Universitatea Politehnică București

Facultatea de Electronică, Telecomunicații și Tehnologia Informației

TCP (Transmission Control Protocol)

Temă Rețele de Calculatoare

Studenți: MAVRU Anca

IONESCU Liviu

Grupa: 442A

Cuprins:

1. **Introducere** (Mavru Anca)

Necesitatea oamenilor de pretutindeni de a comunica a dus la dezvoltarea și optimizarea continuă a rețelelor de comunicații. În lumea modernă, se pot transmite date oricând si oriunde, lucru posibil datorită unor tehnologii ce permit rețelelor să fie eficiente și sigure. Rețelele de calculatoare și alte dispozitive fac posibilă legătura ultra rapidă între oricare părți ale globului. Totuși, tehnologia comunicării prin intermediul rețelelor de calculatoare nu este atât de veche, dar a cunoscut o rată de creștere și dezvoltare exponențială, ceea ce face ca zilele în care transmiterea datelor nu era așa ușoară să fie de mult uitate.

Principiul comunicării, indiferent de tipul ei, este în general unul destul de simplu: un mesaj (informație, date, pachet) este transmis de către expeditor (sursa mesajului) prin intermediul unui canal, care reprezintă mediul ce asigură calea prin care mesajul transmis ajunge la destinatar (receptor), cel care interpretează informația primită. Prin urmare, există trei elemente principale care fac posibilă comunicarea: expeditorul, canalul și receptorul.

Cu toate acestea, pentru a realiza o comunicare eficientă și mai ales corectă, fie între oameni, fie într-o rețea, este necesară stabilirea unor reguli numite protocoale. În cazul unei rețele, succesul comunicării este asigurat de interacțiunea mai multor protocoale diferite. Un grup de protocoale interdependente care sunt necesare pentru a îndeplini o funcție de comunicare se numește o suită de protocoale. Aceste protocale sunt puse în aplicare atât software cât și hardware și sunt încărcate pe fiecare gazda și dispozitiv de rețea.

"Întrepătrunderea dintre domeniul calculatoarelor și cel al comunicațiilor a avut o influență profundă asupra modului în care sunt organizate sistemele de calcul. Vechiul model al unui singur calculator care servește problemelor de calcul ale organizației a fost înlocuit de un model în care munca este făcută de un număr mare de calculatoare separate, dar interconectate. Aceste sisteme se numesc **rețele de calculatoare**." [1]

[1] Andrew Tanenbaum – "Retele de calculatoare" editia III, Ed. Agora 1998

2. Concepte generale

2.1. **Modelul OSI și TCP/IP** (Ionescu Liviu)

Modelul de referință OSI

Modelul OSI este un concept modular, ierarhic și cuprinde 7 straturi (Fizic, Legătură de date, Rețea, Transport, Sesiune, Prezentare, Aplicație). Principiul de bază dupa care funcționează este următorul: fiecare strat oferă servicii stratului superior, bazându-se pe serviciile primate de la stratul inferior. Interfața care definește serviciile este de tipul punct de acces la serviciu (SAP) iar un nivel al unui nod comunică cu nivelul corespunzător altui nod prin protocoale.

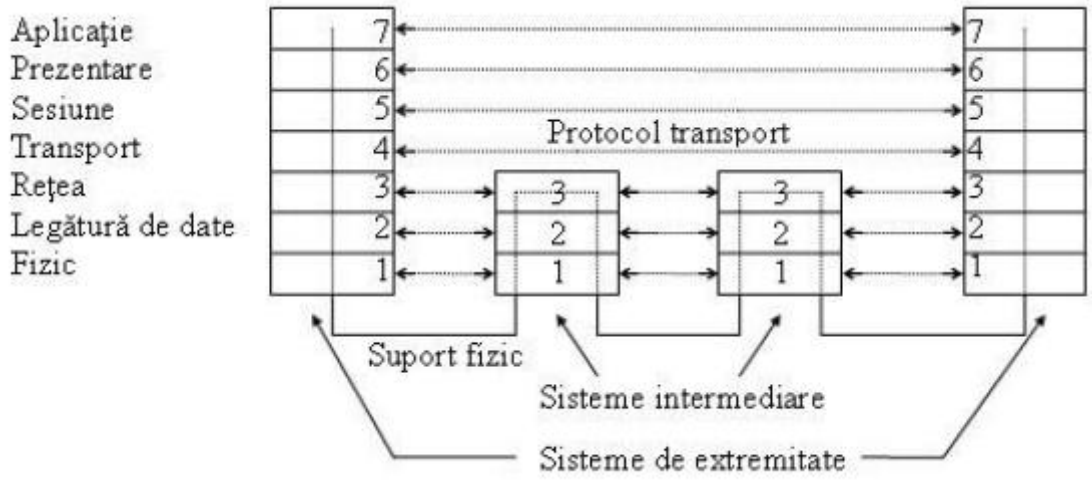

Figura 1: Modelul de referință OSI [2]

[2] Curs Arhitecturi de Rețea și Internet, an III, CTI

Propunerea pentru acest model a fost dezvoltată de Organizația Internațională de Standardizare și a reprezentat un prim pas în procesul de standardizare internațională a protocoalelor folosite pe diferite niveluri. Modelul se ocupă de conectarea sistemelor deschise, de unde și numele de **OSI- Open Systems Interconnection**.

Pentru determinarea celor 7 niveluri ale modelului de referință sunt utilizate următoarele principii: sunt create nivele separate pentru funcțiuni care diferă prin prelucrarea efectuată sau prin tehnologia utilizată, funcțiunile similare se regrupează în același nivel, este posibilă efectuarea de modificări ale funcțiilor si protocoalelor fără a afecta alte nivele.

Modelul OSI precizează numai ce ar trebui să facă fiecare nivel, de aceea nu reprezintă în sine o arhitectură de rețea. Mai mult decât atât, OSI nici nu specifică serviciile și protocoalele utilizate la fiecare nivel. Standardele produse de ISO pentru fiecare nivel nu fac parte din modelul de referință propriu-zis, fiecare standard fiind publicat ca un standard internațional separat.

Modelul de referință TCP/IP

Acest model a apărut ca soluție la neajunsurile modelului ARPANET, la care adăugarea de rețele prin satelit și radio și încercarea de a le interconecta cu protocoalele existente a dus la apariția unor probleme. Astfel a devenit prioritară proiectarea unui model de referință care să permită interconectarea mai multor tipuri diferite de rețele.

Numele modelului TCP/IP este dat de cele două protocoale fundamentale utilizate. Din Figura 3 se observă ca modelul de referință TCP/IP are decât patru nivele: Aplicație, Transport, Internet, Gazdă-la-Rețea.

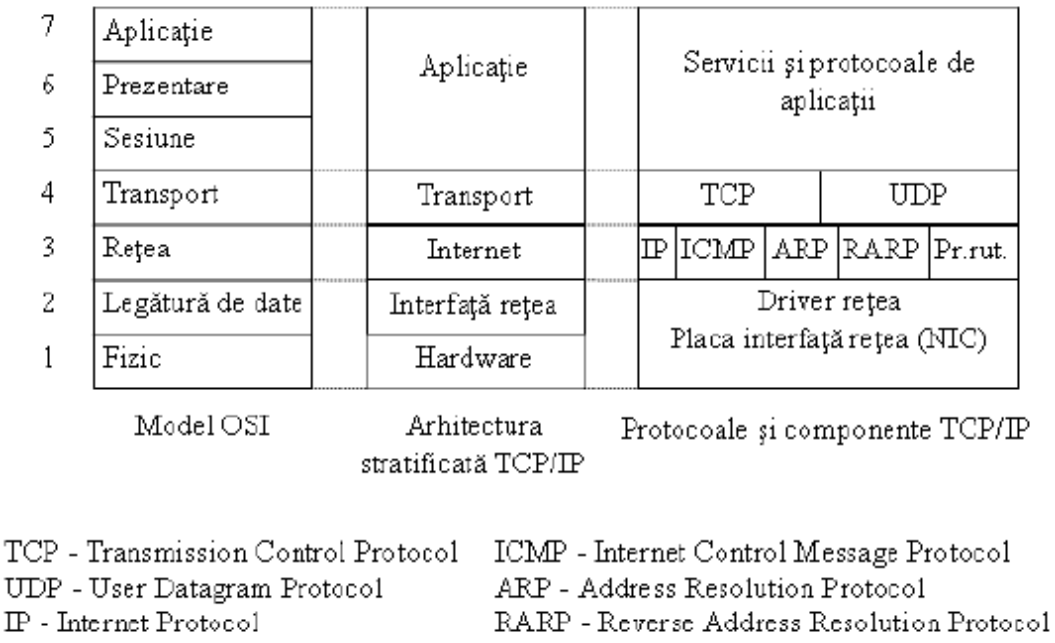

Figura 2: Arhitectura TCP/IP[3]

Application	Telnet, FTP, e-mail, etc.
Transport	TCP, UDP
Network	IP, ICMP, IGMP
Link	device driver and interface card

Figura 3: Cele 4 nivele ale modelului TCP/IP[4]

---

[3] Curs Arhitecturi de Rețea și Internet, an III, CTI

[4] TCP/IP Illustrated, Volume 1-The Protocols- W. Richard Stevens

2.2. Nivelul TRANSPORT în Internet (Ionescu Liviu)

Așa cum se poate observa din Figura 4 nivelul transport este nivelul 4 in ambele modele de referință.

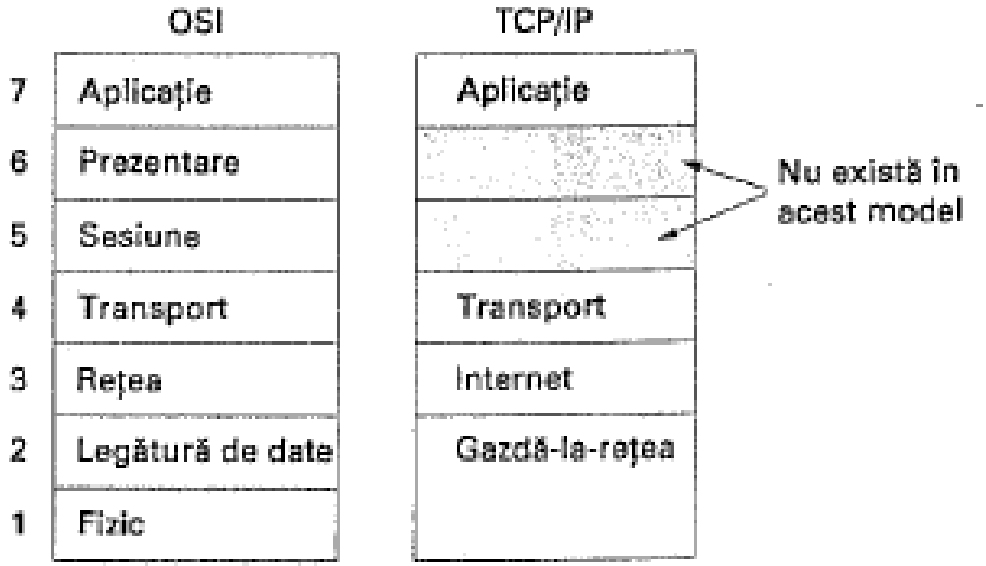

Figura 4: Comparație între arhitecturile modelelor OSI și TCP/IP[5]

În rețelele de comunicații, nivelul transport sau nivelul 4 asigură serviciile de comunicație punct-la-punct pentru aplicații într-o arhitectură pe nivele de componente de rețea şi protocoale. Nivelul transport asigură servicii convenabile cum ar fi suportul pentru stream-uri de date orientate pe conexiune, încredere, controlul fluxului şi muliplexarea. Nivelele de transport sunt conținute atât în modelul TCP/IP, care este fundamentul Internetului, şi modelul OSI (Open Systems Interconnection) de rețelistică generală.

[5] Rețele de Calculatoare- Andrew Tanenbaum, ediția III, Ed. Agora 1998

Nivelul transport este centrul arhitecturii stratificate de rețea. Rolul său este de a oferi servicii de comunicație aplicațiilor ce rulează pe diferite calculatoare. Nivelul transport asigură conexiunea logică dintre calculatorul sursă şi calculatorul destinație, gestiunea conexiunilor (dacă este cazul), fluxul de date, refacerea după întreruperi a comunicațiilor şi corecția erorilor.

Nivelul transport separă nivelele arhitecturale în două categorii: nivelele 1-4, alcătuind furnizorul serviciului de transport și nivelele 5 - 7, alcătuind utilizatorul acestui serviciu. Primele au funcții orientate spre comunicații, iar celelalte spre organizarea dialogului și a transformărilor sintaxei unităților de date comunicate între utilizatori.

Funcții ale nivelului TRANSPORT

- ✓ Un protocol de nivel transport permite comunicația logică (deşi cele două calculatoare nu sunt conectate fizic direct, din punct de vedere al aplicațiilor par a fi conectate fizic direct) între aplicații care rulează pe calculatoare diferite.

- ✓ Pe partea de transmisie, un protocol de nivel transport converteşte mesajele primite de la un proces aplicație transmițător în PDU-uri de nivel 4 (transport). Aceasta se face prin fragmentarea mesajului şi prin adăugarea câte unui antet de nivel transport fiecărui fragment. Apoi fiecare PDU de nivel 4 este transmis la nivelul rețea, unde este încapsulat într-un PDU de nivel 3.

- ✓ Pe partea de recepție, protocolul de nivel transport recepționează PDU-urile de nivel 4, de la nivelul rețea, înlătură antetul de transport din aceste PDU-uri, reasamblează mesajele şi le transmite unui process aplicație de recepție.

- ✓ Orice rețea de calculatoare poate dispune de mai multe protocoale de nivel transport. De exemplu rețeaua INTERNET foloseşte protocoalele TCP şi UDP. Fiecare dintre aceste protocoale oferă seturi diferite de servicii de nivel transport pentru o anumită aplicație.

2.3. Comparației între OSI și TCP/IP din punct de vedere al nivelului Transport

(Mavru Anca)

La o primă vedere modele de referință OSI și TCP/IP au multe lucruri în comun. Deoarece ambele modele de bazează pe conceptul unei stive de protocoale independente, din punct de vedere al funcționalității nivelurile comune sunt asemănătoare.

Cel mai important și evident punct comun este că nivelurile până la nivelul **transport** inclusiv (adică fizic, legătură de date, rețea, transport) sunt necesare pentru a pune la dispoziția proceselor care doresc să comunice un serviciu de transport capăt-la-capăt independent de rețea. Din acest motiv, nivelurile mai sus menționate formează furnizorul de transport. „Nivelurile de deasupra transportului sunt beneficiari orientați pe aplicații ai serviciului transport.”[6]

Pentru a evidenția diferențele dintre modele, trebuie să plecăm de la faptul că modelul OSI a fost conceput înainte de a fi inventate protocoalele, iar modelul TCP/IP a apărut ulterior, fiind de fapt doar o descriere a protocoalelor existente. De aceea, modelul OSI nu a fost orientat către un set specific de protocoale, motiv pentru care este destul de general.

Din punct de vedere strict al nivelului transport, **modelul OSI** suportă numai comunicații orientate pe conexiuni în cadrul acestui nivel, deși la nivel de rețea suportă ambele tipuri de comunicații, adică și cea fără conexiuni. Cu toate acestea, acest lucru este important deoarece serviciul de transport este vizibil utilizatorilor. **Modelul TCP/IP** se afla la polul opus, pentru că are numai un mod fără conexiuni la nivelul rețea, dar suportă ambele moduri la nivelul transport, oferindu-le utilizatorilor posibilitatea de a alege, opțiune care este importantă mai ales pentru protocoalele întrebare-răspuns simple.

[6] Rețele de Calculatoare- Andrew Tanenbaum, ediția III, Ed. Agora 1998

3. TCP(Transmission Control Protocol)

3.1. Modelul serviciului TCP (Mavru Anca)

TCP este protocolul Internet-ului orientat pe conexiuni. Protocolul de comunicație de nivel transport a fost definit în mod oficial în RFC 793, pentru a asigura un flux de octeți de la un capăt la celălalt al conexiunii într-o inter-rețea nesigură. TCP a fost proiectat să se adapteze în mod dinamic la proprietățile rețelei Internet și să fie robust în ceea ce privește mai multe tipuri de defecte.

Serviciul TCP se bazează pe crearea unor puncte finale numite socluri (sockets) atât de către emițător cât și de către receptor. Fiecare soclu are o adresă formată din adresa IP a mașinii gazdă și un numar de 16 biți numit port. Prin urmare, trebuie stabilită o conexiune între un soclu de pe mașina emițătoare și unul de pe mașina receptoare cu scopul obținerii unei conexiuni TCP. Având în vedere că un soclu se poate folosi la un moment dat pentru mai multe conexiuni, rezultă că mai multe conexiuni se pot termina la același soclu. La ambele capete ale conexiunii există identificatori care identifică soclurile.

Un aspect foarte important specific TCP este că toate conexiunile sunt duplex integral și punct-la-punct, adică traficul se poate desfășura în ambele sensuri în același timp, fiecare conexiune având exact două puncte finale.

Mai trebuie menționat că o conexiune TCP este un flux de octeți și nu un flux de mesaje iar dimensiunile mesajele nu se conservă de la un capăt la altul.

O altă caracteristică a serviciului TCP constă în informația urgentă. Atunci când un utilizator apasă tasta DEL sau CTRL-C pentru a întrerupe o prelucrare la distanță, aflată deja în execuție, aplicația emițător plasează o informație de control în fluxul de date și o furnizează TCP-ului împreună cu indicatorul URGENT.

Acest eveniment impune TCP-ului întreruperea acumulării de informație și transmisia imediată a informației disponibile deja pentru conexiunea respectivă.

3.2. Antetul segmentului TCP (Mavru Anca)

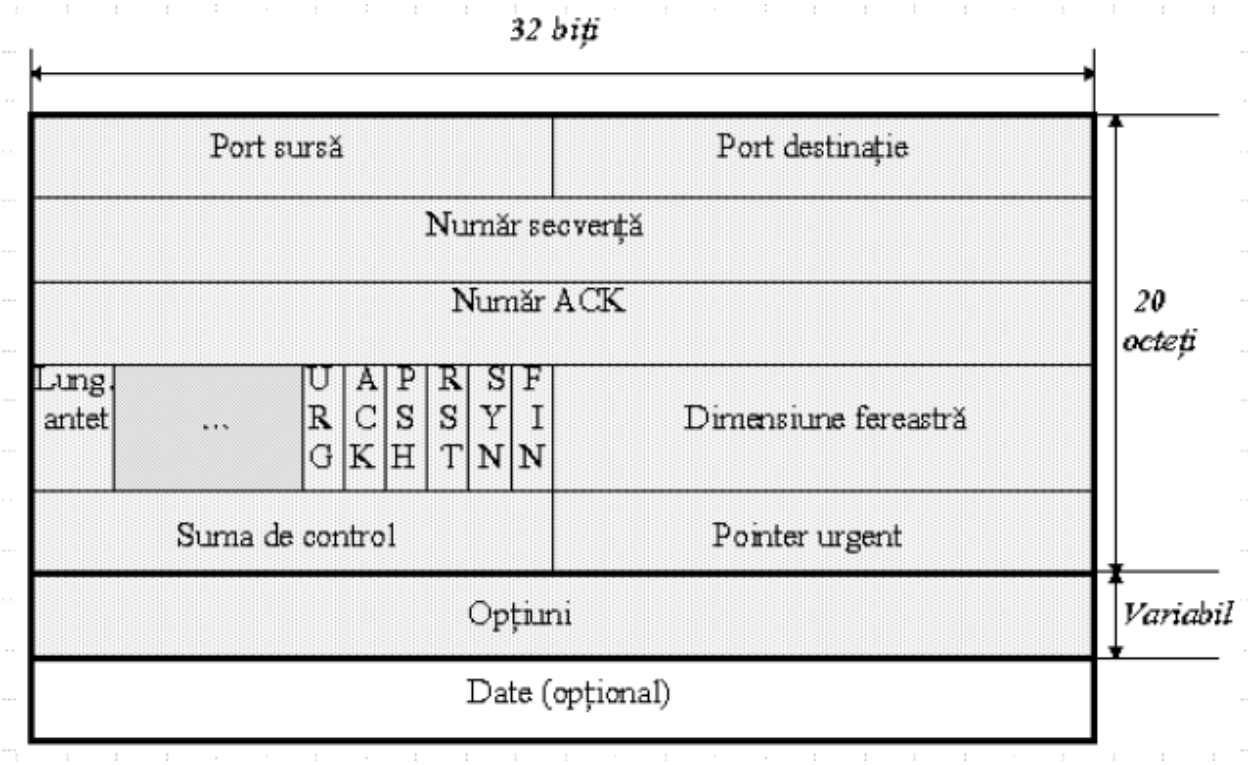

Figura 5: TCP Header

Figura 5 prezintă structura unui segment TCP. Fiecare astfel de segment începe cu un antet fix format dintr-o structură de 20 de octeți dar paote fi urmat și de un set de opțiuni asociate. În cazul în care există opțiuni, ele pot fi urmate de 65 515 octeți de date care formează antetul IP și antetul TCP. În mod frecvent, segmentele care nu conțin octești de date sunt utilizate pentru confirmări și mesaje de control.

Semnificația câmpurilor

- ✓ Port sursă *(Source Port):* punct de acces la serviciile stratului transport TSAP *(Transport Service Access Point),* 16 biți;

socket sursă = adresa IP sursă + port sursă + descriptor de comunicație

- ✓ Port destinație *(Destination Port):* punct de acces la serviciile stratului transport TSAP, 16 biți;

socket destinație = adresa IP dest. + port dest. + descriptor de comunicație

- ✓ Număr secvență *(Sequence Number):* numărul secvenței, pe 32 biți, al primului octet de date din segmentul TCP curent.

Excepție: Pentru segmentele SYN=1 reprezintă numărul secvenței inițiale ISN *(Initial Sequence Number),* primul octet de date fiind ISN+1.

- ✓ Număr ACK *(Acknowledgement Number):* confirmare de tip "piggybacking" care indică numărul de secvență al următorului octet de date ce se așteaptă a fi recepționat.

Câmpurile *Port sursă* şi *Port destinație* identifică punctele finale ale conexiunii şi constituie totodată un identificator al conexiunii.

Câmpurile *Număr de secvență* şi *Număr de confirmare* au semnificația funcțiilor uzuale.

Lungimea antetului TCP indică numărul de cuvinte de 32 de biți care sunt conținute în antetul TCP. Având 3 biți la despoziție, rezultă că această lungime

poate fi de maxim 8 cuvinte, adică spațiul opțional poate avea cel mult 3 cuvinte de 32 biți.

Bitul *URG* poziționat pe 1 arată că *Indicatorul urgent* este valid. Acest indicator este folosit penru a arăta deplasarea în octeți față de numărul curent în secvență la care se află informație urgentă. Această facilitate ține loc de mesaj de întrerupere.

Bitul *ACK* pe 1 indică faptul că *Numărul de confirmare* este valid. Dacă este poziționat pe 0, segmentul în discuție nu conține o confirmare si câmpul *Număr de confirmare* este ignorat.

Bitul *PSH* indică informația forțată, adică trebuie livrată aplicației îndată ce a fost recepționată, fără a mai fi memorată în buffere din rațiuni de eficiență.

Bitul *RST* este folosit pentru a desființa o conexiune care a devenit inutilizabilă din cauza unur defecțiuni ale maşinilor gazdă sau din alte motive. Este de asemenea utilizat pentru a refuza deschiderea unei conexiuni inițiată de un segment invalid de call request.

Bitul *SYN* este folosit pentru stabilirea unei conexiuni. Cererea de conexiune conține SYN=1 şi ACK=0, iar răspunsul la o astfel de cerere este confirmată prin combinația SYN=1 şi ACK=1.

Bitul *FIN* este folosit pentru a încheia o conexiune. El arată că transmițătorul nu mai are informații de transmis. Totuşi închiderea unei conexiuni este un proces lung, în care receptorul încă mai poate primi date întârziate din rețea. De aceea, segmentele SYN şi FIN conțin numere de secvență pentru ca prelucrarea lor să se facă în ordinea firească. În TCP, fluxul de control este tratat prin ferestre glisante de dimensiune variabilă.

Câmpul *Dimensiune fereastră* indică numărul de octeți care pot fi trimişi începând de la octetul confirmat.

Câmpul *Sumă de control* conține suma de control calculată pentru antet, informație utilă şi pseudo-antet. Pseudo-antetul conține adresele IP ale sursei şi detinației, numărul de protocol (pentru TCP este 6) şi câmpul *lungime segment TCP.*

Câmpul *Opțiuni* a fost introdus pentru a permite adăugarea unor facilități suplimentare care nu au fost prevăzute în antetul obişnuit. Cea mai importantă opțiune este aceea de a specifica încărcarea unei maşini cu informație utilă TCP

maximă pe care este dispusă să o accepte. Utilizarea unor segmente de date lungi este mau eficientă decât în cazul segmentelor scurte, mai ales îm cazul unor maşini performante care pot procesa blocuri mari de date. În acest scop, la stabilirea conexiunii fiecare maşină anunță dimensiunea maximă acceptată şi aşteaptă de la parteneri să i se comunice acelaşi lucru. Dacă nici o maşină nu foloseşte această opțiune, mărimea implicită a segmentului de date utilizator este 536 octeți, astfel încât 536+20 antet=556, lumgime minimă pe care trebuie să o accepte orice maşină din Internet. Nici dimensiunile prea mari ale segmentelor nu sunt întotdeauna eficiente.

3.3. Stabilirea conexiunii și controlul congestiei (Ionescu Liviu)

Stabilirea conexiunii

La prima vedere, stabilirea unei conexiuni pare o chestiune simplă, realizabilă doar prin 2 primitive: *Connection Request* şi *Connection Accepted.* Problema se complică atunci când rețeaua pierde pachete, când apare congestia, când are loc multiplicarea pachetelor în rețea etc. Chiar şi la desfacerea unei conexiuni pot apărea probleme dacă nu s-au terminat de recepționat toate pachetele sau dacă mai există pachete stocate în buffere etc. Pachetele de stabilire a conexiunii (Call Request) întârziate pot apărea şi după ce o conexiune a fost deja stabilită. Gazda destinație trebuie să poată sesiza şi rezolva asemenea anomalii şi să refuze duplicarea unei conexiuni. Pentru aceasta se pot folosi mai multe reguli: contor de timp, contor de duplicare, număr de secvență etc. Analiza acestor reguli a condus la concluzia căcel mai potrivit este algoritmul de stabilire a

conexiunii în 3 paşi. Acest protocol nu necesită ca ambele părți să înceapă să transmită cu acelaşi număr de secvență. Gazda 1 alege un număr de secvență *x* pe care îl transmite gazdei 2 în pachetul *connection request.* Gazda 2 răspunde cu connection accepted, confirmă numărul de secvență *x* şi transmite şi numărul propriu de secvență *y.* Gazda 1 va confirma recepția numărului de secvență *y* în primul pachet de date trimis spre gazda 2. Dacă în timp va sosi la gazda 2 un pachet duplicat de la o cerere de conexiune mai veche, gazda 1 nu va şti în primul moment despre acest lucru. Gazda 2 îi va trimite confirmarea cererii că 1 a încercat să stabilească o conexiune, iar aceasta din urmă ştiind că nu a făcut o asemenea cerere, o va refuza. Cele două scenarii posibile descrise anterior se pot vedea în Figura 6.

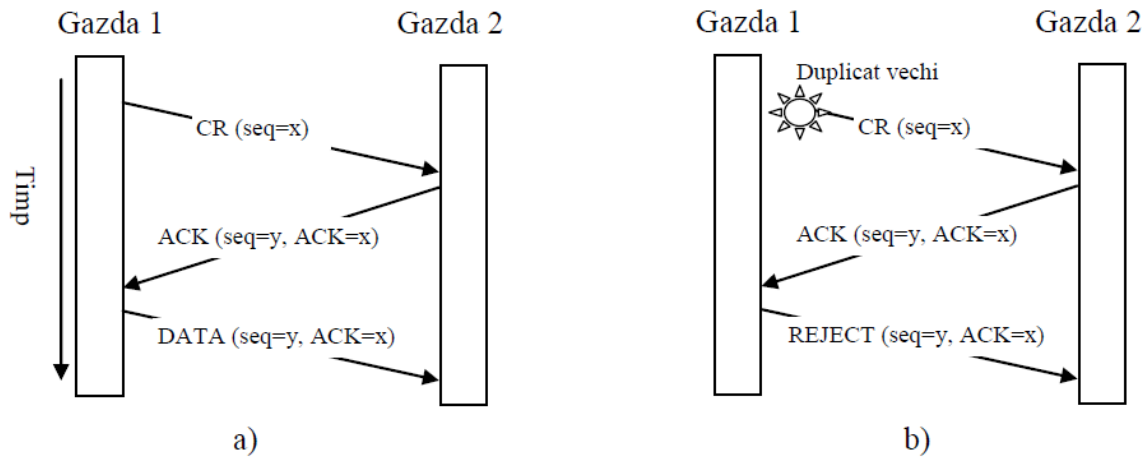

Figura 6: Algoritmul stabilirii conexiunii în 3 pași: pentru cazul normal (a) și cazul cu pachete duplicate (b)

Controlul congestiei

Congestia apare în rețele de calculatoare atunci când încărcarea cu date a unei rețele depăşeşte posibilitățile ei de transfer a datelor. Încărcarea unei rețele la un moment dat nu depinde numai de capacitatea ei de transmitere, adică de ratele biților suportate de canalele de comunicație ci şi de erorile care apar în mediul de transmisie, de viteza de prelucrare în noduri şi de mecanismele de confirmare folosite de receptor. În lipsa unui algoritm de control al congestiei, odată aprută ea are tendința de a se automenține ca urmare a repetării transmiterii pachetelor pierdute sau neconfirmate. Cea mai simplă metodă de a controla congestia este de nu intrduce un nou pachet în rețea până nu a fost confirmat precedentul. Confirmarea individuală reduce mult rata de transfer efectivă şi, de cele mai multe ori, nu este o soluție practică. Principala modalitate de control a congestiei este reducerea ddebitului srsei. Sarcina controlului congestiei revine în primul rând nivelului transport, deşi unele încercări se pot face şi la nivel rețea sau chiar legătură de date.

Controlul congestiei presupune în primul rând detecția acesteia şi apoi reducerea progresivă sau bruscă a debitului sursei. La stabilirea unei conexiuni TCP transmițătorul şi receptorul aleg o fereastră de dimensiune convenabilă ambilor. De exemplu receptorul poate propune o fereastră (lungimea segmentului) egală cu dimensiunea buffer-ului său, pe care emițătorul o poate accepta sau nu. Nici unul dintre ei nu va şti dacă rețeaua această capacitate de transfer şi mai ales dacă ea nu se modifică în timp. Cel mai corect este ca emițătorul să mențină două ferestre: una acceptată de receptor şi alta acceptată de rețea, numită *fereastră de congestie*, şi să transmită cu cea minimă dintre ele. Problema este de a determina valoarea ferestrei de congestie. O posibilitate este ca la stabilirea conexiunii, sursa să transmită un segment de fereastră maximă şi aşteaptă confirmarea. Dacă aceasta vine într-un timp mai mic decât cel maxim convenit, mai adaugă un segment şi aşteaptă din nou confirmarea. Procedura se repetă până când confirmarea vine după expirarea timpului de confirmare şi astfel poate stabili valoarea maximă a ferestrei de congestie. O variantă mai rapidă de aflare a ferestrei de congestie a rețelei este dublarea lungimii

segmentului curent față de cel precedent până la depăşirea timpului de confirmare. În acest caz mărimea segmentului creşte exponențial, dar precizia, dar precizia determinării ferestrei de rețea este mai mică. Internetul combină cele 2 variante, folosind creşterea exponențială la început şi apoi creşterea liniară. Se începe cu segmentul maxim de 64 Kocteți. Dacă se depăşeşte timpul de confirmare, se reduce lungimea segmentului la jumătate, adică 32 K şi se verifică din nou timpul de confirmare. Dacă este bun, lungimea maximă se va afla între cele două valori. Pentru a fi determinată exact, până la lungimi de 32 K se foloseşte creşterea exponențială, iar peste 32 creşterea liniară. Procedurile de confirmare / retransmisie au la bază un contor de timp.

TCP foloseşte mai multe contoare de timp, dintre care cel mai important este **contorul de retransmisie**. Atunci când este transmis un segment, se porneşte un contor de retansmisie. Dacă segmentul este confirmat înainte de expirarea timpului, contorul este oprit. Dacă expiră timpul de retansmisie şi nu s-a făcut confirmarea, segmentul este retansmis şi contorul repornit. Problema care se pune este cât de mare trebuie să fie acest timp de expirare?. Dacă la nivel legătură de date întârzierea aşteptată este uşor predictibilă, la nivel transport este foarte greu de pezis când se va întoarce o confirmare. O limită inferioară este desigur timpul de propagare dus întors pe ruta cea mai scurtă. O valoare maximă dată de propagarea dus-întors pe ruta cea mai lungă pote fi şi ea lată în calcul, dar nu se poate cunoaşte timpii de aşteptate în cozi şi de prelucrare în noduri. Soluția este utilizarea unui algoritm dinamic care să ajusteze periudic timpul de retransmisie pe baza unor măsurători ale performanței rețelei. Algoritmul utilizat în mod curent de TCP a fost elaborat de Jacobson în 1988.

3.4. TCP fără fir (Mavru Anca)

În principiu protocoalele de transport ar trebui să fie independente de tehnologia nivelului rețea, adică mai precis, pentru TCP nu ar trebui să conteze dacă IP rulează peste o rețea cablu sau o rețea radio.

Cu toate acestea, TCP a fost implementat cu optimizări pentru rețelele cu cabluri, optimizări care nu sunt valabile și în cazul rețelelor radio. În concluzie, în practică este foarte important pe ce tip de rețea se lucrează, TCP având abordări diferite pentru diverse tipuri de rețele.

Chiar dacă din punct de vedere logic o implementare în care se ignoră proprietățile de transmisie fără fir este corectă, performanțele sunt foarte nesatisfăcătoare.

Majoritatea implementărilor TCP pleacă de la premisa că depășirile de timp sunt cauzate de congestive și nu de ăpierderea pachetelor, deci principal problemă ar fi algoritmul de control al congestiei. Conform algoritmului startului lent al lui Jacobson atunci când expiră un contor, TCP încetinește ritmul și trimite pachete cu mai puțină vigoare. O soluție pentru aceastp problemă ar fi reducerea încărcării rețelei pentru a diminua problemele cauzate de congestie.

Una dintre principalele probleme ale legăturilor bazate pe transmisia fără fir este că pierd tot timpul pachete, motiv pentru care sunt profund nefiabile. O rezolvare a acestei probleme ar fi retrimiterea pachetelor pierdute cât mai repede posibil.

Abordarea problemei pierderii pachetelor este diferită în cazul rețelelor cu cablu față de cele fără fir. În primul caz, dacă se pierde un pachet, emițătorul ar trebui să încetinesacă ritmul. La polul opus, dacă se pierde un pachet pe o rețea

fără fir, emițătorul trebuie să mărească ritmul. Problema apare atunci când emițătorul nu cunoaște tipul rețelei și astfel luarea unei decizii este îngreunată.

În 1995, Bakne și Badrinath au propus ca soluție pentru această problemă, **TCP indirect.** Mai exact, conexiunea TCP este separată în două conexiuni: prima conexiune pleacă de la emițător la stația de bază, iar cea de-a doua leagă stația de bază de receptor, așa cum se poate observa și în Figura 7.

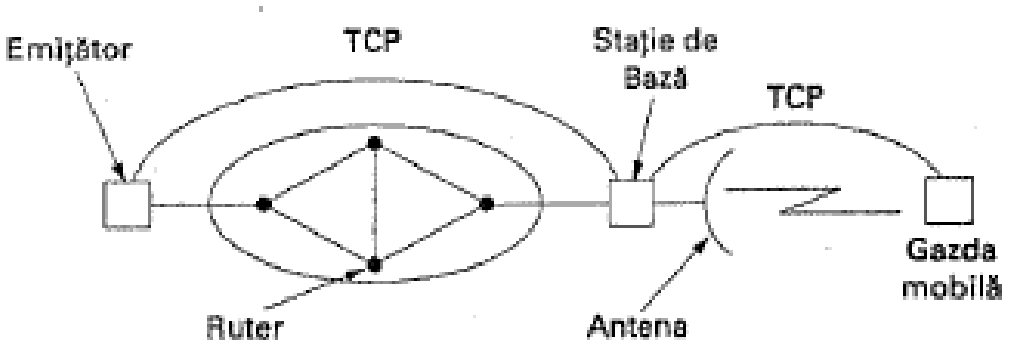

Figura 7: Spargerea conexiunii TCP în două conexiuni[7]

Rolul stației de bază este de a copia pachetele din cele două conexiuni în ambele direcții. Deși această schemă are avantajul omogenității conexiunilor, este negată semantica TCP.

---

[7] Rețele de Calculatoare- Andrew Tanenbaum, ediția III, Ed. Agora 1998

Pentru că era nevoie de o soluție care să nu încalce semantica TCP, Balakrishnan a propus unele modificări în codul nivelului rețea din stația de bază. De asemenea, algoritmul de mai sus oferă și soluții pentru problema pierderii segmentelor generate de către gazda mobilă.

În cazul în care stația de bază constată o pauză în interiorul domeniului numerelor de secvență, utilizează o opțiune TCP pentru a genera o cerere pentru o repetare selectivăa octetului lipsă. Astfel legătura fără fir devine mai fiabilă în ambele sensuri, fără prețul modificării semanticii TCP. Ar mai fi de menționat că comunicația fără fir mai poate afecta și alte domenii decât cel al performanțelor, cum ar fi lărgimea de bandă.

3.5. **Vulnerabilitățile TCP** (Mavru Anca)

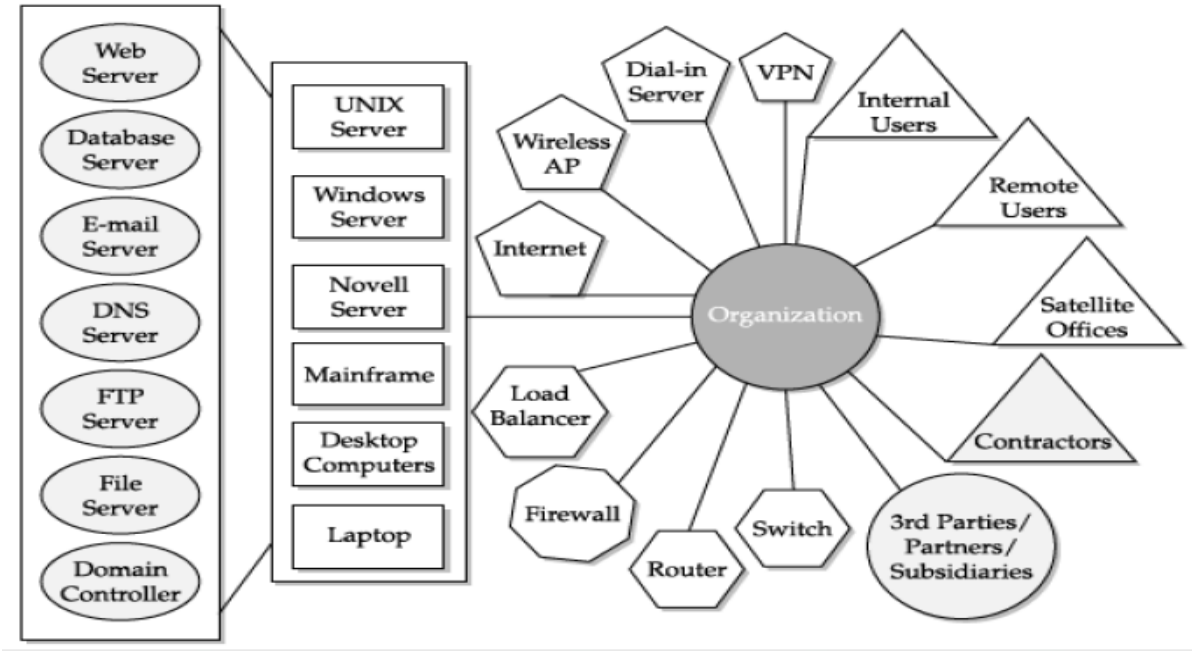

Figura 8: Vulnerabilități[8]

[8] http://profs.info.uaic.ro/~busaco/

Multe implementări TCP utilizează algoritmi uşor predictibili de generare a numerelor de secvență a pachetelor. Acest lucru, corelat cu incapacitatea de autentificare, creează premisele unor fraude privind interceptarea, modificarea sau furtul unor pachete.

Se pot stabili conexiuni frauduloase la sisteme, pentru accesarea unor fişiere importante sau, mai subtil, pentru instalarea unor "trape" pregătitoare ale unor accese ulterioare nestingherite pe acel sistem.

Cele mai des întâlnite vulnerabilități[9] ale TCP-ului sunt:

- Connection-flooding attacks (Naphta and FIN-WAIT-2 flooding attacks)
- TCP send buffer attacks (Netkill and closed windows)
- TCP receive buffer attacks

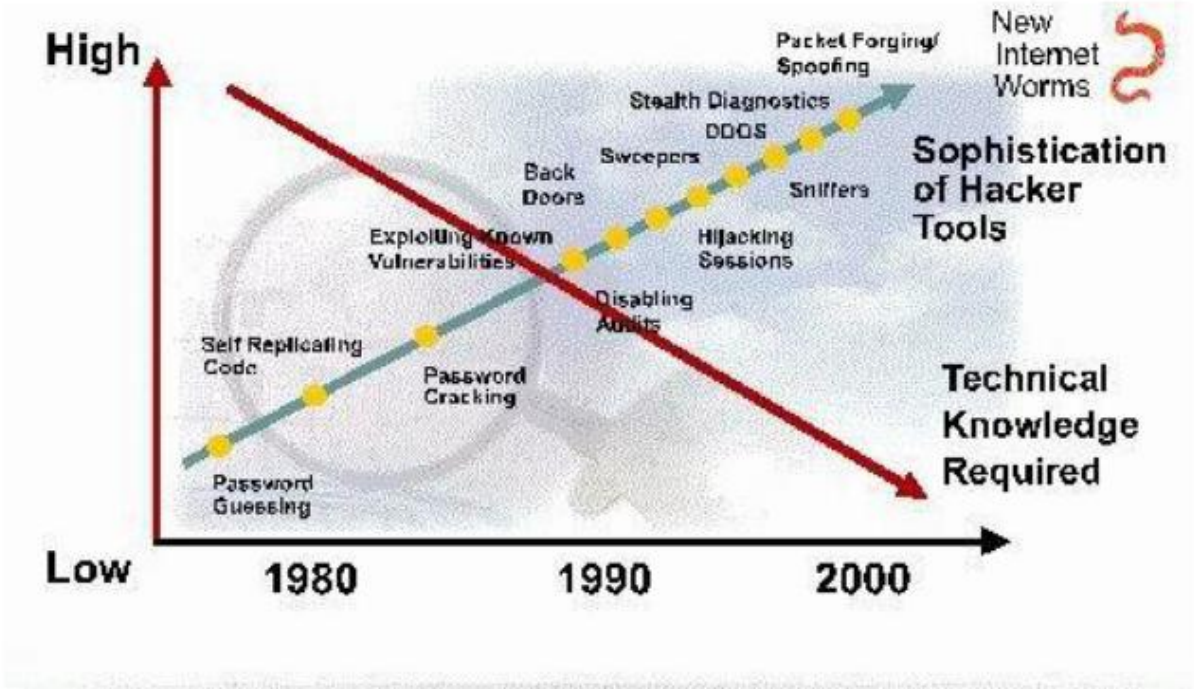

Figura 9: Evoluția tipurilor de atacuri la securitate[10]

[9] Some insights about the recent TCP DoS (Denial of Service) vulnerabilities- Fernando Gont, HACK.LU 09 Conference October 28-30, 2009. Luxembourg

[10] http://www.cisco.com/warp/public/778/security/vuln_stats_02-03-00.html

Connection-flooding attacks (Naphta)

Crearea și menținerea unei conexiuni TCP necesită ca memoria sistemului să fie distribuită între conexiunile TCP locale și cele care nu sunt locale. Faptul că memoria sistemului este o resursă limitată este o slăbiciune exploatată de vectorul cunoscut sub numele de Naphta, care realizează un atac de tipul DoS (Denial of Service).

Atacul de tip Denial of service (DoS) are ca efect dezafectarea sau coruperea sistemelor, rețelelor sau serviciilor, cu intenția de a întrerupe accesul la servicii utilizatorilor în drept. Atacurile DoS implică fie căderea sistemului, fie încetinirea acestuia până în punctul în care devine inutilizabil. Atacurile DoS pot fi extrem de simple precum ştergerea sau modificarea informațiilor. În majoritatea cazurilor, atacul presupune rularea unui hack sau script. Atacatorul nu are nevoie apriori de acces la ținta sa deoarece o modalitate de acces reprezintă tot ceea ce este necesar. Pentru acest motiv, atacurile DoS attacks se numără printre cele mai temute.

Atacuri DoS au mai multe forme de manisfestare. În final, ele întrerup accesul persoanelor autorizate de a folosi servicii prin utilizarea în întregime a resurselor calculatorului victimă. Alte exemple de amenințări DoS:

Ping of death

Acest tip de atac modifică porțiunea IP a header-ului, indicând faptul că sunt mai multe date în pachet decât în mod real; rezultatul este căderea sistemului.

SYN flood attack

Acest atac deschide la întâmplare mai multe porturi TCP,ocupând sistemul atacat cu atâtea cereri false astfel încât sesiunile reale nu mai pot avea loc. Acest tip de atac se poate realiza prin intermediul analzoarelor de protocoale sau altor programe.

4. Concluzii (Ionescu Liviu)

Protocoalele de transport trebuie să fie capabile să controleze conexiunea în sisteme nefiabile. Stabilirea conexiunii este complicată de existența pachetelor duplicate întârziate, care pot apărea la momente inoportune. Pentru a le face față, stabilirea conexiunii trebuie făcută prin intermediul protocoalelor cu înțelegere în trei pași. eliberarea unei conexiuni este mai simplă decât stabilirea sa, dar este încă departe de a fi banală.

Chiar și în cazul unui nivel rețea complet fiabil, nivelul transport are sufficient de mult de lucru. El trebuie să controleze toate primitivele de serviciu, toate conexiunile și contoarele de timp și trebuie să aloce și să utilizeze credite.

Principalul protocol de transport în Internet este TCP. El utilizează un antet de 20 de octeți pentru toate segmentele. Segmentele pot fi fragmentate de ruter în interiorul Internetului, deci calculatoarele gazdă trebuie să fie pregătite să le asambleze. S-a depus un mare efort pentru optimizarea performanțelor TCP-ului, utilizând algoritmii Nagle, Clark, Jacobson, Karn și alții.

5. Bibliografie

1. Rețele de Calculatoare- Andrew Tanenbaum, ediția III, Ed. Agora 1998

2. TCP/IP Illustrated, Volume 1-The Protocols- W. Richard Stevens

3. Some insights about the recent TCP DoS (Denial of Service) vulnerabilities- Fernando Gont, HACK.LU 09 Conference October 28-30, 2009. Luxembourg

4. Curs ARI și SC, an III CTI

5. http://en.wikipedia.org/wiki/Transmission_Control_Protocol