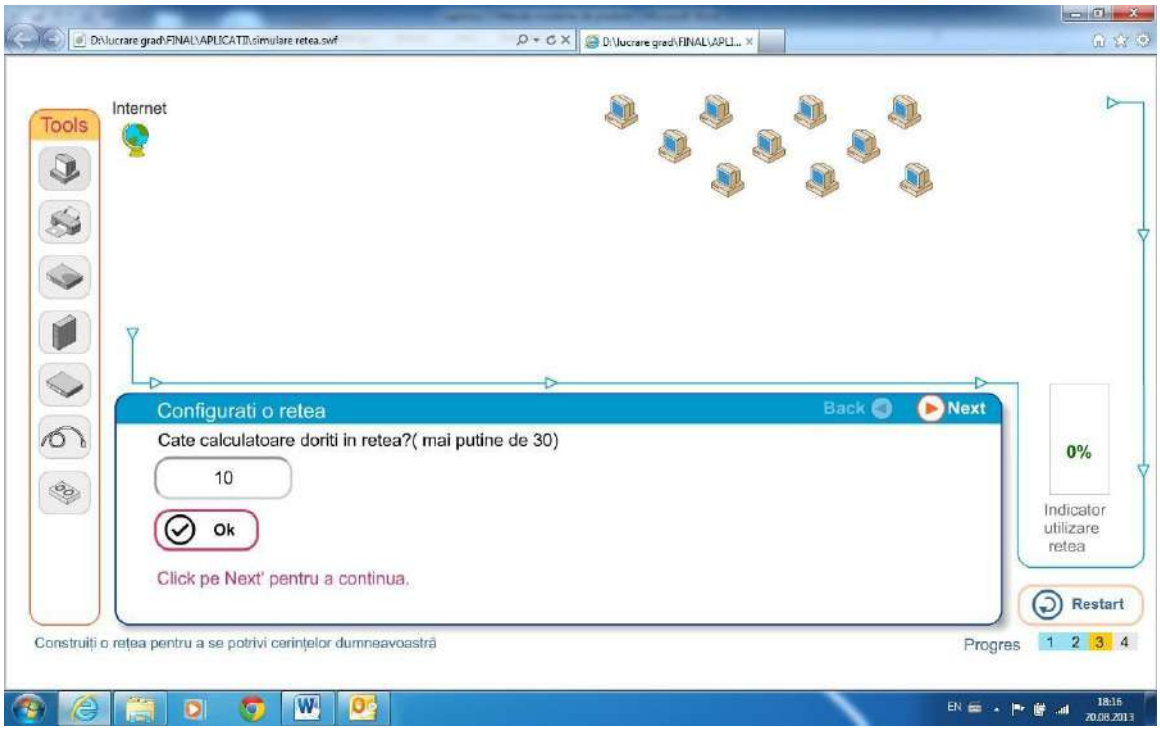
Se dezvoltă reţeaua şi se reprezintă grafic neajunsurile ei:



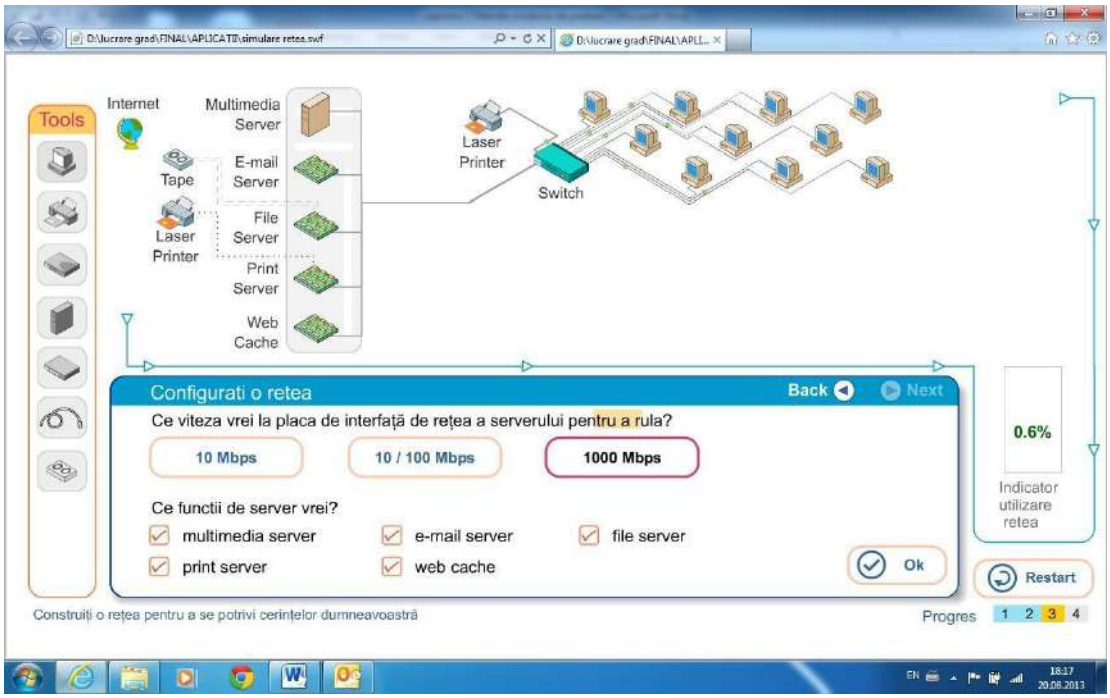
Indicaţiile finale legate de planificarea reţelei sunt date la sfârşit.

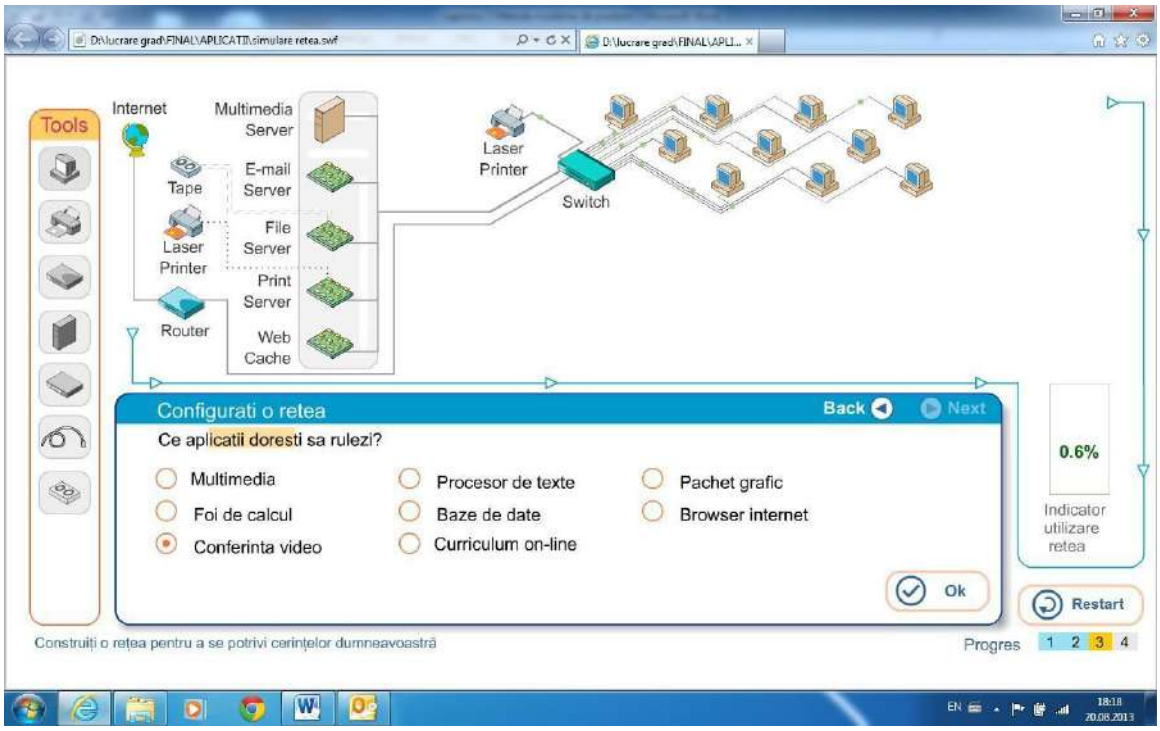


Simularea presupune construirea unei reţele în mod interactiv, utilizatorul putând alege numărul componentelor din reţea, tipul acestora şi proprietăţile tehnice ale dispozitivelor.

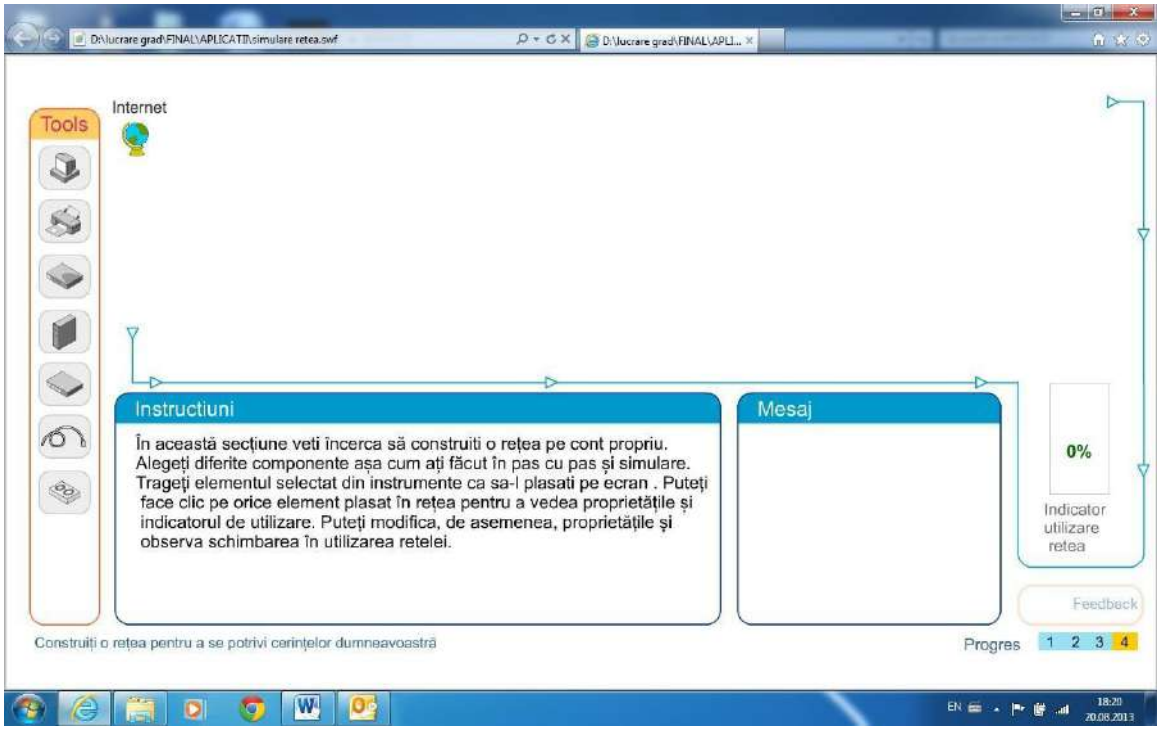


În funcţie de necesităţile reţelei pe care doreşte să o construiască, dar şi de performanţa reţelei, se aleg componentele hard şi soft:





Configurează o reţea reprezintă testul final al softului, deoarece aici utilizatorul este pe cont propriu, făcând alegeri între componentele puse la dispoziţie şi abia la sfârşit se poate vedea nivelul utilizării reţelei.



BIBLIOGRAFIE

1. *Administrarea reţelelor de calculatoare şi a laboratoarelor informatice SEI-Extinderea competenţelor IT în învăţământul preuniversitar - utilizarea eficientă a laboratoarelor informatizate* POSDRU/87/1.3/S/62260
2. Lupşa, R.L. *Reţele de calculatoare,* Casa Cărţii de Ştiinţă, 2008
3. Munteanu, A., Greavu, V. *Reţele locale de calculatoare*, Editura Polirom, Iaşi, 2006
4. Tanenbaum, A. *Reţele de calculatoare,* Editura Byblos, Bucureşti, 2003
5. Tanenbaum, A., *Organizarea structurată a Calculatoarelor*, Editura Byblos, Bucureşti, 2004
6. Tanasă, Ş.C., *Reţele de calculatoare,* Universitatea „Al. I. Cuza", Iaşi, 2006
7. Timofte, C., Constantinescu, R., Ilie-Nemedi, I. *Reţele de calculatoare*. Caiet de seminar.
8. Păstrăvanu, O., *Sisteme de evenimente discrete. Tehnici calitative bazate pe formalismul reţelelor*, Bucureşti, 1997.
9. Radu, G. *Modele de trafic pentru reţele de date,* Bucureşti, 2003
10. http://www.regielive.ro
11. http://biblioteca-digitala.ase.ro/biblioteca/ 20.05.2009
12. http://cisco.netacad.net/cnams/dispatch 01.05.2009
13. http://stud.usv.ro/~imacoveiciuc/
14. http://www.neculau.go.ro/atestat/html/placa_de_retea.html
15. http://www.iso.org