6. Nivelul transport

6.1. Cuprins modul

Introducere

Nivelul transport este situat între nivelul aplicație şi nivelul rețea reprezentând un element central al unei arhitecturi de rețea stratificate. Prin transferul sigur şi eficace al datelor de la gazda sursă la gazda destinație nivelul transport furnizează direct servicii de comunicații proceselor aplicație care rulează pe diferite gazde, independent de rețeaua sau rețelele fizice utilizate. În continuare vom examina posibilele servicii oferite de nivelul transport şi principiile care stau la baza diverselor abordării pentru furnizarea acestor servicii. Se va acorda o atenție deosebita protocoalelor Internet reprezentate la nivel transport de TCP şi UDP

Obiective

După parcurgerea acestei unități de curs studenții vor fi capabili:

- ✓ Să descrie nivelul transport în Internet;
- ✓ Să analizeze relația dintre nivelul transport şi rețea;
- ✓ Să explice cum se efectuează multiplexarea şi de-multiplexarea;

Durată medie de studiu

Durata medie de studiu individual : 2 ore

individual

6.2. Principiile nivelului transport

Rolul şi serviciile furnizate de nivelul transport:

Protocolul nivelului transport oferă o comunicație logică între procesele aplicație ce rulează pe diferite gazde. Prin comunicație ”logică”, înțelegem că, deşi procesele aplicație ce comunică nu sunt conectate fizic între ele, din punctul de vedere al aplicației este ca şi cum acestea ar fi conectate fizic. Ele se pot afla în locații geografice diferite fiind conectate prin numeroase rutere şi legături. Procesele aplicație folosesc comunicarea logică oferită de nivelul transport pentru a-şi trimite mesaje reciproc, fără a tine cont de detaliile infrastructurii fizice folosite. Figura 6.1 ilustrează noțiunea de comunicare logică.

Aşa cum se arată în Figura 6.1, protocoalele nivelului transport nu sunt implementate în ruterele din rețea ci numai în terminale. Ruterele operează doar cu câmpurile antetelor datagramelor nivelului rețea, ele nu acționează în câmpurile antetelor segmentelor de nivel transport.

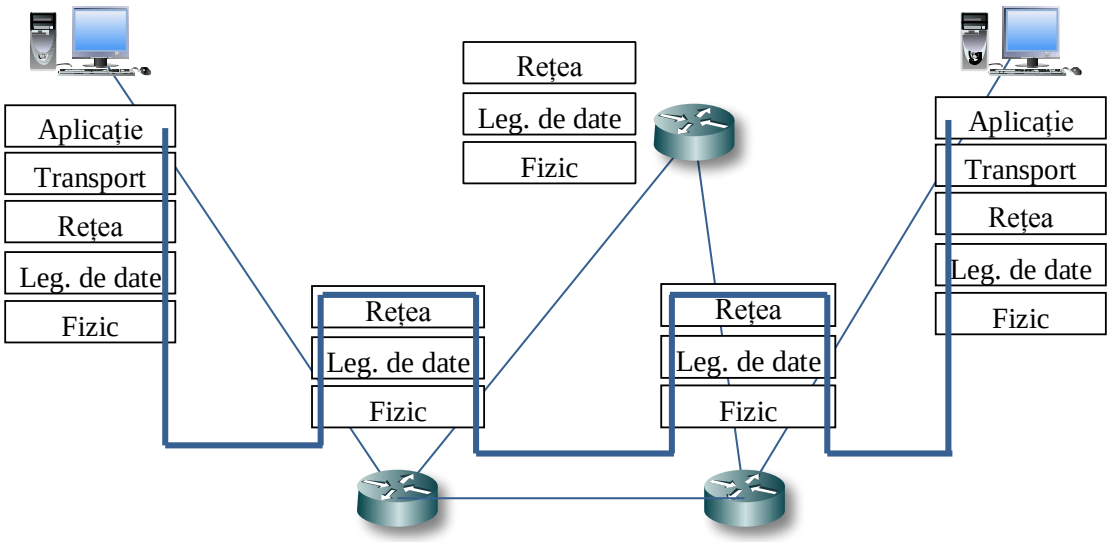

Figura 6.1 Comunicarea logică între procesele aplicație

La emisie, nivelul transport împarte în segmente mesajele pe care le primeşte de la un proces aplicație. Reamintim că unitățile de date ale protocolului nivelului transport sunt denumite segmente. Această împărțire se face prin divizarea (eventual) a mesajelor aplicației în părți mici şi adăugarea unui antet al nivelului transport la fiecare parte, creând astfel segmente. După ce le creează, nivelul transport transferă segmentele nivelului rețea, iar fiecare segment este încapsulat într-o datagramă. La recepție, nivelul transport primeşte segmentele de la nivelul rețea

şi le înlătură antetul transport de la segmente, reconstituie mesajul şi îl livrează procesului aplicație destinatar.

O rețea de calculatoare poate pune la dispoziția aplicațiilor de rețea mai mult de un protocol de nivel transport. De exemplu, Internet-ul (stiva TCP/IP) dispune de două protocoale de nivel transport– TCP şi UDP. Fiecare protocol furnizează un set diferit de servicii ale nivelului transport pentru aplicații. Toate protocoalele nivelului transport furnizează un serviciu de multiplexare/demultiplexare a aplicațiilor. În plus față de serviciul de multiplexare/demultiplexare, un protocol de transport poate oferi aplicațiilor şi alte servicii, printre care: transferul fiabil de date, controlul fluxului, garantarea lărgimii de bandă şi a întârzierilor.

6.3. Relația dintre Nivelurile Transport şi Rețea

Din perspectiva aplicațiilor de rețea, nivelul transport reprezintă o parte din infrastructura de bază a comunicării. De fapt, infrastructura de comunicație reprezintă mai mult decât nivelul transport. Întrucât un protocol al nivelului transport oferă comunicație logică între procesele care rulează pe diferite gazde, un protocol al nivelului rețea oferă comunicație logică între gazde. Această diferență este subtilă, dar importantă. Nivelul transport se află deasupra nivelului rețea în stiva de protocoale. Să examinăm această diferență cu ajutorul unui exemplu din viața cotidiană.

Planul analogiei	Relația dintre nivelul rețea şi nivelul transport
Imaginați-vă că suntem în anul 1940. Considerăm o firmă, cu două sedii, unul în Braşov (15 angajați) şi altul în Cluj (20 de angajați). Pentru a realiza producția, angajații din cele două sedii trebuie să corespondeze unii cu alții. Scrisorile de la Braşov sunt colectate zilnic de către unul din colegi, Andrei şi predate unui angajat al Poştei din oraş. După 2-3 zile (suntem în 1940!) scrisorile ajung la Cluj. Aici, un angajat al Poştei dă scrisorile lui Cornel pentru a le distribui colegilor săi. El colectează şi scrisorile de la colegii săi din Cluj şi le predă poştaşului pentru a fi expediate la Braşov. Cei doi angajați, Andrei şi Cornel, oferă	Exemplu poate fi considerat o analogie pentru a explica modul în care nivelul transport relaționează şi interacționează cu nivelul rețea: Gazdele (numite şi terminale) = sediile din Braşov şi Cluj Protocolul nivelului transport =Andrei şi Cornel

comunicație logică între angajații celor două sedii, iar poşta română oferă comunicație logică între cele două sedii: ea nu expediază scrisorile de la un angajat la altul ci de la un sediu la altul. Pentru angajații din cele două sedii, Andrei şi Cornel joacă rolul de „ajutori de postaşi" ca şi când ar fi o parte a serviciului „Poşta Română", capătul sistemului de distribuție a scrisorilor.	Procesele = angajații din firmă Mesajele aplicațiilor = scrisorile în plicuri Protocolul nivelului rețea = serviciul poştal
Dacă rămânem în planul analogiei, constatăm că Andrei şi Cornel se ocupă de colectarea şi distribuirea scrisorilor în sediul lor, ca şi când ar fi poştaşi. Ei nu sunt implicați în activitatea din poşta centrală unde se sortează scrisorile pentru toată țara şi nici în transportul acestora de la Braşov la Cluj sau invers.	Părăsind planul analogiei putem spune că protocoalele nivelului transport rulează pe sistemele capăt, asemenea lui Andrei şi Cornel. La capătul sistemului, un protocol de transport transferă mesajele de la procesele aplicație la marginea rețelei şi invers; acesta nu este însă implicat în modul în care mesajele sunt mutate în cadrul rețelei de bază. Într-adevăr după cum este ilustrat în Figura 5-1, ruter-ele intermediare nici nu acționează, nici nu recunosc, vreo informație pe care nivelul transport ar fi ataşat-o mesajelor.
Să presupunem că în luna august, Andrei şi Cornel pleacă în concediu, iar activitatea lor de poştaşi este preluată de alți doi angajați, cu mai puțină experiență. Aceştia nu-şi cunosc prea bine atribuțiile, uită să colecteze scrisorile zilnic, le pierd pe unele. Şeful fiecărui sediu constată că serviciul oferit in luna august este mai slab decât în restul anului.	Similar, putem spune că o rețea de calculatoare poate pune la dispoziție mai multe protocoale de transport, fiecare protocol oferind un alt model de servicii pentru aplicații.
Deşi Andrei şi Cornel îşi fac exemplar serviciul de „poştaşi" servicii oferite de ei sunt influențate de modul în care lucrează poşta	În mod similar, serviciile pe care un protocol transport le oferă sunt adesea constrânse de tipul de rețea care stă la baza nivelului rețea. Dacă protocolul nivelului rețea nu poate oferi

română. De exemplu, dacă trenul Braşov Cluj întârzie, cei doi nu pot duce şi distribui scrisorile colegilor lor astfel că aceştia nu pot comunica la timp cu angajații din celălalt sediu.	o întârziere sau lățime de bandă pentru segmentele trimise între gazde, atunci protocolul nivelului transport nu poate oferi întârzierea sau lățimea de bandă garantate pentru ca mesajele să se transmită între procese.

Cu toate acestea, anumite servicii pot fi oferite de protocolul de transport chiar şi atunci când nivelul rețea nu oferă serviciile corespunzătoare nivelului transport. De exemplu, un protocol de transport poate oferi servicii fiabile de transfer de date unei aplicații chiar şi atunci când protocolul de rețea nu este corespunzător, adică acesta pierde, reordonează sau multiplică pachete. Ca un alt exemplu un protocol transport poate utiliza criptare pentru a garanta că mesajele aplicației nu sunt înțelese de intruşi, chiar şi atunci când nivelul rețea nu poate garanta confidențialitatea segmentelor.

6.4. Prezentare generală a nivelului transport în Internet

Înainte de a analiza protocoalele UDP şi TCP, este util să spunem câteva cuvinte despre nivelul rețea din Internet. Protocolul nivelului rețea pentru Internet este denumit – IP (Internet Protocol), abrevierea în limba engleză de la ”Protocolul Internet”. Fiecare gazdă are o adresă IP care oferă comunicații logice cu alte gazde. Vom examina în detaliu adresarea IP în următoarele unitate de învățare; în acest capitol trebuie doar să avem în vedere faptul că fiecare gazdă are o adresă IP unică. Modelul serviciilor IP este unul de livrare „best effort” (cel mai bun efort). Aceasta înseamnă că IP depune toate eforturile de a livra segmente între gazde, dar nu aduce nici o garanție: nu garantează livrarea segmentelor, nu garantează livrarea segmentelor în ordine şi nu garantează integritatea datelor din segmente. Din aceste motive, IP nu este un serviciu fiabil

Internetul, şi în general o rețea TCP/IP, pune la dispoziție nivelului aplicație două protocoale de transport distincte: UDP (User Datagram Protocol) şi TCP (Transmission Control Protocol). Protocolul UDP oferă aplicațiilor un serviciu fără conexiune, cu fiabilitate scăzută. Celălalt protocol (TCP) oferă aplicațiilor un serviciu fiabil, orientat conexiune. Când se proiectează o aplicație de rețea, dezvoltatorul:

- trebuie să aleagă unul dintre aceste protocoale;

- poate proiecta aplicații care să aleagă dinamic folosirea TCP sau UDP, în funcție de serviciile momentane puse la dispoziție de rețea. In acest ultim caz, creşte

complexitatea şi efortului de implementare. Astfel de opțiuni se întâlnesc de exemplu la DNS şi aplicația de telefonie şi mesagerie instantă Skype.

După ce am aruncat o privire asupra modelului de servicii IP, să rezumăm modelul de servicii pentru UDP şi TCP. Rolurile UDP şi TCP sunt:

- să extindă serviciul de livrare IP între două terminale la un serviciu de livrare între două procese ce rulează pe terminale, extindere numită multiplexare şi demultiplexare. Vom discuta despre multiplexare şi demultiplexare în secțiunea următoare.

- să verifice integritatea segmentelor incluzând în antet câmpuri de detecție a erorilor.

Aceste servicii ale nivelului transport – livrare gazdă-gazdă şi verificarea erorilor – sunt singurele servicii pe care le oferă UDP! Ca şi IP-ul, UDP nu este un serviciu de încredere – nu garantează că datele trimise vor ajunge intacte la destinație.

Spre deosebire de protocolul UDP şi de IP, TCP oferă şi alte câteva *servicii adiționale*: oferă transfer de date fiabil şi realizează controlul congestiei. Folosind controlul fluxului, numere de secvența, confirmări şi cronometre TCP oferă garanția transferului de date. Cu alte cuvinte, datele sunt trimise de la procesul emițător la cel receptor corect şi în ordine transformând serviciul nefiabil dintre terminale al IP-ului într-un serviciu de transport de încredere între procese. TCP utilizează şi controlul congestiei. Controlul congestiei este un serviciu pentru binele general al Internet-ului nu doar un serviciu pentru aplicația curenta. Controlul congestiei la TCP-ului previne inundarea de către gazdele comunicante a legăturilor si a comutatoarelor cu o cantitate excesivă de trafic. În principiu, TCP permite conexiunilor TCP să traverseze o legătura congestionată partajând în mod egal lățimea de bandă a acelei legături. Acest lucru se face prin reglarea ratei la care un emițător TCP poate trimite date in rețea.

Spre deosebire de TCP, traficul UDP nu este reglat automat. O aplicație care foloseşte transportul UDP poate să facă trafic la orice rată doreşte, pentru cât timp vrea.

Un protocol care oferă transport de date de încredere şi controlul congestiei este în mod necesar un protocol complex. Pentru a acoperi principiile transferului de date fiabil şi a controlului congestiei, sunt necesare mai multe secțiuni şi chiar secțiuni suplimentare pentru a acoperi protocolul TCP în sine.

6.5. Multiplexarea şi demultiplexarea aplicațiilor

În această secțiune vom analiza multiplexarea şi demultiplexarea la nivel transport, care reprezintă extinderea serviciului de livrare gazdă la gazdă furnizat de nivelul rețea la un serviciu de

livrare proces la proces pentru aplicațiile ce rulează pe acea gazdă. Pentru a menține discuția concretă, vom analiza acest serviciu în contextul protocoalelor Internet. Menționăm totuşi că serviciul de multiplexare/ demultiplexare este necesar pentru toate rețelele de calculatoare.

La gazda destinație, nivelul transport primeşte segmente de la nivelul rețea aflat mai jos. Nivelul transport are responsabilitatea livrării datelor conținute în aceste segmente către aplicațiile corespunzătoare ce rulează pe acea gazdă. Să aruncăm o privire asupra următorului exemplu: presupunem că ne aflăm în fața calculatorului şi descărcăm pagini Web iar în acelaşi timp rulăm o sesiune FTP şi două sesiuni Telnet/SSH. Avem deci patru procese (aplicații de rețea) care rulează: două procese Telnet, un proces FTP şi un proces HTTP. Atunci când nivelul transport de pe acea gazdă primeşte date de la nivelul rețea, acesta trebuie să direcționeze datele recepționate către unul din aceste procese. Să examinăm modul în care se face acest lucru.

Un proces (care este parte a unei aplicații de rețea) poate dispune de unul sau mai multe socluri; astfel nivelul transport nu livrează datele direct către proces ci către soclurile intermediare. Deoarece la un moment de timp pot exista mai multe socluri la gazda receptoare, fiecare soclu necesită un identificator unic. Formatul acestui identificator depinde de tipul soclului: TCP sau UDP, aşa cum vom vedem în continuare.

Să analizăm modul în care gazda receptoare direcționează un segment sosit la nivel transport către soclul corespunzător. Fiecare segment de nivel transport dispune de o serie de câmpuri; nivelul transport de la receptor examinează aceste câmpuri pentru a identifica soclul receptor şi direcționează segmentul către acel soclu. Acțiunea de livrare a datelor dintr-un segment de nivel transport către soclul corespunzător poartă denumirea de **demultiplexare**. Acțiunea de colectare a datelor de la diversele socluri de pe gazda sursă, încapsularea datelor cu informații de antet pentru crearea de segmente şi transferul acestora către nivelul rețea poartă denumirea de **multiplexare**. Remarcăm faptul că nivelul transport din Figura 6.2 trebuie să demultiplexeze segmente ce sosesc de la nivelul rețea către procesul P1 sau către procesul P2; acest lucru este efectuat prin direcționarea datelor din segmentele sosite către soclul procesului corespunzător. Nivelul transport trebuie de asemenea să colecteze date de la aceste socluri, să formeze segmente de nivel transport şi să le transfere nivelului rețea. Cu toate că am introdus multiplexarea şi demultiplexarea în contextul protocoalelor de transport din Internet, trebuie înțeles că astfel de acțiuni trebuie efectuate mereu atunci când un protocol este folosit de mai multe protocoale de nivel superior.

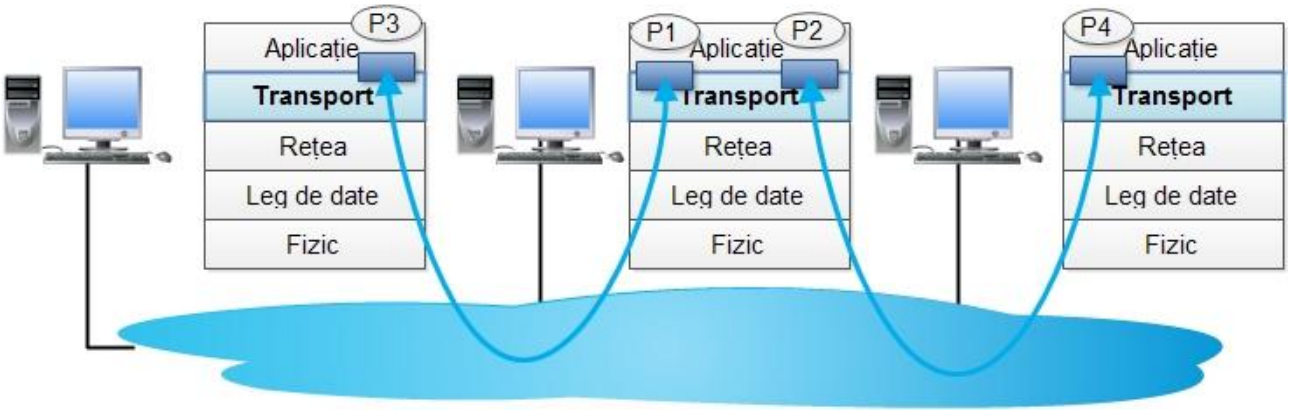

Figura 6.2 Multiplexarea şi demultiplexarea la nivel transport

Din discuția de mai sus, deducem că multiplexarea la nivel transport necesită:

- Ca soclurile să aibă identificatori unici
- Ca fiecare segment să dispună de câmpuri speciale care indică soclul căruia i se livrează segmentul

Aceste câmpuri speciale, ilustrate în Figura 6.3 sunt **numărul de port sursă** şi **numărul de port destinație** (segmentele TCP şi UDP mai dispun şi de alte câmpuri). Fiecare număr de port este un întreg pe 16 biți, ce poate lua valori de la 0 la 65535. Acest interval este divizat în trei categorii: porturi bine-cunoscute (de la 0 la 1023), porturi înregistrate (de la 1024 la 49151) şi porturi dinamice sau private (de la 49152 la 65535) [RFC 1700]. Porturile bine-cunoscute sunt rezervate pentru protocoale de nivel aplicație uzuale cum ar fi HTTP (portul 80), FTP (portul 21), SMTP (portul 25) etc. Pentru ca o aplicație să utilizeze un port bine-cunoscut aceasta necesită de regulă privilegii de administrator pe gazda respectivă. Atunci când dezvoltăm o aplicație trebuie să-i asignăm un număr de port.

Port sursă	Port destinație	Alte câmpuri	Încărcătura utilă

Figura 6.3 Porturi sursă şi destinație în antetul segmentelor

În acest moment ar trebui să fie clar modul în care nivelul transport ar putea implementa serviciul de demultiplexare: fiecare soclu de la gazdă are asignat un număr de port, iar atunci când soseşte un segment la gazdă, nivelul transport examinează portul destinație din segment şi direcționează segmentul către soclul corespunzător. Datele din segment traversează soclul către procesul ataşat. Aşa cum vom vedea, UDP-ul procedează în acest mod. Pentru TCP însă multiplexarea/demultiplexarea este ceva mai complicată.

6.5.1. Multiplexarea şi demultiplexarea fără conexiune

Să ne reamintim faptul că în Python se poate crea un soclu UDP astfel:

```
clientSocket = socket(AF_INET, SOCK_DGRAM)
```

Atunci când se creează un soclu UDP folosind instrucțiunile de mai sus, nivelul transport îi asignează în mod automat un număr de port. Mai exact nivelul transport asignează un număr de port din intervalul 1024-65535 care în mod curent nu este utilizat de un alt soclu UDP. Alternativ, putem adăuga o line în program, care după crearea soclului să-i asocieze un număr de port specificat (de ex. 12345) folosind metoda `bind`:

```
clientSocket.bind(('', 12345))
```

De obicei partea de client a aplicației, lasă alegerea portului în seama nivelului transport, pe când partea de server asignează un număr de port specificat.

Având numere de porturi asignate soclurilor UDP, putem descrie în detaliu multimplexarea/demultiplexarea UDP. Presupunem că un proces de pe gazda A având numărul de port 19157 doreşte să trimită date aplicație unui proces de pe gazda B având numărul de port 46428. Nivelul transport de la gazda A creează un segment de nivel transport care include datele aplicație, numărul portului sursă (19157), numărul portului destinație (46428), precum şi alte câmpuri. Apoi nivelul transport transferă segmentul nivelului rețea. Nivelul rețea încapsulează segmentul într-o datagramă IP şi încearcă să livreze datagrama la gazda destinație folosind serviciul best-effort. Dacă segmentul ajunge la gazda B, nivelul transport de la receptor examinează portul destinație (46428) din antetul segmentului şi livrează segmentul soclului identificat prin portul 46428. Remarcăm faptul că gazda B ar fi putu rula mai multe procese, fiecare având propriul soclu şi număr de port asociat. Pe măsură ce sosesc segmente din rețea, gazda B direcționează (demultiplexează) fiecare segment către soclul corespunzător examinând numărul de port destinație al segmentului.

Este important de remarcat că un soclu UDP este complet identificat printr-un tuplu format din adresa IP şi numărul de port destinație. Ca o consecință, dacă două segmente UDP au adrese IP sursă sau porturi sursă diferite, însă au aceeaşi adresă IP şi port destinație atunci cele două segmente vor fi direcționate către acelaşi proces prin acelaşi soclu.

Poate v-ați întrebat care este rolul portului sursă? Aşa cum se poate vedea din Figura 6.4 în segmentul A->B portul sursă serveşte ca adresă de retur – atunci când B doreşte să-i răspundă lui A, portul destinație din segmentul B->A va fi obținut din valoarea portului sursă a segmentului A->B (Adresa completă de retur este adresa IP a lui A şi portul sursă). Ca exemplu, în

UDPServer.py, pentru extragerea adresei IP şi portului clientului din segmentele sosite se foloseşte metoda `recvfrom()`; apoi se trimite un segment nou clientului, având ca destinație adresa IP şi portul extras.

```
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
```

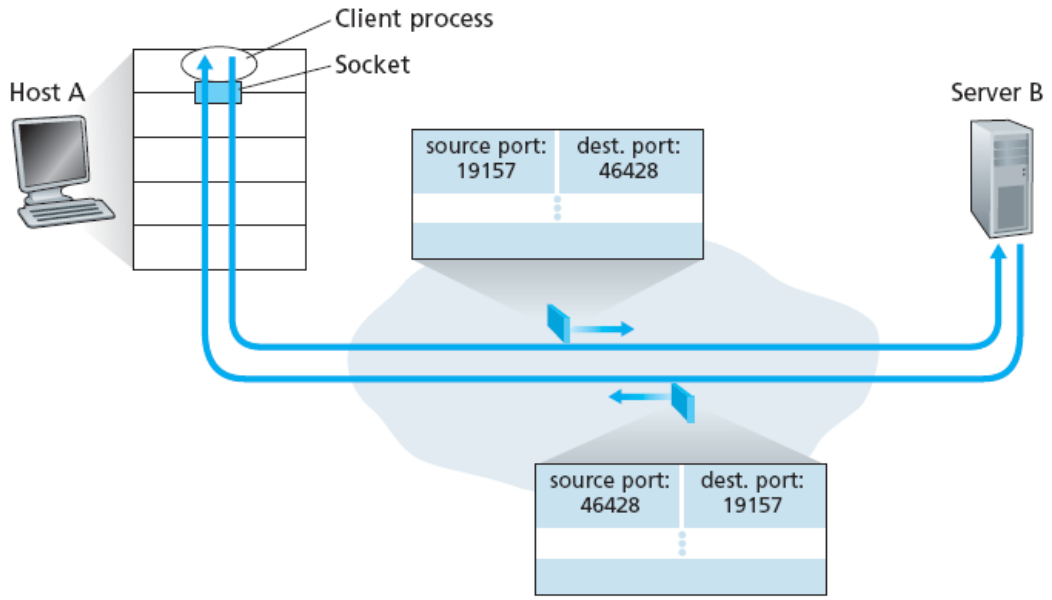

Figura 6.4 Inversarea portului sursă şi a portului destinație

6.5.2. Multiplexarea şi demultiplexarea orientată conexiune

Pentru a înțelege demultiplexarea TCP va fi nevoie să aruncăm o privire mai detaliată asupra soclurilor TCP şi stabilirii conexiunii TCP. O diferența subtilă între un soclu TCP şi un soclu UDP este faptul că un soclu UDP este identificat printr-un tuplu de patru elemente (adresă IP sursă, port sursă, adresă Ip destinație, port destinație). Astfel, atunci când la o gazdă soseşte un segment TCP din rețea , gazda va utiliza toate cele patru valori pentru a direcționa (demultiplexa) segmentul către soclul corespunzător. În particular, spre deosebire de UDP, două segmente TCP sosite având adrese IP sursă diferite, sau porturi sursă diferite vor fi direcționate către două socluri diferite. (cu excepția segmentului TCP de stabilire a conexiunii). Pentru detalii suplimentare vom avea în vedere programarea folosind socluri TCP:

- Aplicația serverul TCP dispune de un „soclu de întâmpinare”, care aşteaptă cereri de stabilire a conexiunii provenite de la clienții TCP pe portul 12345
- Clientul TCP creează un soclu şi trimite cererea de stabilire a conexiunii folosind liniile de cod:

clientSocket = socket (AF_INET, SOCK_STREAM)

clientSocket.connect((serverName,12345))

- o cerere de stabilire a conexiunii nu este nimic altceva decât un segment TCP având ca destinație portul 12345 având setat un bit special din antetul TCP. Segmentul include şi portul sursă ales de client.
- Atunci când sistemul de operare al gazdei care rulează aplicația server primeşte segmentul de cerere de conectare având ca destinație portul 12345, acesta identifică procesul server care aşteaptă sosirea de conexiunii pe portul 12345. Apoi procesul server creează un nou soclu:

  connectionSocket, addr = serverSocket.accept()
- De asemenea, nivelul transport de la server extrage următoarele patru valori din segmentul de stabilirea a conexiunii (1) port sursă, (2) adresă IP sursă (3) port destinație şi (4)propria adresă IP. Noul soclu de conexiune creat este identificat prin aceste patru valori; toate segmentele ce vor sosi în continuare având valori identice pentru câmpurile menționate vor fi demultiplexate către acest soclu. Având conexiunea stabilită, clientul şi serverul pot comunica.

Gazda server poate suporta mai multe socluri TCP conectate, fiecare soclu fiind ataşat la un proces, fiecare soclu fiind identificat de propriul 4-tuplu. Atunci când la gazdă soseşte un segment TCP, pentru direcționarea segmentului spre soclul corespunzător se folosesc toate cele patru câmpuri (adresă IP sursă, port sursă, adresă Ip destinație, port destinație).

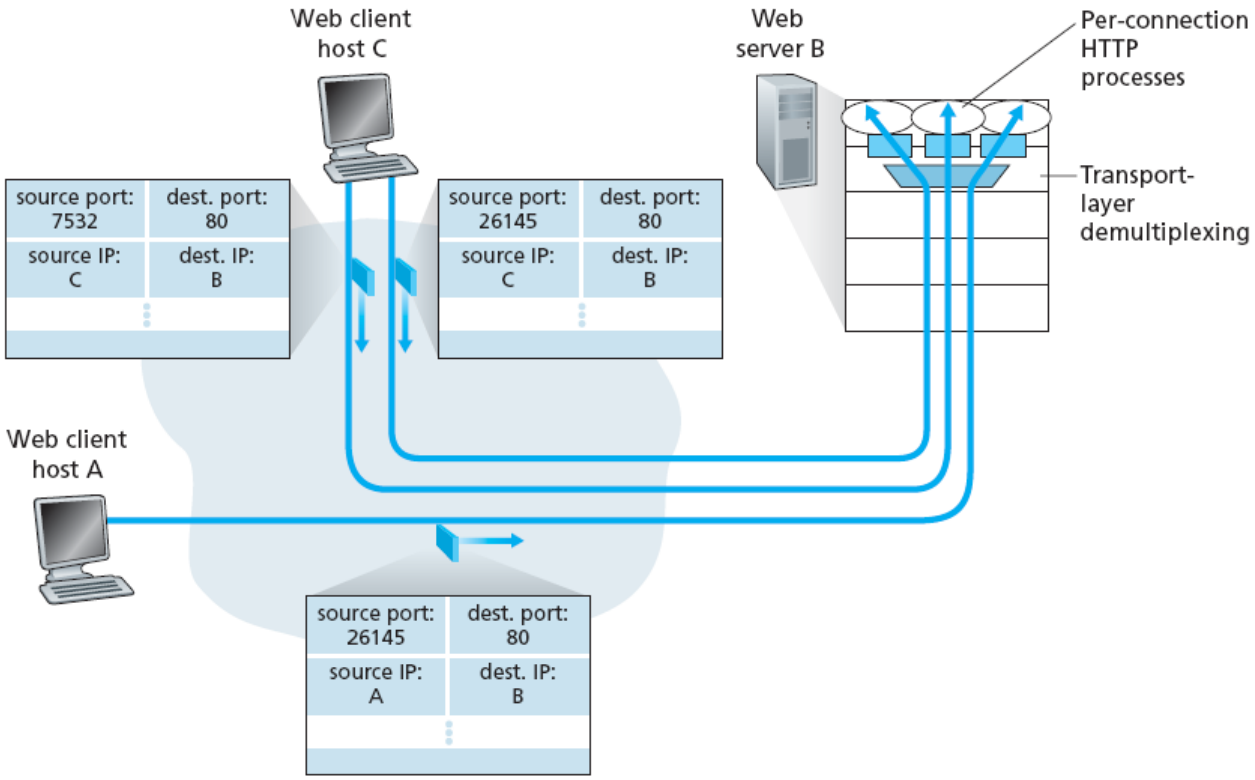

Figura 6.5 Doi client care utilizează acelaşi număr de port destinație (80) pentru a comunica cu serverul

Situația este ilustrată în Figura 6.5, unde gazda C inițiază două sesiuni HTTP cu serverul B şi gazda A inițiază o sesiune HTTP cu B. Gazda A şi C şi serverul B au fiecare propriile adrese IP unice: adresa A, adresa B şi adresa C. gazda C asignează două numere de port diferite (26145 şi 7532) pentru sesiunile HTTP. Deoarece gazda A alege numerele de port sursă independent de C, gazda C poate asigna numărul de port sursă 26145 pentru conexiunea sa HTTP. Însă nu va exista nici un fel de problemă – serverul B va fi capabil să demultiplexeze corect cele două conexiuni având acelaşi număr de port sursă, întrucât cele două conexiuni au adrese Ip sursă diferite.

6.6. Protocolul UDP

Stiva de protocoale Internet pune la dispoziție un protocol de transport fără conexiune, UDP (User Datagram Protocol – Protocolul Datagramă Utilizator) descris în RFC768. Acesta permite aplicațiilor de rețea trimiterea de mesaje încapsulate în datagrame IP cu supraîncărcare minimă, fără a fi nevoie de stabilirea unei conexiuni.

Antetul UDP constă din 8 octeți, fiind prezentat în Figura 6.6. Antetul este urmat de informația utilă. Portul sursă şi destinație asigură identificarea punctelor capăt de pe gazda sursă şi destinație. La sosirea unui segment UDP nivelul transport realizează demultiplexarea, livrând informația utilă procesului ataşat portului destinație. Avantajul UDP-ului față de folosirea directă a protocolului IP este tocmai prezența porturilor sursă şi destinație care permit multiplexare/demultiplexarea. Fără câmpurile port sursă şi port destinație nivelul de transport nu ar şti cărui proces să-i livreze segmentul; cu ajutorul lor, segmentele se livrează corect.

Port sursă (16 biți)	Port destinație (16 biți)
Lungime (16 biți)	Sumă de control (16 biți)

Figura 6.6 Antetul segmentelor UDP

Portul sursă este în primul rând necesar atunci când trebuie transmis înapoi la sursă un răspuns. Prin copierea câmpului port sursă din segmentul care soseşte în câmpul port destinație al segmentului care pleacă, procesul ce trimite răspunsul specifică ce proces de pe gazda expeditoare urmează să-l primească.

Câmpul lungime include antetul de 8 octeți şi datele, deci valoarea minimă este 8. Câmpul sumă de control UDP este opțional şi stocat ca 0 (zero) dacă nu este calculat (o valoare de adevăr 0 rezultată în urma calculelor este memorată ca un şir de biți 1). Dezactivarea acestuia nu este benefică, excepție făcând cazul în care calitatea informației chiar nu contează (de exemplu, transmisii audio în timp real).

Merită probabil menționate,în mod explicit, unele dintre lucrurile pe care UDP-ul nu le face. Nu realizează controlul fluxului, controlul erorii, sau retransmiterea unui segment recepționat incorect. Toate acestea sunt la latitudinea proceselor utilizatorului. Ceea ce face este să ofere protocolului IP o interfață cu facilități adăugate de demultiplexare a mai multor procese, folosind porturi. Aceasta este tot ceea ce face UDP-ul.

Pe lângă transmisiile audio-video în timp real, UDP-ul se foloseşte şi în cazul aplicațiilor de tip client-server orientate pe tranzacții; clientul trimite o cerere scurtă server-ului şi aşteaptă înapoi un răspuns scurt. Dacă se pierde ori cererea ori răspunsul, clientul poate pur şi simplu să încerce din nou după ce a expirat timpul. În acest fel vor fi necesare mai puține mesaje (câte unul în fiecare direcție) decât la un protocol care solicită o inițializare inițială, însă eventuale retransmisii trebuie tratate la nivel aplicație. O aplicație care foloseşte UDP-ul în acest fel este DNS-ul, deja studiat. Pe scurt, un program care trebuie să caute adresele de IP ale unor nume gazdă, de exemplu www.unitbv.ro, poate trimite un pachet UDP, conținând numele gazdei, către un server DNS. Serverul răspunde cu un pachet UDP conținând adresa IP a gazdei. Doar două mesaje traversează rețeaua. Nu este necesară nici o inițializare în avans şi nici o închidere a conexiunii.

Câteva caracteristici ale UDP-ului sunt:

- Fiind un protocol capăt la capăt poate identifica procesele ce rulează pe gazde
- este un serviciu de tip datagramă: cererile de trimitere de date primite de la nivelul superior sunt tratate independent;
- comunicarea are loc fără stabilirea unei conexiuni logice: nu există mecanisme de stabilire şi terminare a unei conexiuni; toate datele sunt trimise în cadrul unei singure datagrame IP,
- nu se garantează ajungerea ala destinație a datelor: nu există mecanisme de confirmare, ajungerea la destinație nu este anunțată sursei;
- nu se garantează ajungerea segmentelor în ordinea în care au fost trimise
- datele transportate sunt protejate de o sumă de control (deşi este inițial introdusă ca opțională, ea este specificată ca necesară în RFC 1122);
- supraîncărcarea introdusă este minimă: doar 8 octeți.

Câteva utilizări tipice ale UDP-ului:

- servicii de rezolvare a numelor (DNS) şi servicii de alocare automată a adreselor IP (DHCP): deoarece întrebările şi răspunsurile scurte pot fi mai eficient implementate peste UDP;
- fluxuri multimedia: deoarece mecanismele complicate de confirmare şi control al fluxului ale TCP-ului ar deprecia interactivitatea;

- server de fişiere (NFS): deoarece acest tip de aplicații sunt în general rulate în rețele locale cu fiabilitate ridicată care nu necesită mecanismele TCP;
- protocoale de rutare (RIP).
- managementul rețelei (SNMP);

Rezumat

Protocolul nivelului transport oferă o comunicație logică între procesele aplicație ce rulează pe diferite gazde, nefiind implementate în ruterele din rețea ci numai în terminale. O rețea de calculatoare poate pune la dispoziția aplicațiilor de rețea mai mult de un protocol de nivel transport. Un protocol al nivelului transport oferă comunicație logică între procesele care rulează pe diferite gazde. Protocolul nivelului rețea pentru Internet este denumit IP. Fiecare gazdă are o adresă IP care oferă comunicații logice cu alte gazde. Internetul, şi în general o rețea TCP/IP, pune la dispoziție nivelului aplicație două protocoale de transport distincte: UDP şi TCP. Nivelul rețea din Internet (protocolul IP) livrează datagramele doar între două sisteme gazdă făcând necesar un nivel suplimentar care să transfere mesajele între procesele aplicație care rulează pe aceste stații gazdă. Extinderea livrării gazdă–la-gazdă la livrarea proces-la-proces este sarcina serviciului de multiplexare/demultiplexare implementat la nivelul transport. Stiva de protocoale Internet pune la dispoziție un protocol de transport fără conexiune, UDP care permite aplicațiilor de rețea trimiterea de mesaje încapsulate în datagrame IP transmiterea lor cu supraîncărcare minimă, fără a fi nevoie de stabilirea unei conexiuni precum şi un protocol orientat conexiune TCP, care pune la dispoziție un flux de octeți sigur între procesele comunicante.

Bibliografie

Andrew S. Tanenbaum, *Computer Networks*, 4/E, Prentice Hall, 2003

James F. Kurose and Keith W. Ross, *Computer Networking a Top Down Approach*, 5/E, Pearson Education, 2009

William Stallings, *Data and Computer Communications*, 9/E, Pearson Education, 2011