UNIVERSITATEA TEHNICĂ CLUJ-NAPOCA

DEPARTAMENTUL ELECTROTEHNICĂ ŞI MĂSURĂRI

Cursul 11 PROTOCOALELE DE TRANSPORT TCP ŞI UDP

Introducere curs	În acest curs sunt prezentate protocoalele de transport TCP şi UDP
Obiective curs	• Introducere. • TCP (Transmission Control Protocol). • Prezentarea funcțiilor de transport. • Analiza drumului parcurs de un mesaj. • Porturi şi socket-uri. • Comunicația dintre TCP şi nivelul superior. • Porturi active şi pasive. • Temporizatoarele TCP. • Temporizarea retransmisiei. • Timpul de pasivitate. • Timpul de perseverență. • Timpul de menținere în stare activă şi timpul de aşteptare. • Blocuri de control a transmisiei şi fluxului. • Formatul segmentului TCP. • Conexiunile TCP. • Stabilirea unei conexiuni. • Transferul datelor. • Închiderea unei conexiuni. • UDP (User Datagram Protocol.) • Prezentare. • Formatul segmentului UDP.
Durată medie de studiu individual	Durata medie de studiu individual: 100 minute. Acest interval de timp presupune parcurgerea continutului cursului şi rezolvarea testelor de autoevaluare

Cuprins:

11.1. Introducere. 2

11.2 TCP *(Transmission Control Protocol).* 3

11.2.1. Prezentarea funcțiilor de transport. 3

11.2.2. Analiza drumului parcurs de un mesaj, între aplicația sursă şi cea destinație. 4

11.2.3. Porturi şi socket-uri. 6

11.2.4. Comunicația dintre TCP şi nivelul superior. 9

11.2.5. Porturi active şi pasive. 10

11.2.6. Temporizatoarele TCP 10

11.2.7. Blocurile de control a transmisiei şi fluxului 11

11.2.8. Formatul segmentului TCP 12

11.2.9. Conexiunile TCP. 14

11.3. UDP (User Datagram Protocol). 19

11.3.1. Prezentare. 19

11.3.2. Formatul segmentului UDP 19

TESTE DE AUTOEVALUARE 21

RĂSPUNSURI 23

Bibliografie 23

11.1. Introducere.

Protocoalele corespunzătoare nivelului **OSI 4** (transport), pe care le foloseşte suita **TCP/IP**, sunt **TCP** *(Transmission Control Protocol)* şi **UDP** *(User Datagram Protocol)*.

TCP este unul dintre cele mai cunoscute protocoale de transport, în rețelele de date. Acest protocol a fost dezvoltat dintr-o implementare originală în ARPANET, iar acum se foloseşte, pentru transferul datelor, în întreaga lume.

Deşi în această carte protocoalele şi soluțiile adoptate par să respecte cerințele prevăzute în standardele OSI, de această dată **TCP** a fost "sursa de inspirație" pentru modelarea nivelului de transport. Teoretic, protocolul nivelului de transport nu este un lucru simplu, datorită nesiguranței în livrarea pachetelor la nivelul inferior (IP OSI 3).

IP nu garantează livrarea la destinație a datagramelor, deoarece nu este un protocol bazat pe conexiune. Sarcina sa este doar aceea de a "împinge" (ruta) pachetele de date spre destinație.

În cazul apariției unor erori, sau a congestiei, există riscul de-a abandona pachete, fără a notifica terminalul sursă, despre aceasta. Constatarea stării datagramelor trimise şi cererea de re-transmitere a informațiilor abandonate, sau degradate, cade în sarcina protocolului de transport [6].

Cei mai mulți utilizatori cred că **TCP** şi **IP** sunt o pereche strânsă şi nedespărțită. **TCP** poate să fie, însă, folosit şi împreună cu alte protocoale de nivel OSI 3 (rețea), aşa cum şi **IP** poate transmite prin rețea alte protocoale de transport, aşa cum se poate vedea în fişierul `/etc/protocol`.

11.2 TCP *(Transmission Control Protocol)*.

11.2.1. Prezentarea funcţiilor de transport.

TCP (*Transmission Control Protocol* - protocol cu controlul transmisiei) furnizează nivelelor superioare şi aplicaţiilor servicii de transport a informaţiilor.

Principalele caracteristici ale unui serviciu de transport, în sensul asigurării livrării la destinaţie în bune condiţii a datelor, sunt următoarele:

• **Transmisia Full duplex**, permite fiecărui capăt al conexiunii să transmită date în orice moment chiar şi simultan.

• **Punctualitatea** *(Timeliness)*, prin utilizarea temporizatoarelor se asigură că datele ajung la destinaţie într-un interval de timp rezonabil

• **Succesiune ordonată** *(ordered)*, datele trimise de către o aplicaţie sunt recepţionate, la capătul celălalt, în aceeaşi ordine. Aceasta are loc cu toate că segmentele pot să parcurgă reţeaua pe căi şi în intervale de timp diferite. TCP re-asamblează mesajele (segmentele) în ordinea firească, înainte de a le trimite spre nivelul superior.

• **Etichetarea**, capetele îşi pun de acord prioritatea şi o valoarea siguranţei conexiunii.

• **Controlul fluxului**, **TCP** poate regla fluxul informaţiei prin utilizarea tampoanelor de memorie şi prin limitarea dinamică dimensiunii ferestrei.

• **Corecţia erorilor**, prin suma de control se asigură un transfer de date lipsit de erori (în limita algoritmului folosit) [1].

Prin realizarea unei **conexiuni virtuale**, se garantează că datagramele trimise sunt recepţionate corect. **TCP** foloseşte un mecanism "cu confirmare de primire" ceea ce permite o comunicaţie sigură, care în cazul degradării datelor, sau al pierderii lor pe drum, cere retransmiterea lor. **TCP** nu este doar un program, sau o simplă rutină apelată de un program, ci o colecţie de componente software, care defineşte întregul nivel de transport.

TCP permite gestionarea fluxului datagramelor, dinspre nivelele superioare spre nivelul **IP** şi a celor care vin dinspre **IP** şi trebuie trimise spre cele superioare, cu garantarea riguroasă a respectării priorităţilor.

Acest protocol trebuie să fie capabil să încheie o aplicaţie, aflată deasupra şi care aşteaptă datagrame, sau să izoleze erori de la nivelele inferioare. Acest lucru este posibil doar dacă întreţine o tabelă de stare, cu toate şirurile de date care vin din afara nivelului său.

Gruparea acestor servicii, într-un nivel separat, permite aplicaţiilor să fie proiectate fără să se ţină seama de controlul fluxului de date sau de siguranţa acestora. Fără acest nivel, fiecare aplicaţie ar trebui să aibă propriile mecanisme de control al erorilor şi fluxului, aşa cum sunt aplicaţiile de transfer de date prin modem.

TCP este poziţionat între nivelul **IP** şi cel al aplicaţiilor, aşa cu se poate vedea în **figura 11.1**, deci poate fi găsit doar în dispozitivele care procesează acest tip de datagrame. Dispozitivele care doar manevrează pachete, în vederea rutării (porţi de acces, rutere), nu conţin această componentă. Ele fac parte din nivelul OSI 3 (reţea). Există, însă şi porţi de transfer care au funcţii de filtrare *(firewall)* a serviciilor. Pentru aceasta este necesară prezenţa componentelor de nivel OSI 4 (transport).

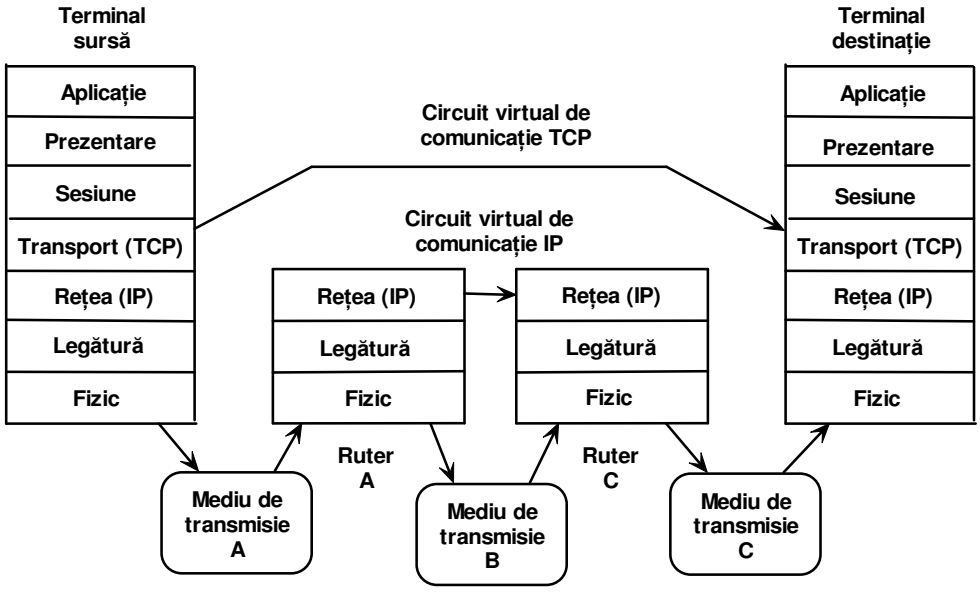

Fig. 11.1. Circuit virtual **TCP**, între un terminal sursă şi unul destinație.

Deoarece **TCP** este un protocol bazat pe conexiune, trebuie să asigure transferul datagramelor între sursă şi destinație *(end-to-end communications)*. Aceasta se realizează prin faptul că terminalul destinație răspunde cu un mesaj de confirmare a primirii datagramei (acknowledgement), sau cu un mesaj de eroare, dacă datagrama a fost recepționată eronat. Se stabileşte astfel un **circuit virtual,** între terminalul sursă şi cel destinație.

11.2.2. Analiza drumului parcurs de un mesaj, între aplicația sursă şi cea destinație.

Pentru a clarifica rolul **TCP**, trebuie să se urmărească un mesaj, de-a lungul căii dintre aplicația sursă şi cea de destinație. Pentru început, lucrurile vor fi prezentate mai simplu, urmând ca subiectul să fie dezvoltat la momentul potrivit. Aplicația de pe terminalul sursă trimite asincron mesajul, nivelului inferior (în cazul modelului **TCP/IP** direct nivelului de transport), sub forma unui şir de caractere.

Această metodă poate transfera blocuri cu dimensiune variabilă, ceea ce o deosebeşte de cele folosite de alte protocoale de transport care transferă blocuri de date de dimensiune fixă,. Deşi metoda din urmă pare mai convenabilă, dimensiunea fixă a blocurilor nu permite reglarea dinamică a fluxului de date.

TCP primeşte şirul de octeți (caractere) pe care îl divizează în **secvențe**, pe care le încapsulează în **segmente TCP** (unitatea de date a protocolului de transport).

În procesul de segmentare şi asamblare, în fața datelor se adaugă un antet care conține informații de control. **Antetul TCP** conține, printre alte informații, numărul de ordine a secvenței *(sequence number)*, lungimea segmentului şi suma de control. Lungimea segmentului este fixată administrativ, sau se determină de **TCP** în funcție de alți parametri. Lungimea segmentului **TCP** nu este egală cu lungimea pachetului **IP**, cu care este confundat de mulți utilizatori!

Atunci când se cere comunicația în ambele sensuri, cum ar fi cazul aplicațiilor **Telnet** sau **FTP**, se stabileşte mai întâi un **circuit (conexiune) virtual** între sursă şi destinație. Acest proces începe prin trimiterea, de către componenta **TCP** a terminalul sursă spre terminalul destinație, a unei cereri de conexiune **TCP**. În mesaj se include un număr *(socket number)* care identifică unic conexiunea, deschisă de aplicația de pe terminalul sursă. Componenta **TCP**, de pe terminalul destinație, alocă propriul număr de identificare a conexiunii şi îl trimite înapoi terminalului care a inițiat cererea de conexiune. Aceste două numere definesc "canalul de comunicație" (**conexiunea** sau **circuitul virtual**) între cele două terminale, până la închiderea acestuia.

După stabilirea circuitului virtual, **TCP** trimite segmentele nivelului inferior (**IP**), care le încapsulează într-un pachet (datagramă) împreună cu propriile informații de control. **IP** poate să

efectueze, asupra segmentului, operaţii proprii de fragmentare şi reasamblare, fără ca **TCP** să observe acest lucru. După ce datele străbat reţeaua (parcurgând şi celelalte nivele, inferioare lui **IP**) pachetul ajunge la componenta **IP** de pe terminalul destinaţie. Acesta decapsulează pachetul, extrage segmentul şi îl trimite nivelului superior **TCP**, care îl procesează şi îl trimite spre nivelul superior (aplicaţie).

Dacă mesajul a fost împărţit în mai multe segmente, componenta **TCP** a destinaţiei reasamblează mesajul (şirul de octeţi) utilizând numărul de ordine al secvenţei, conţinut în antetul fiecărui segment. Dacă un segment lipseşte, sau este degradat (ceea ce se poate determina cu ajutorul sumei de control), **TCP** întoarce, spre sursă, un mesaj care conţine numărul secvenţei greşite. Componenta **TCP**, de pe terminalul sursă, poate astfel să repete trimiterea segmentului lipsă.

Dacă mesajul încape într-un singur segment, după compararea sumei de control cu cea calculată la sosire, iar dacă suma se potriveşte, componenta **TCP** de pe terminalul destinaţie răspunde cu un mesaj de confirmare (ACK - *acknowledgment*), iar dacă nu, cu o cerere de retransmitere.

Pentru a preveni aglomerarea tamponului de recepţie, componenta TCP, de pe terminalul destinaţie, poate realiza un control al fluxului. Aceasta se realizează cu ajutorul unui procedeu de "decupare" (*windowing* - ferestruire) a numărului de segmente pe care le poate trimite sursa, între două mesaje de confirmare.

Valoarea care reprezintă mărimea ferestrei (*window number*) , înscrisă în antetul segmentului, se alege în funcţie de capacitatea terminalului destinaţie de a primi şi procesa segmentele. Fiecare terminal trebuie să respecte mărimea ferestrei cerută de celălalt. După trimiterea unei succesiuni de segmente în număr (n) egal cu dimensiunea ferestrei, sursa aşteaptă mesajul de confirmare *(ACK)*, prin care se cere transmiterea unei noi succesiuni de segmente, începând de la (n+1).

În mesajul de confirmare se poate afla aceeaşi valoare a dimensiunii ferestrei, sau una nouă depinzând de numărul de segmente pierdute între două confirmări. Acest mecanism de negociere dinamică a fluxului *(Three Way Handshaking Protocol)* permite diminuarea, la nevoie, a ratei de transmitere a segmentelor între terminale, astfel încât numărul de segmente (şi pachete) trimise dar pierdute, să fie cât mai mic. Utilizarea unei ferestre variabile *(sliding window)* este mult mai eficientă decât trimiterea, unui singur segment, urmată de confirmarea primirii acestuia. O fereastră variabilă presupune trimiterea mai multor segmente succesive, urmată de un singur mesaj de confirmare. Configurarea optimă a acestui mecanism (protocol) asigură o utilizare eficientă a reţelei şi o capacitate de transfer *(throughput)* mai ridicată.

Dacă apar probleme de transfer, înainte de încheierea unei succesiuni de segmente (m<n), după un anumit interval de timp, destinaţia trimite un mesaj de confirmare, prin care se cere o nouă succesiune de segmente, dar care începe cu segmentul (m+1).

Ca la toate protocoalele bazate pe conexiune, temporizarea este foarte importantă. Utilizarea "temporizatoarelor" asigură că nu se aşteaptă, un timp nedefinit, primirea unui mesaj de confirmare (ACK), sau un mesaj de eroare. Dacă timpul expiră, se poate presupune că datagrama transmisă este incompletă, sau s-a pierdut şi, de obicei, într-un astfel de caz se trece la retransmiterea ei.

Temporizatoarele pot crea şi probleme. Documentaţia **TCP** prevede confirmarea doar a primei datagrame din fereastră, care a ajuns cu bine la destinaţie, dar aceasta nu poate gestiona, cum trebuie, fragmentele recepţionate. Dacă un mesaj este compus din mai multe datagrame, iar acestea nu ajung în ordinea transmiterii,

TCP nu poate confirma primirea lor decât după ce toate au ajuns cu bine la destinaţie. Deci, chiar dacă toate datagramele, mai puţin una din interiorul unei secvenţe, ajung cu bine la destinaţie, dar timpul de aşteptare expiră, toate datagramele (recepţionate cu bine, de altfel) sunt abandonate şi se cere retransmiterea lor, începând cu datagrama pierdută. Pentru mesaje mari, aceasta poate duce la trafic suplimentar în reţea.

Dacă destinaţia recepţionează aceeaşi datagramă în dublu exemplar (situaţii datorate retransmiterii după expirarea timpului de aşteptare, sau datorită unei retransmisii datorate nivelului **IP**), **TCP** abandonează datagrama fără să trimită un mesaj de eroare spre sursă. Sistemul sursă are grijă doar ca mesajul să ajungă la destinaţie, fără să fie interesat de numărul de copii.

TCP nu are definite funcții de confirmare negativă (NAK - *negative acknowledgment*), ci se bazează pe temporizator pentru a detecta lipsa confirmării. Dacă timpul de aşteptare, după transmiterea datagramei, expiră înainte de primirea unui mesaj de confirmare, se presupune că aceasta s-a pierdut şi se retransmite. Componenta **TCP** a terminalului sursă ține, într-un tampon, copii ale tuturor datagramelor confirmate, până când toate secvențele au fost confirmată corect. Când acest lucru se întâmplă, temporizatoarele sunt oprite şi datagramele confirmate sunt îndepărtate din tampon. Există cazuri în care confirmarea primirii corecte a datagramelor o asigură aplicația. Există, de asemenea, situații în care este nevoie ca un segment să părăsească cât mai repede terminalul, fără să fie nevoie de confirmarea primirii. Pentru aceste cazuri, **TCP** permite să trimită, peste rând, o datagramă primită de la nivelul superior, fără să o pună şi să o țină în tampon. Acest tip de datagrame sunt marcate cu un atribut înscris în câmpul **Psh** din antetul segmentului **TCP** [1].

11.2.3. Porturi şi socket-uri.

Toate aplicațiile (sau componentele nivelelor superioare) care folosesc **TCP** sau **UDP**, au un număr de identificare, numit port. Teoretic, numerele de port pot fi alocate la întâmplare, după dorința administratorului de rețea. Există însă convenții de alocare, pentru a permite o comunicație bună între componentele, de transport, implementate pe cele două terminale. Aceasta permite ca numărul portului să identifice tipul serviciului pe care îl oferă (sau cere) un sistem, celuilalt.

Convențiile de alocare a numărului de port se regăsesc în orice implementare **TCP/IP**, sub forma unui fişier numit *services*. Convențiile utilizate pentru numerotare porturilor sunt controlate de IANA *(Internet Assigned Numbers Authority)* în urma propunerilor făcute de către producătorii de software. Lista cu numărul şi semnificația serviciilor se actualizează periodic şi poate fi descărcată din Internet. Un exemplu de astfel de fişier (fragment) se poate vedea la sfârşitul acestui paragraf. Numerele 0 şi 255 sunt rezervate, ca de altfel şi altele, pentru scopuri administrative. Conținutul unui fişier `/etc/services`, se poate vedea mai jos.

```
# /etc/services
# NUMERELE PORTURILOR (Aşa cum sunt publicate periodic de IANA).
# (ultima actualizare 26-11-2007)
# Numerele de porturi sunt împărțite în trei domenii: "Porturi bine cunoscute",
# "Porturi recomandate" şi "Porturi alocate dinamic şi/sau cu utilizare privată".
# Porturile bine cunoscute sunt cuprinse între 0 şi 1023.
# Porturile recomandate sunt cele între 1024 şi 49151.
# Porturile alocate dinamic şi/sau cu utilizare privată
sunt cele între 49152 şi 65535.
### NUMERELE CARE NU SUNT ATRIBUITE AR TREBUI SĂ NU FIE FOLOSITE.
# IANA VA ALOCA NUMĂRUL PORTULUI DUPĂ CE APLICAȚIA A FOST ACCEPTATĂ. ###
# PORTURILE BINE CUNOSCUTE
# Aceste porturi au fost alocate de către IANA şi sunt folosite, pe cale
mai multe sisteme, doar de procesele, sau programele, folosite de sistem sau de utilizatorii
privilegiați.
# Porturile sunt folosite de TCP (RFC793) pentru a numi "capetele"
unei conexiuni logice de lungă durată, folosită la transmiterea datelor.
# În scopul furnizării de servicii pentru cereri necunoscute, se defineşte un port de contact.
Această listă precizează portul utilizat de un proces server ca port de contact. Porturile de
contact sunt numite uneori "Porturi bine cunoscute".
# Aceleaşi porturi se folosesc, în măsura posibilităților, şi de către UDP
[RFC768].
# Domeniul de atribuire a numerelor de porturi, administrate de IANA, este
între 0 şi 1023.
# Alocarea porturilor:
# Cuvânt          Valoare
# cheie           zecimală   Descriere                      Recomandări
# -------         -------    -----------                     ----------
                  0/tcp    Rezervat
                  0/udp    Rezervat
#                          Jon Postel <postel@isi.edu>
tcpmux            1/tcp    TCP Port Service Multiplexer
tcpmux            1/udp    TCP Port Service Multiplexer
#                          Mark Lottor <MKL@nisc.sri.com>
compressnet       2/tcp    Management Utility
compressnet       2/udp    Management Utility
compressnet       3/tcp    Compression Process
compressnet       3/udp    Compression Process
#                          Bernie Volz <VOLZ@PROCESS.COM>
#                 4/tcp    Unassigned (Nealocat)
```

```
#                4/udp    Unassigned (Nealocat)
rje              5/tcp    Remote Job Entry
rje              5/udp    Remote Job Entry
#                         Jon Postel <postel@isi.edu>
#                6/tcp    Unassigned
#                6/udp    Unassigned
echo             7/tcp    Echo
echo             7/udp    Echo
#                         Jon Postel <postel@isi.edu>
#                8/tcp    Unassigned
#                8/udp    Unassigned
discard          9/tcp    Discard
discard          9/udp    Discard
#                         Jon Postel <postel@isi.edu>
#               10/tcp    Unassigned
#               10/udp    Unassigned
systat          11/tcp    Active Users
systat          11/udp    Active Users
#                         Jon Postel <postel@isi.edu>
#               12/tcp    Unassigned
#               12/udp    Unassigned
daytime         13/tcp    Daytime (RFC 867)
daytime         13/udp    Daytime (RFC 867)
#                         Jon Postel <postel@isi.edu>
#               14/tcp    Unassigned
#               14/udp    Unassigned
#               15/tcp    Unassigned [was netstat]
#               15/udp    Unassigned
#               16/tcp    Unassigned
#               16/udp    Unassigned
qotd            17/tcp    Quote of the Day
qotd            17/udp    Quote of the Day
#                         Jon Postel <postel@isi.edu>
msp             18/tcp    Message Send Protocol
msp             18/udp    Message Send Protocol
#                         Rina Nethaniel <---none--->
chargen         19/tcp    Character Generator
chargen         19/udp    Character Generator
ftp-data        20/tcp    File Transfer [Default Data]
ftp-data        20/udp    File Transfer [Default Data]
ftp             21/tcp    File Transfer [Control]
ftp             21/udp    File Transfer [Control]
#                         Jon Postel <postel@isi.edu>
ssh             22/tcp    SSH Remote Login Protocol
ssh             22/udp    SSH Remote Login Protocol
#                         Tatu Ylonen <ylo@cs.hut.fi>
telnet          23/tcp    Telnet
telnet          23/udp    Telnet
#                         Jon Postel <postel@isi.edu>
                24/tcp    any private mail system
                24/udp    any private mail system
#                         Rick Adams <rick@UUNET.UU.NET>
smtp            25/tcp    Simple Mail Transfer
smtp            25/udp    Simple Mail Transfer
ş.a.m.d...
```

Fiecare circuit virtual de comunicaţie între (şi în afara a) două nivele TCP, sunt identificate unic prin două numere de port, care împreună se numesc *socket* (soclu, mufă). Acesta este compus din adresa **IP** a terminalului şi din numărul portului, utilizat de **TCP**. Fiecare dintre cele două componente care comunică au astfel de socket-uri.

Deoarece adresa IP este unică într-o reţea, iar numărul de port este, de asemenea, unic pentru un anumit terminal, numărul caracteristic al unui *socket*, este de asemenea, unic. Această identificare unică permite proceselor să comunice, unele cu altele, de-a lungul unei reţele şi că prin aceasta un protocol poate fi folosit în mai multe conexiuni, în acelaşi timp. Acest lucru este echivalent cu procesele de multiplexare în timp a semnalelor. Trebuie remarcat faptul că numărul portului (conexiunii) şi numărul de identificare a protocolului, sunt lucruri distincte.

În timpul procesului de iniţiere a comunicaţiei, componenta **TCP** de pe terminalul sursă face o cerere componentei **TCP** de pe terminalul destinaţie, folosind pentru socket un identificator unic, aşa cum se poate vedea în **figura 11.2**.

Dacă, de exemplu, aplicaţia client de pe terminalul sursă doreşte să stabilească o sesiune de Telnet cu aplicaţia server de pe terminalul de la distanţă, va iniţia conexiunea folosind ca număr de port sursă prima valoare nealocată peste 1024, de exemplu 1027 şi va trimite cererea la portul

destinație 23 (valoare utilizată de serverul de Telnet ca port de contact). Serverul de Telnet va răspunde cererii, prin trimiterea răspunsului de pe portul sursă 23 spre portul destinație, al aplicației client, 1027.

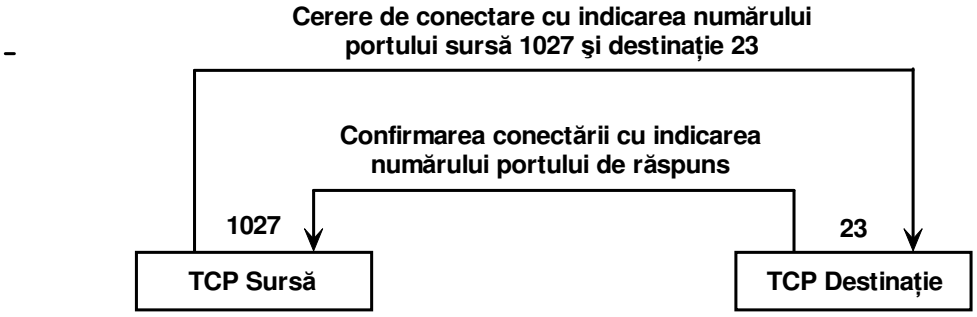

Fig. 11.2. Stabilirea unui circuit virtual cu ajutorul numerelor de port.

Componentele **TCP** păstrează o tabelă în care sunt trecute toate numerele corespunzătoare porturilor active. Terminalele, implicate într-o conexiune, au perechile de numere inversate între ele, ceea ce reprezintă "legătura" *(binding)* dintre cele două, aşa cum se poate vedea în **figura 11.3.**

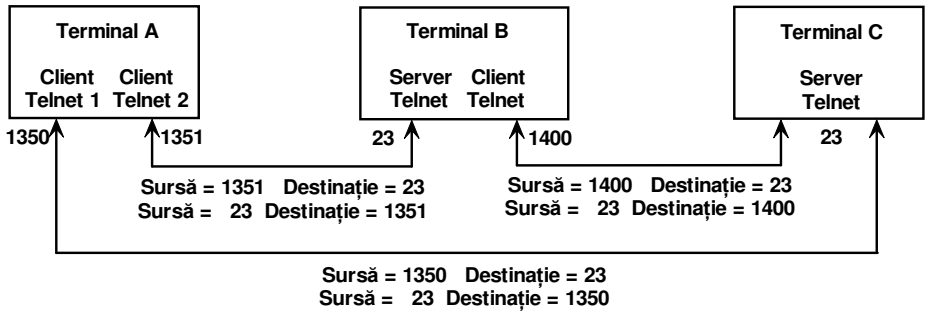

Fig. 11.3. Modul în care **TCP** ține evidența conexiunilor prin numere de porturi.

Dacă terminalul sursă face mai multe cereri, va folosi numere de porturi diferite, chiar dacă portul destinație este acelaşi. De exemplu, dacă terminalul sursă doreşte să stabilească, cu acelaşi server, trei sesiuni ale aplicației **Telnet**, va folosi pentru sursă numerele de porturi 1350 şi 1351 (dacă acestea nu sunt asociate cu nici un serviciu pe acel terminal, vezi fişierul *services*), iar pentru destinație 23. Toate sesiunile de Telnet vor trimite cererea spre acelaşi port de contact, 23. Dacă un terminal client doreşte să stabilească sesiuni de Telnet cu mai multe aplicații server aflate pe terminale diferite, poate folosi pentru portul destinație al fiecăreia acelaşi număr de port 23, deoarece adresa **IP** a destinațiilor este diferită. De asemenea, dacă mai multe terminale client cer sesiuni de **Telnet** aceleiaşi aplicații server, aceasta va putea răspunde pe acelaşi număr de port destinație, pe baza aceluiaşi proces de multiplexare. Datorită faptului că adresele IP sunt diferite, nu există nici un pericol ca datagramele să greşească destinația.

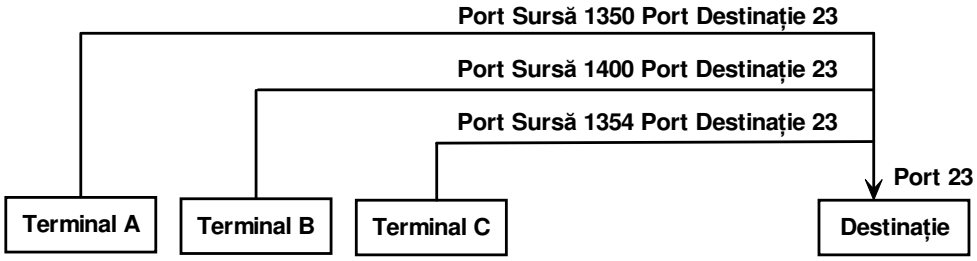

Fig. 11.4. Multiplexarea unui port destinație.

11.2.4. Comunicaţia dintre TCP şi nivelul superior.

Datagramele ajunse la nivelul transport (OSI 4) nu se opresc aici. **TCP** trebuie să comunice atât cu nivelul superior, cât şi cu cel inferior lui.

Pentru a putea comunica cu nivelul superior (în cazul modelului **TCP/IP** - aplicaţie) s-a definit un sistem compus din mesaje. Acest lucru nu este necesar pentru comunicaţia cu nivelul inferior (în general **IP**, dar nu obligatoriu). **TCP** aşteaptă ca nivelul de sub el să definească metoda de comunicaţie, care este de obicei asincronă. Pentru nivelul superior (**ULP** - *Upper Layer Protocol*), metoda de comunicaţie presupune un set de cereri elementare de serviciu *(service request primitives)* care sunt prezentate în tabelul 11.1.

Tabelul 11.1. Cereri elementare de servicii ULP-TCP.

Comandă	***Parametri aşteptaţi***
Cereri elementare de servicii dinspre ULP spre TCP	
ABORT	Numele conexiunii locale (Local connection name)
ACTIVE-OPEN	Portul local, socketul destinaţie (remote socket)
	Opţional: ULP timeout, acţiunea după timeout, prioritate (precedence), securitate, opţiuni
ACTIVE-OPEN-WITH-DATA	Port sursă, socket destinaţie, data, lungimea datei (segment), atirbut "push", atribut "urgent"
	Opţional: ULP timeout, acţiunea după timeout, prioritate (precedence), securitate
ALLOCATE	Numele conexiunii locale (Local connection name), lungimea datelor (segment)
CLOSE	Numele conexiunii locale
FULL-PASSIVE-OPEN	Port sursă, socket destinaţie
	Opţional: ULP timeout, acţiunea după timeout, prioritate, securitate, opţiuni
RECEIVE	Numele conexiunii locale, adresa tamponului, numărul de octeţi, atributul "push", atributul "urgent"
SEND	Numele conexiunii locale, adresa tamponului, lungimea datelor (segmentului), atributul "push",atributul "urgent"
	Opţional: ULP timeout, acţiunea după timeout
STATUS	Numele conexiunii locale
UNSPECIFIED-PASSIVE-OPEN	Portul local
	Opţional: ULP timeout, acţiunea după timeout, prioritate, securitate, opţiuni
***Cereri elementare de servicii dinspre* TCP spre ULP**	
CLOSING	Numele conexiunii locale
DELIVER	Numele conexiunii locale, adresa tamponului, lungimea datelor (segmentului), atributul "urgent"
ERROR	Numele conexiunii locale, descrierea erorii
OPEN-FAILURE	Numele conexiunii locale
OPEN-ID	Numele conexiunii locale, socketul destinaţie, adresa destinaţie
OPEN-SUCCESS	Numele conexiunii locale
STATUS RESPONSE	Numele conexiunii locale, portul sursă, adresa sursă, socketul destinaţie, starea conexiunii, fereastra de recepţie *(receive window)*, fereastra de trimitere *(send window)*, timpul de aşteptare a confirmării de primire a segmentului *(amount waiting ACK)*, timpul de aşteptare a pachetului de confirmare *(amount waiting receipt)*, urgenţă, prioritate, securitate, timeout, acţiunea după timeout
TERMINATE	Numele conexiunii locale, descriere

11.2.5. Porturi active şi pasive.

TCP acceptă două metode pentru stabilirea a conexiunii: activă şi pasivă. O conexiune activă se stabileşte atunci când **TCP** emite o cerere pentru o conexiune, bazată pe o indicație a unui protocol de nivel superior care furnizează numărul socket-ului. O conexiune pasivă are loc atunci când protocolul de nivel superior obligă **TCP** să aştepte până când apare o cerere, pentru o conexiune, din partea unui sistem de la distanță (de obicei, o conexiune activă din cealaltă parte). Când **TCP** primeşte o cerere de conexiune, acesta alocă un număr de port, ceea ce permite deschiderea rapidă a unei conexiuni.

Există două metode elementare de deschidere a unei conexiuni. O conexiune pasivă se deschide pentru nivele normale de prioritate şi de securitate. O deschidere pasivă nespecificată deschide portul la orice cerere, aşa cum este cazul aplicațiilor server care aşteaptă cereri din partea unor clienți nedefiniți.

TCP are reguli stricte referitoare la utilizarea conexiunilor active sau pasive. De obicei o deschidere pasivă este realizată pe un terminal, iar o deschidere activă pe celălalt, folosindu-se informații specifice despre numărul de identificare a socket-ului, nivelul de prioritate şi de securitate.

Deşi, majoritatea conexiunilor **TCP** se stabilesc pe baza unei cereri active adresată unui port pasiv, este posibil să se deschidă o conexiune fără ca să existe un port pasiv care să aştepte. În acest caz, componenta **TCP** care trimite cererea pentru conexiune, include în aceasta numărul ambelor porturi (local şi destinație). Dacă **TCP** de la destinație (care răspunde), permite cererea bazată pe criteriile impuse de aplicație, conexiunea se poate deschide.

11.2.6. Temporizatoarele TCP

TCP utilizează câteva temporizatoare pentru a se asigura că, în timpul comunicației, nu apar întârzieri exagerate şi informația se transmite cum trebuie, de la un capăt la celălalt.

Temporizarea retransmisiei.

Timpul de retransmitere este un interval, scurs din momentul expedierii unei datagrame, după care în cazul absenței mesajului de confirmare, are loc retransmiterea acelei datagrame.

RTO (*Retransmission Timeout* - expirarea timpului de retransmitere) controlează expirarea valabilității unei datagrame, iar valoarea acestui parametru variază în funcție de tipul rețelei. Dacă timpul expiră, datagrama este retransmisă cu o valoare **RTO** mărită exponențial. Dacă este depăşită şi limita maximă, conexiunea se consideră ratată şi se generează un mesaj de eroare, care se trimite nivelului superior (aplicației).

Valoarea de timp aleasă pentru acest parametru se determină prin măsurarea timpului mediu scurs între transmiterea primelor datagrame şi recepția mesajului de confirmare, valoare care se numeşte **RTT** (*Round Trip Time* - timp de călătorie dus-întors).

Valoarea medie a acestui timp se determină cu ajutorul unei relații experimentale şi se numeşte **SRTT** (*Smoothed Round-Trip Time* - timp de călătorie dus-întors ajustat). Aceasta va fi apoi crescută, în funcție de întârzierile neprevăzute.

Timpul de pasivitate.

După închiderea unei conexiuni **TCP**, se poate întâmpla ca un segment "întârziată" pe undeva prin rețea, să încerce să acceseze portul închis. Timpul de "tăcere" *(Quiet Timer)* are rolul de a preveni ca portul închis mai înainte să fie redeschis şi să recepționeze această ultimă datagramă.

Acest timp se alege de două ori mai mare decât timpul de viață maxim al segmentelor din şir **TTL** (*Time To Live*, stabilit în antetul **IP**), asigurând astfel că toate segmentele întârziate, care continuă să ajungă la port, vor fi abandonate. Acesta are ca efect practic faptul că portul devine inaccesibil, un anumit timp (circa 30 secunde), după închiderea conexiunii. În acest interval de timp componenta **TCP** de pe terminalul destinație răspunde cu un mesaj de eroare aplicației care face cererea.

Timpul de perseverență.

Există situaţii foarte rare când fereastra de recepţie este aleasă zero (în mod deliberat sau dintr-o eroare), ceea ce face ca terminalul care urmează să transmită datagrame să înceteze transmisia, pentru un timp nedefinit de lung. Dacă mesajul de reluare a transmisiei nu mai ajunge, terminalul care trebuie să reia transmisia nu o va face, ceea ce poate conduce la o stare de "acățare" pentru o perioadă nedefinită de timp.

Temporizatorul de "perseverență" aşteaptă un anumit timp prestabilit, după care trimite, la un interval dat, segmente de câte un octet, pentru a se asigura că terminalul de la capătul celălalt mai este încă împiedicat, din diferite motive, să recepţioneze date. Dacă acesta din urmă continuă să fie aglomerat va trimite în continuare mesaje în care dimensiunea ferestrei, semnificând capacitatea ei de recepţie, este zero. Dacă însă capacitatea de recepţie se modifică, acest lucru va fi consemnat în antetul segmentului următor ca o valoare, a dimensiunii ferestrei, diferită de zero.

Această valoare va fi folosită în continuare pentru comunicaţie, până la o nouă cerere de modificare a dimensiunii ferestrei.

Timpul de menţinere în stare activă şi timpul de aşteptare.

Aceşti parametri au fost adăugaţi în specificaţiile **TCP** mai târziu pentru a putea controla menţinerea unui port deschis chiar în absenţa unor date care trebuie transmise. Astfel, temporizatorul de menţinere în stare activă *(Keep Alive Timer)* trimite periodic un segment fără date pentru a se asigura că terminalul de la celălalt capăt ţine conexiune activă. Dacă nu primeşte răspuns după intervalul indicat de timpul de aşteptare (Idle Timer), conexiunea se consideră că este închisă. Valoarea timpului de menţinere în activitate este stabilit de o aplicaţie şi are valori cuprinse între 5 şi 45 secunde, iar timpul de aşteptare are, în general, valoarea de 360 secunde.

TCP foloseşte algoritmi adaptivi pentru stabilirea întârzierilor. Temporizatoarele îşi modifică automat valoarea intervalului, în funcţie de media întârzierilor specifice unei anumite conexiuni.

Pentru conexiuni mai rapide, cu o capacitate de recepţie mare a destinaţiei, valoarea întârzierilor admise va fi mai mică, iar pentru conexiuni mai lente, aceasta va fi mai mare.

11.2.7. Blocurile de control a transmisiei şi fluxului

TCP trebuie să ţină evidenţa fiecărei conexiuni, care se face cu ajutorul blocurilor de control a transmisiei (**TCB** - *Transmission Control Blocks*). Acestea conţin informaţiile necesare cum ar fi, numărul socket-ului local şi de la celălalt capăt, adresa tamponului de transmisie şi de recepţie, valoarea indicatorilor de securitate şi de prioritate.

TCB controlează cozile de transmisie şi de recepţie, împreună cu numărul de ordine în coadă, a segmentului curent. Pentru aceasta se folosesc câteva variabile de indicare a stării recepţiei şi transmisiei şi pentru controlul fluxului, care sunt prezentate în tabelul 11.2.

Tabelul 11.2. Variabile TCP folosite la transmisie şi recepţie.

Numele Variabilei	***Descriere***
Variabile folosite la transmisie	
SND.UNA	SendUnacknowledged (trimite fără confirmare)
SND.NXT	Send Next (trimite următorul segment)
SND.WND	Send Window (trimite dimensiunea ferestrei)
SND.UP	Sequence number of last urgent set (numărul secvenţei la care s-a modificat bit de urgenţă)
SND.WL1	Sequence number for last window update (numărul secvenţei la care s-a modificat fereastra)
SND.WL2	Acknowledgment number for last window update (numărul de confirmare la care s-a modificat fereastra)
SND.PUSH	Sequence number of last pushed set (numărul secvenţei la care s-a modificat bitul *push*)
ISS	Iniţial send sequence number set (numărul secvenţei iniţiale)

Variabile folosite la recepție	
RCV.NXT	Sequence number of next received set (numărul secvenței la care s-a schimbat dimensiunea următoarei ferestre)
RCV.WND	Number of sets that can be received (numărul de secvențe care pot fi recepționate)
RCV.UP	Sequence number of last urgent data (numărul ultimei secvențe cu bit de urgență activ)
RCV.IRS	Initial receive sequence number (numărul inițial al secvenței de recepționat)

Cu ajutorul acestor variabile, **TCP** controlează fluxul informației între cele două socket-uri. Pentru a înțelege mai uşor modul în care se folosesc aceste variabile, se va urmări un exemplu. Totul începe când terminalul A doreşte să trimită cinci blocuri terminalului B.

Dacă dimensiunea ferestre este limitată la şapte blocuri, atunci se pot trimite maxim şapte blocuri, fără să fie nevoie de confirmare de primire. Variabila SND.UNA de pe terminalul A indică numărul de blocuri trimise, până la acel moment, fără confirmare (în acest exemplu, 5), iar variabila SND.NXT are valoarea numărului de ordine al următorului bloc din acea secvență (în acest exemplu, 6).

Valoarea variabilei SND.WND este 2 (şapte blocuri posibile minus cele cinci trimise), deci doar două blocuri mai pot fi trimise până la depăşirea dimensiunii ferestrei. Terminalul B întoarce un mesaj care conține numărul de blocuri primite şi dimensiunea ferestrei pe care o poate recepționa fără probleme.

Transferul mesajelor, într-o direcție sau alta, poate să devină mai complex. Să analizăm exemplul următor. Terminalul sursă trimite blocuri fără cerere de confirmare, până la limita dimensiunii ferestrei, dar aşteaptă simultan mesaje de confirmare pentru segmente trimise mai înainte, dar care au fost deja eliminate din coada de aşteptare. Apoi trimite blocuri suplimentare pentru a umple fereastra, până la dimensiunea cerută.

Într-o astfel de situație, evidența blocurilor, deşi nu este foarte complicată, se poate "pierde" în cazul unei dimensiuni mari a ferestrei şi a unui trafic ridicat, ceea ce face că unele blocuri să se rătăcească. Cu toate acestea, este uimitor faptul că, în cele mai variate situații, un astfel de mecanism funcționează bine!

11.2.8. Formatul segmentului TCP

Aşa cum s-a putut vedea în paragraful anterior, **TCP** trebuie să comunice cu nivelul său inferior, de cele mai multe ori **IP** şi cu aplicația de la nivelul superior, folosind cererile elementare de servicii (primitive **TCP-ULP**).

TCP trebuie să comunice, de asemenea, cu componentele **TCP** de pe celelalte terminale cu care comunică în rețea. Pentru aceasta se foloseşte de unitățile elementare de date (**PDU**), cere la acest nivel se numesc segmente.

Totodată, aplicația aflată la nivelul superior celui de transport, trebuie să transmită nivelului rețea (**OSI 3**), adresa sursă (adresa proprie a terminalului) şi adresa destinație a terminalului pe care se află aplicația căreia îi este destinat mesajul. Pentru aceasta, componenta de transport de pe terminalul sursă, adaugă un antet provizoriu (pseudo-antet), care este interpretat de către nivelul rețea (OSI 3) după care este eliminat, în câmpul de date al pachetului fiind păstrat doar segmentul propriu-zis, specific protocolului de transport.

Antetul unui astfel de segment se poate vedea în **figura 11.5**.

Pseudo - antet

Adresă sursă (32 biți)		
Adresă destinație (32 biți)		
Rezervat (8 biți)	Protocol (8 biți)	Lungime TCP (16 biți)

Segment TCP

Port sursă (16 biți)								Port destinație (16 biți)
Numărul secvenței curente (32 biti)								
Numărul secvenței de primit (32 biți)								
Decalaj date (4 biți)	Rezervat (6 biți)	URG	ACK	PSH	RST	SYN	FIN	Fereastră de confirmare (16 biți)
Suma de control (16 biți)								Indicator de urgență (16 biți)
Opțiuni speciale								Completare cu 0
Date (variabil) + completare cu 0 până la multiplu de 32 biți								

Fig. 11.5. Structura segmentului **TCP**.

Semnificaţia câmpurilor din pseudo-antet este următoarea:

• **Adresă sursă**, (32 biţi) adresa proprie a terminalului de pe care se trimite mesajul,

• **Adresa destinaţie**, (32 biţi) adresa terminalului destinaţie pe care se află aplicaţia spre care trebuie trimis mesajul,

• **Rezervat**, (8 biţi) câmp rezervat pentru dezvoltări ulterioare. Nefolosit în prezent, acesta trebuie umplut cu 0.

• **Protocol**, (8 biţi) codul protocolului de transport, cu semnificaţie identică cu cel din antetul pachetului IP în care va fi încapsulat segmentul. Pentru TCP valoarea înscrisă este 6.

• **Lungime TCP**, (16 biţi) defineşte lungimea segmentului curent, care va fi încapsulat în pachet.

Semnificaţia câmpurilor, din antetul **TCP** propriu-zis, este următoarea:

• **Portul sursă** este un câmp de 16 biţi prin care se identifică portul TCP (serviciul de la nivelul superior) de pe terminalul local.

• **Portul destinaţie** este un câmp de 16 biţi care identifică portul TCP (serviciul de la nivelul superior) de pe terminalul de la distanţă,

• **Numărul secvenţei curente** este un câmp de 32 biţi care indică numărul de ordine în mesaj al blocului curent. Acelaşi număr este folosit între două componente TCP, pentru a defini, la trimitere, secvenţa iniţială *(ISS - Initial Send Sequence)*.

• **Numărul secvenţei de primit** *(Acknowledgment - confirmare)* este un câmp de 32 biţi care indică numărul secvenţei care urmează să fie transmisă de către celălalt terminal (primită de terminalul local). De obicei, dacă secvenţe trecută a fost recepţionată cu bine şi în întregime, acest număr este egal numărul ultimei secvenţe plus 1.

• **Decalajul** (Data offset) este un câmp de 4 biţi, prin care TCP identifică punctul de pornire al câmpului de date, în cuvinte de 32 biţi.

• **Rezervat** este un câmp de 6 biţi fără semnificaţie actuală, poate fi folosit într-o dezvoltare viitoare. Cu toate astea cei şase biţi trebuie să fie 0.

• **Atributul de urgenţă** (Urg flag) un bit a cărui valoare 1 activează câmpul indicatorului de urgenţă.

• **Atributul de confirmare** (Ack flag), dacă este 1 atunci câmpul cu numărul de confirmare este semnificativ.

• **Atributul Psh** (Push flag) un bit a cărui valoare egală cu 1 indică obligativitatea componentei TCP de a transmite imediat nivelului superior, segmentele recepționate până la acel moment, fără a mai aştepta sosirea celorlalte (în cazul în care mai urmează altele).

• **Atributul Rst** (Reset) un bit a cărui valoare 1 indică o conexiune care trebuie să fie resetată.

• **Atributul Syn** (Syncrinise) un bit folosit la indicarea unei conexiuni care trebuie sincronizate. Acest atribut se foloseşte în cursul stabiliri conexiunii.

• **Atributul Fin** (Final) este un bit care indică faptul că terminalul sursă nu mai are date de transmis. Acesta este echivalent cu marcajul de sfârşit de transmisie.

• **Fereastra de confirmare** (Window) este un câmp de 16 biți care defineşte numărul de blocuri de date pe care terminalul le poate recepționa între două confirmări succesive.

• **Suma de control** (Checksum) este un câmp de 16 biți a cărui valoare se calculează cu ajutorul complementului față de 1 a sumei complementelor față de 1 a biților din cuvintele de 16 biți din antet, pseudo-antet şi conținut. Aceasta presupune un proces destul de lung pentru a introduce şi suma de control în antet!

După eliminarea pseudo-antetului, se va recalcula suma de control a segmentului rămas care va fi înscrisă în câmpul corespunzător din antetul acestuia.

• **Indicatorul de urgență** *(Urgent Pointer)* este un câmp de 16 biți (se foloseşte împreună cu atributul de urgență egal cu 1) care indică porțiunea urgentă dintr-un mesaj, prin definirea decalajului față de numărul secvenței, aflat în antet. TCP nu defineşte măsura care trebuie luată în caz de urgență, aceasta rămâne la latitudinea aplicației.

• **Opțiuni** este un câmp cu dimensiune variabilă, asemănător cu cel din antetul IP, utilizat la specificarea unor opțiuni **TCP**. Pentru fiecare opțiune se folosesc două câmpuri. Primul, de un octet, defineşte mărimea câmpului care urmează imediat după acesta şi al doilea care defineşte opțiunea propriu-zisă.

Pentru **TCP** se definesc doar trei opțiuni şi anume:
0 Sfârşitul listei de opțiuni
1 Fără acțiune (No operation)
2 Dimensiunea maximă a segmentelor (Maximum segment size)

• **Completare cu zerouri** (Padding) un câmp de valoare nulă, folosit pentru a asigura că dimensiunea antetului este un număr multiplu de 32.

După antetul **TCP** urmează câmpul de date, care are dimensiune variabilă, ceea ce poate duce la situația posibilă în care terminalul sursă poate construi un segment mai lung decât poate să recepționeze terminalul destinație.

Suma de control se calculează pentru toată lungimea segmentului, inclusiv pentru cei 96 biți din "pseudo-antet". Acesta este adăugat înaintea antetului **TCP** şi conține informații necesare la întocmirea antetului IP, adresa sursă, adresa destinație, identificatorul protocolului de transport şi lungimea segmentului. Aceşti parametri sunt trimişi, în faza de transmisie, nivelului inferior (**IP**) şi sunt interpretați de acesta fără să fie însă încapsulați în pachetul de rețea.

11.2.9. Conexiunile TCP.

TCP are definite reguli stricte referitoare la comunicațiile sale. Aceste reguli, împreună cu procesele care le urmează pentru a stabili o conexiune, transmite date şi încheia conexiunea, sunt prezentate, de obicei, sub forma diagramelor de stare.

Datorită complexității proceselor de transmisie a datelor, aceste diagrame sunt, în general, complicate. Cu toate astea, sunt cea mai bună metodă de înțelegere a modului de funcționare a acestui protocol.

Stabilirea unei conexiuni.

O conexiune, între două terminale, se poate stabili doar dacă sunt întrunite câteva condiţii şi anume: nu există încă o conexiune între cele două socket-uri, ambele acceptă conexiunea şi dacă au implementate resurse TCP potrivite cu serviciile cerute.

O conexiune poate fi declanşată de o aplicaţie sau de un proces de sistem. Când se stabileşte o conexiune, acesteia i se atribuie unele caracteristici care rămân valabile până la închiderea conexiunii. De obicei, acestea sunt, prioritatea şi nivelul de siguranţă care sunt puse de acord, de către aplicaţiile de pe cele două terminale, în timpul procesului de stabilire a conexiunii.

În cele mai multe cazuri, o conexiune presupune două aplicaţii, între care apare o cerere de deschidere a unei conexiuni active, sau pasive. În **figura 11.6**. se prezintă procesul de deschidere a unei conexiuni **TCP**. Procesul începe când **TCP** de pe terminalul A primeşte o cerere de la protocolul de nivel superior (ULP), prin care se trimite, terminalului B, o cerere elementară de serviciu.

Segmentul, pregătit pentru trimitere, are atributul SYN activ (egal cu 1) şi un număr de ordine a secvenţei, în acest caz **ISS** *(Initial Sequece Number)* egal cu 50 (**SYN SEQ=50**).

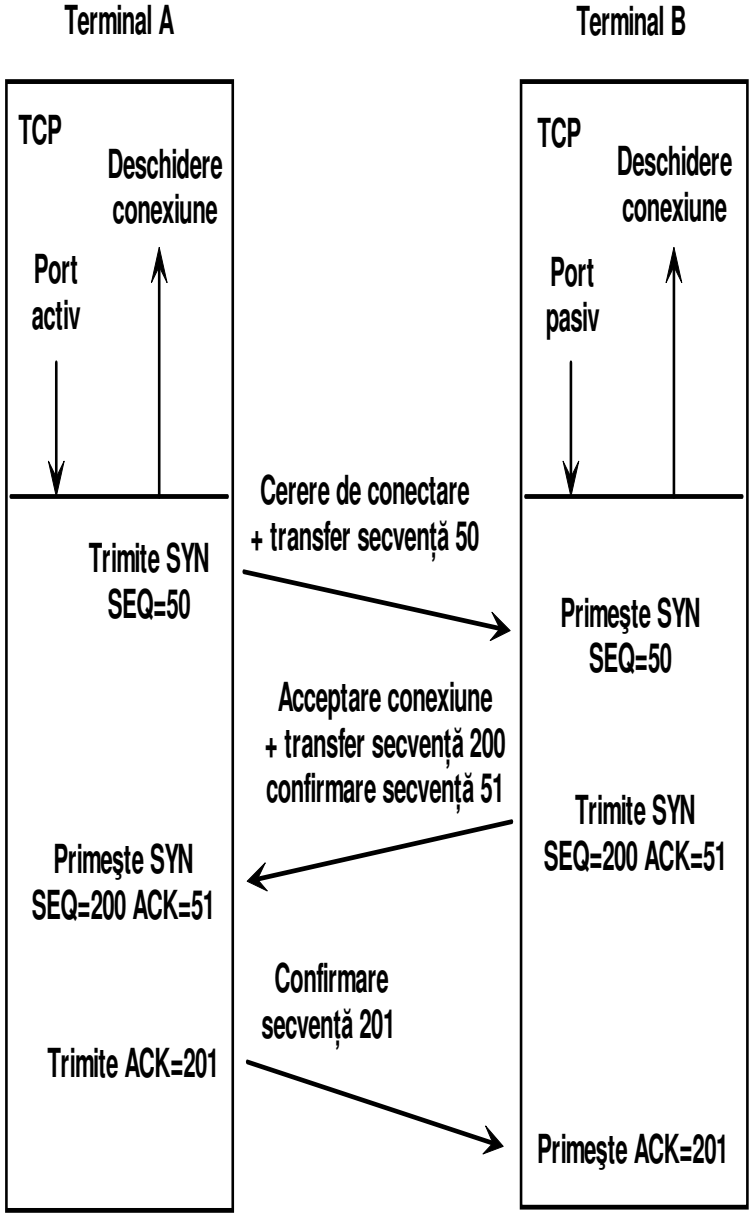

Fig. 11.6. Stabilirea (deschiderea) unei conexiuni.

Componenta **TCP**, de pe terminalul B, a fost instruită deja, cum să deschidă o conexiune pasivă, de către o aplicaţie corespunzătoare aflată pe acelaşi terminal. Când segmentul care conţine **SYN SEQ=50** ajunge la terminalul B, componenta sa **TCP** trimite înapoi spre A confirmarea de primire a segmentului.

În mesajul de confirmare este inclus numărul secvenţei următoare egal cu 51, alături de propriul număr iniţial de secvenţă *(ISS)*, în acest caz 200 şi atributul **SYN** activat (**ACK=51 SYN SEQ=200**). În cazul în care terminalul A primeşte acest mesaj, va răspunde cu un mesaj de confirmare care conţine numărul de secvenţă 201 (**ACK=201**), ceea ce înseamnă că aşteptă, de la B, secvenţa cu numărul 201. Prin acest schimb de mesaje (*Three Way Handshacking* - "trei strângeri de mână") conexiunea s-a stabilit, fiecare componentă **TCP** trimite mesaje protocolului de nivel superior, prin care anunţă acest lucru.

Nu este însă necesar ca terminalul de la distanţă să aibă o instrucţiune de deschidere pasivă. În acest caz, terminalul sursă furnizează ambele numere de *socket*, atât pe cel sursă, cât şi pe cel destinaţie, temporizările, nivelul de prioritate şi de siguranţă. Se întâlneşte destul de des situaţia în care ambele aplicaţii să ceară, în acelaşi timp, o conexiune activă. Aceasta se rezolvă în acelaşi mod, atâta doar că presupune un trafic suplimentar în reţea.

Transferul datelor.

După stabilirea conexiunii, transferul datelor se face direct, în ambele sensuri, aşa cum se poate vedea în **figura 11.7.** Pe fiecare bloc de date primit de la protocolul de nivel superior (**ULP**), componenta **TCP** de pe terminalul A îl încapsulează şi trimite terminalului B cu o creştere a numărului secvenţei.

După ce terminalul B recepţionează mesajul, incrementează numărul secvenţei şi trimite confirmarea, ocazie cu care anunţă terminalul A că a primit corect toate segmentele până la acel număr de secvenţă.

Pentru creşterea eficienţei transferului, nu se confirmă fiecare secvenţă în parte, ci doar un set, de mai multe secvenţe, număr definit de fereastra de confirmare *(window)*. Pentru aceasta, se vor trimite mai multe secvenţe, una după cealaltă, iar după recepţionarea lor se va trimite numărul de confirmare (ACK) egal cu numărul de ordine al primei secvenţe din setul următor, aşa cum se poate vedea în **figura 11.8.**

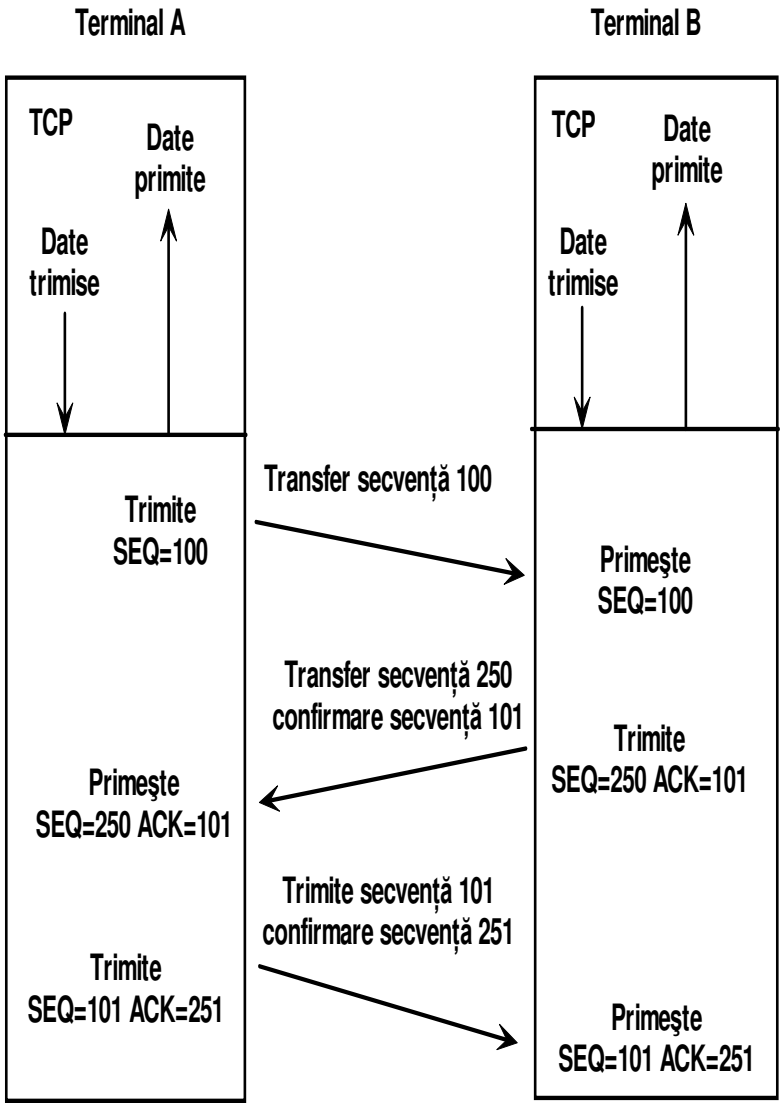

Fig. 11.7. Transferul datelor după deschiderea conexiunii.

Fereastra de confirmare se transmite de fiecare terminal în parte şi depinde de capacitatea fiecărui sistem de operare de a gestiona componenta TCP. Când aceasta primeşte un segment, cu WIN=50 şi ACK=101, va transmite 50 de secvenţe începând cu secvenţa 101, fără să mai aştepte o altă confirmare. După transmiterea celei de-a 150-a secvenţe, va aştepta confirmarea trimiterii setului următor, care începe cu secvenţa 151 (ACK=151), după care procesul se repetă.

În cazul în care se pierd secvenţe pe drum, de exemplu nu s-au primit consecutiv decât secvenţele cuprinse între 151 şi 181, celelalte sosind cu discontinuităţi, cererea se va face prin trimiterea confirmării începând cu numărul de ordine al primei secvenţe pierdute (ACK=182). După primirea acestei confirmări, se mai încerca transmiterea unui set de secvenţe egal cu fereastra de confirmare (WIN=50), chiar dacă o parte dintre ele au sosit deja. Acest lucru, deşi pare să încarce inutil fluxul de date, simplifică mult sarcina de ordonare şi reasamblare a secvenţelor.

În cazul pierderii pe drum a segmentelor, sau al primirii lor incomplete sau cu erori, acestea se vor retransmite prin confirmarea cererii primei secvenţe pierdute şi cu o dimensiune de fereastră egală cu ceai iniţială.

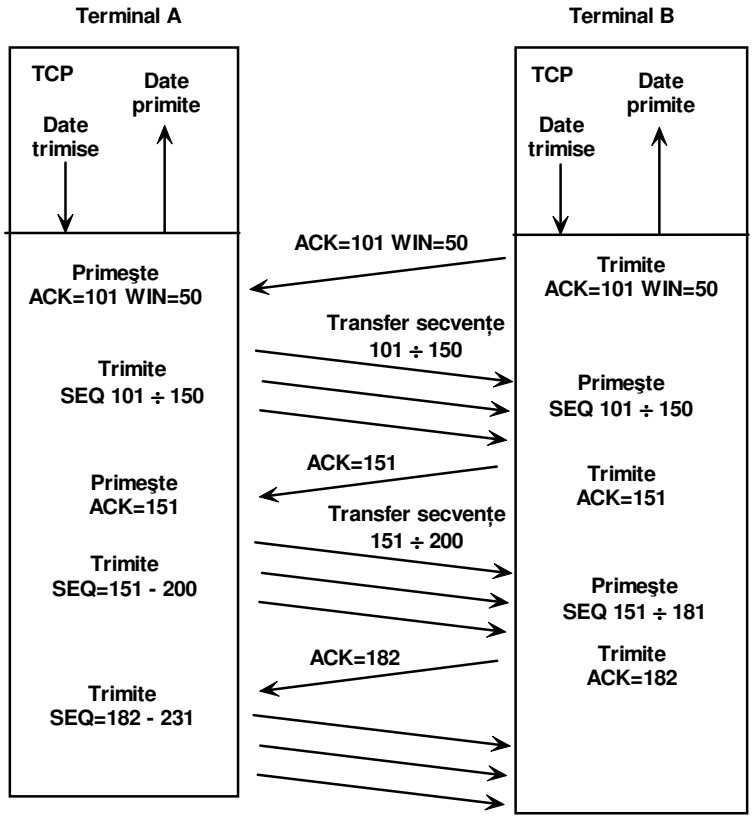

Fig. 11.8. Transferul unui set de secvenţe, folosind fereastra de confirmare multiplă.

În cazul în care fenomenul de pierdere a segmentelor se repetă, fereastra de confirmare va fi redusă. În cazul dispariţiei fenomenului de pierdere, sau degradare, a segmentelor, fereastra poate fi crescută. Modificarea dimensiunii ferestrei de confirmare se face cu variaţie în progresie geometrică. Modificarea dinamică a dimensiunii ferestrei permite controlul fluxului de date, dintre sursă şi destinaţie.

Prin modificarea numărului de secvenţe transferate în timp, la o valoare optimă, se stabileşte un echilibru între pierderi şi trafic inutil.

Închiderea unei conexiuni.

Pentru a închide o conexiune, una dintre componentele **TCP** primeşte o cerere elementară de la protocolul de nivel superior şi emite un mesaj în care este activat atributul FIN.

În **figura 11.9**, terminalul A trimite terminalului B o cerere de închidere în care este inclus numărul secvenţei următoare (ACK=475).

Terminalul B confirmă primirea cererii şi trimite secvenţa sa curentă şi numărul următoarei secvenţe aşteptate (ACK=351). După aceasta, terminalul B trimite mesajul de închidere spre nivelul următor celui de transport (aplicaţie) şi aşteaptă ca aplicaţia să confirme închiderea. Acest pas nu este absolut necesar, **TCP** poate închide conexiunea fără ca aplicaţia să accepte acest lucru, dar de obicei aceasta este informată de orice schimbare a stării conexiunii.

Închiderea conexiunii va fi acceptată de componenta **TCP** de pe terminalul B doar după transmiterea şi confirmarea tuturor secvenţelor. Odată cu transmiterea ultimei secvenţe (SEQ=475) şi confirmarea sosirii acesteia (ACK=476), terminalul B va trimite bitul FIN, confirmând astfel închiderea conexiunii.

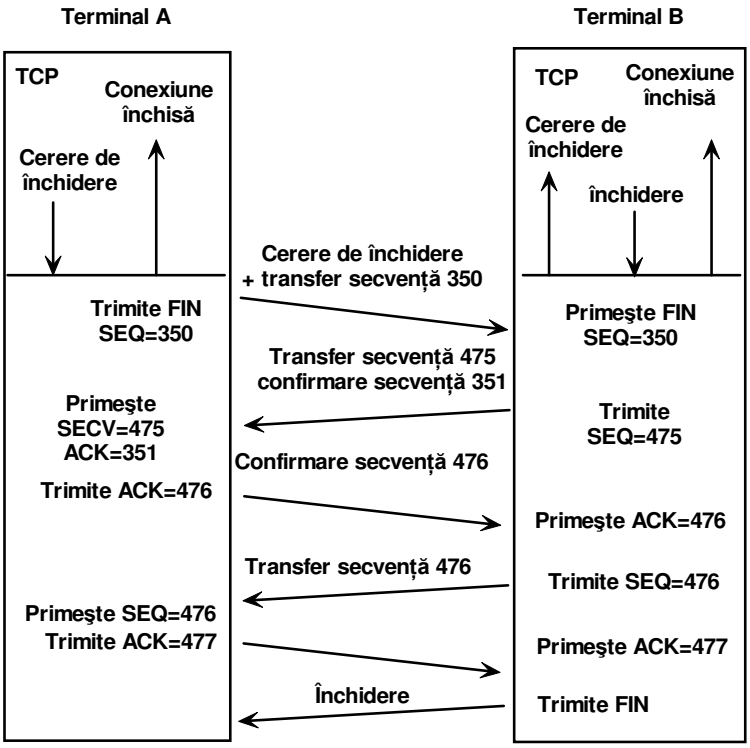

Fig. 11.9. Închiderea unei conexiuni.

După primirea, de la aplicație, a confirmării de închidere a conexiunii (sau după expirarea timpului de aşteptare a acesteia), componenta **TCP** de pe terminalul B trimite spre A un segment în care este activat atributul **FIN**. Terminalul A confirmă închiderea şi conexiunea este terminată.

O închidere bruscă a unei conexiuni poate să apară atunci când una dintre părți închide socket-ul. Aceasta se poate întâmpla fără să se trimită nici un mesaj, în acest sens, celuilalt terminal. Astfel de întreruperi pot să apară atunci când terminalul se opreşte datorită unei pene de tensiune, oprirea unui proces, sau a unei aplicații, de către utilizator.

Conexiunea poate fi întreruptă şi de către o componentă de monitorizare, în cazul în care se decide administrativ că la un moment dat conexiunea trebuie întreruptă. Celălalt capăt al conexiunii poate să nu-şi dea seama de acest lucru decât după expirarea timpului de aşteptare a confirmării de primire.

Pentru a ține evidența tuturor conexiunilor, **TCP** foloseşte o tabelă în care se găseşte câte o intrare, pentru fiecare conexiune, însoțită de informațiile specifice corespunzătoare.

Conținutul acestui tabel se poate afişa cu ajutorul comenzii `netstat -an`, aşa cum se poate vedea în cele ce urmează:

```
root@masserv:~# netstat -an
Active Internet connections (servers and established)
Proto Rec-Q Snd-Q Local Address      Foreign Address     State
tcp        0      0 0.0.0.0:3306       0.0.0.0:*           LISTEN
tcp        0      0 0.0.0.0:80         0.0.0.0:*           LISTEN
tcp        0      0 193.226.5.178:80   79.113.140.88:37552 SYN_RECV
tcp        0      0 193.226.5.178:80   79.113.140.88:13239 SYN_RECV
tcp        0      0 0.0.0.0:21         0.0.0.0:*           LISTEN
tcp        0      0 193.226.5.178:21   193.226.5.35:39851  ESTABLISHED
tcp        0      0 0.0.0.0:22         0.0.0.0:*           LISTEN
tcp        0      0 193.226.5.178:1578 193.226.5.33:21     ESTABLISHED
tcp        0      0 193.226.5.178:22   193.226.5.35:49848  ESTABLISHED
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
unix  2      [ ACC ]     STREAM     LISTENING     659    /var/run/mysql/mysql.sock
unix  4      [ ]         DGRAM                    114    /dev/log
unix  2      [ ACC ]     STREAM     LISTENING     677    /dev/gpmctl
unix  2      [ ]         DGRAM                    40     @udevd
unix  2      [ ]         DGRAM                    201920
unix  2      [ ]         DGRAM                    117
root@masserv:~#
```

Semnificația fiecărei coloane este următoarea:

• **Protocol,** protocolul de transport folosit,

• **Recv-Q**, numărul de octeți necopiați din coada de recepție, de către aplicația de la nivelul superior,

• **Send-Q**, numărul de octeți neconfirmați de terminalul de la distanță,

• **Adresa locală**, adresa IP a terminalului local, care în stare de aşteptare are valoarea 0.0.0.0.

• **Portul local,** numărul portului local,

• **Adresa de la distanță** Adresa IP a terminalului de la distanță,

• **Portul de la distanță** numărul portului de pe terminalul de la distanță,

• **Starea conexiunii** (închisă, deschisă, în aşteptare, închidere etc),

11.3. UDP (User Datagram Protocol).

11.3.1. Prezentare.

S-a văzut mai înainte că **TCP** este un protocol bazat pe conexiune. Sunt şi unele situații în care este nevoie de un protocol care să nu efectueze toți paşii de inițiere, menținere şi închidere a procedurii de transport, adică un protocol fără conexiune.

Un astfel de protocol poate fi mai eficient şi mai rapid, dar poate fi folosit doar în cazul unor rețele la care pericolul degradării datelor este mic, sau pentru transferul informațiilor fără mare importanță, a căror pierdere nu este un lucru foarte grav.

Protocoalele care nu folosesc mecanisme de retransmitere a segmentelor pierdute, sau de control al fluxului, nu oferă nici o garanție asupra reuşitei unei transmisii. Rămâne în sarcina protocolului de nivel superior să verifice integritatea segmentelor de date şi să ceară, atunci când este cazul, retransmiterea lor. **UDP** (User Datagram Protocol) este un astfel de protocol şi se foloseşte, la transmiterea de datagrame, alături de **TCP** [1].

Două dintre cele mai des întâlnite aplicații care folosesc **UDP** sunt **TFTP** (Trivial File Transfer Protocol) şi **RPC** (Remote Procedure Call). **TFTP** se foloseşte pentru a încărca sisteme de operare sau fişiere de configurare, de pe un terminal server pe una client, sau invers, fără autentificarea utilizatorului. **RPC** sunt cereri înaintate de un client şi rulate de către un server, aflate pe acelaşi terminal, sau pe terminale diferite.

11.3.2. Formatul segmentului UDP.

UDP este mult mai simplu decât **TCP** şi operează doar ca agent de transport a datagramelor, dintr-o direcție în cealaltă.

Antetul segmentului **UDP** este şi el mult mai simplu decât cel **TCP** şi trebuie să aibă dimensiunea un număr multiplu de 32 biți, aşa cum se poate vedea în **figura 11.10.**

Ca şi la **TCP**, dinspre nivelul superior trebuie să se transmită anumite informații (adresa de rețea, sursă şi destinație) nivelului rețea (OSI 3) pentru care, la începutul fiecărui segment **UDP** trebuie adăugat un antet provizoriu (pseudo-antet). Acesta se va elimina după ce nivelul rețea va extrage din el informațiile necesare. Semnificația câmpurilor din pseudo-antet este aceeaşi ca şi în cazul segmentului **TCP**, cu diferența că valorile trecute în câmpul **Protocol** se referă la **UDP**, adică valoarea 17.

Pseudo - antet

Adresă sursă (32 biți)		
Adresă destinație (32 biți)		
Rezervat (8 biți)	Protocol (8 biți)	Lungime UDP (16 biți)

Segment UDP

Port sursă (16 biți)	Port destinație (16 biți)
Lungime segment (16 biți)	Suma de control (16 biți)
Date (variabil) + completare cu 0 până la multiplu de 32 biți	

Fig. 11.10. Structura segmentului UDP.

Semnificaţia câmpurilor este următoarea:

- **Portul sursă** este un câmp opţional de 16 biţi, care defineşte numărul portului. În cazul în care acesta nu este specificat, valoarea sa este 0.
- **Portul destinaţie** este un câmp de 16 biţi care defineşte numărul portului (serviciului) de pe terminalul destinaţie
- **Lungimea** este un câmp de 16 biţi reprezentând dimensiunea datagramei, inclusiv antetul (date plus antet).
- **Suma de control** este un câmp de 16 biţi care reprezintă fiecare complementul faţă de unu a celor 16 biţi ai sumei biţilor datagramei, inclusiv antet. Similar cu suma de control de la **TCP**.

Suma de control este un câmp opţional, dar dacă nu este folosit (trebuie umplut cu zero) nu există nici o metodă de verificare a datelor transmise. Suma de control de la nivelul **IP** nu verifică decât integritatea antetului propriu.

Se observă că acest protocol nu are nici un fel de mecanism pentru controlul transmisiei secvenţelor pierdute, sau degradate pe drum, ceea ce înseamnă că nu este un protocol de transport, propriu-zis.

Datagramele pot să ajungă la destinaţie în altă ordine decât cea în care au pornit de la sursă, sau se pot pierde pe drum.

În aceste situaţii, aplicaţia este obligată să verifice şi eventual să ceară retransmiterea datelor eronate. Cu toate acestea, protocolul **UDP** se pretează pentru transmiterea mesajelor care nu sunt sensibile în cazul degradării (video şi audio conferinţe), sau a interogărilor transmise într-o singură datagramă. Pentru cazul din urmă, mesajele se transmit în reţea, confirmarea lor făcându-se prin răspunsul primit în urma interogării.

TESTE DE AUTOEVALUARE

1. Protocolul TCP oferă următoarele funcții:

   a) Segmentarea şi rutarea datelor, controlul erorilor, conexiune virtuală;
   b) Segmentarea datelor, controlul fluxului şi al erorilor, conexiune virtuală;
   c) Segmentarea datelor şi numerotarea secvențelor, controlul fluxului şi al erorilor, conexiune virtuală;
   d) Adresarea terminalelor, controlul fluxului şi al erorilor, conexiune virtuală;
   e) Segmentarea datelor, controlul fluxului şi al erorilor, alocarea automată a identificatorului de port, pentru aplicațiile server.

2. Deschiderea unei conexiuni TCP se realizează în următoarele etape:
   a) Aplicația server face o cerere pentru un port activ, componenta TCP sursă trimite bitul SYN=1 spre terminalul destinație, terminalul destinație trimite bitul SYN=1 confirmând acceptarea conexiunii, terminalul sursă confirmă secvența de acceptare trimițând bitul SYN=0;
   b) Aplicația server face o cerere pentru un port pasiv, componenta TCP sursă trimite bitul SYN=1 spre terminalul destinație, terminalul destinație trimite bitul SYN=1 confirmând acceptarea conexiunii, terminalul sursă confirmă secvența de acceptare trimițând bitul SYN=0;
   c) Aplicația client face o cerere pentru un port activ, componenta TCP sursă trimite bitul SYN=1 spre terminalul destinație pe care operează aplicație server. Terminalul destinație trimite bitul SYN=1 confirmând acceptarea conexiunii, terminalul sursă confirmă secvența de acceptare trimițând bitul SYN=0;
   d) Aplicația client face o cerere pentru un port activ, componenta TCP sursă trimite bitul SYN=1 spre terminalul destinație, terminalul destinație trimite bitul SYN=0 confirmând acceptarea conexiunii, terminalul sursă confirmă secvența de acceptare trimițând bitul SYN=1;
   e) Aplicația client face o cerere pentru un port activ, componenta TCP sursă trimite bitul SYN=1 spre terminalul destinație, terminalul destinație trimite bitul FIN=1 confirmând acceptarea conexiunii, terminalul sursă confirmă secvența de acceptare trimițând bitul SYN=0.

3. Protocolul UDP oferă următoarele funcții:
   a) Segmentarea, numerotarea secvențelor, confirmare de primire, conexiune virtuală;
   b) Segmentarea, numerotarea secvențelor, confirmare de primire, conexiune virtuală, identificarea aplicațiilor prin număr de port;
   c) Segmentarea datelor, numerotarea secvențelor, confirmare de primire, conexiune virtuală, identificarea aplicațiilor prin număr de port;
   d) Segmentarea datelor, identificarea aplicațiilor prin număr de port, verificarea integrității cadrelor;
   e) Segmentarea datelor, numerotarea secvențelor, conexiune virtuală, identificarea aplicațiilor prin număr de port, verificarea integrității cadrelor

I. Alegeți răspunsul corect:

1. Protocolul TCP asigură livrarea corectă la destinație a datelor, prin mecanismele de control al erorilor şi de confirmare de primire.

   □ adevărat □ fals

2. Perechea socket-sursă socket destinație identifică ruta pe care o parcurge segmentul cu date, între terminalul sursă şi cel destinație.
   □ adevărat □ fals

3. Protocolul UCP, datorită funcției de confirmare a secvențelor, asigură o viteză de transfer mai mare decât TCP, ceea ce îl face eficient în aplicațiile multimedia.
   □ adevărat □ fals

RĂSPUNSURI

I. Varianta de răspuns corectă:

1. b, c
2. d
3. d

II. Alegeți răspunsul corect:
1. adevărat
2. fals
3. fals

Bibliografie

1. **Parker, T**., *Teach Yourself TCP/IP in 14 Days - Second Edition*, Sams Publishing, 1996