4. Nivelul transport

4.a. Caracteristici ale nivelului Transport

# Adresarea

Atunci când un proces aplicaţie (de exemplu, un proces utilizator) doreşte să stabilească o conexiune cu un proces aflat la distanţă, el trebuie să specifice cu care proces doreşte să se conecteze. . Metoda folosită în mod normal este de a defini adrese de transport la care procesele pot să aştepte cereri de conexiune.

Un scenariu posibil pentru stabilirea unei conexiuni la nivel transport este următorul:

1. Un proces server care furnizează ora exactă şi care rulează pe gazda 2 se ataşează la TSAP 122 aşteptând un apel. Felul în care un proces se ataşează la un TSAP nu face parte din modelul de reţea şi depinde numai de sistemul de operare local. Poate fi utilizat un apel de tip LISTEN din capitolul precedent.
2. Un proces aplicaţie de pe gazda 1 doreşte să afle ora exactă; atunci el generează un apel CONNECT specificând TSAP 1208 ca sursă şi TSAP 1522 ca destinaţie. Această acţiune are ca rezultat în cele din urmă stabilirea unei conexiuni la nivel transport între procesele aplicaţie de pe gazda 1şi serverul 1 de pe gazda 2.
3. Procesul aplicaţie trimite o cerere o cerere pentru timp
4. Procesul server de timp răspunde cu timpul curent.
5. Conexiunea transport este apoi eliberată.

# Stabilirea conexiunii

La prima vedere, ar părea suficient ca o entitate de transport să trimită numai un TPDU CONNECTION REQUEST şi să aştepte replica CONNECTION ACCEPTED . Problema apare deoarece reţeaua poate pierde, memora sau duplica pachete. Acest comportament duce la complicaţii serioase. Se poate imagina o subreţea care este atât de congestionată încât confirmările ajung greu înapoi, şi, din această cauză, fiecare pachet ajunge să fie retransmis de câteva ori.

Punctul crucial al problemei este existenţa duplicatelor întârziate. El poate fi tratat în mai multe feluri, dar nici unul nu este într-adevăr satisfăcător. O posibilitate este de a utiliza adrese de transport valabile doar pentru o singură utilizare. O altă posibilitate este de a atribui fiecărei conexiuni un identificator (adică, un număr de secvenţă incrementat pentru fiecare conexiune stabilită), ales de cel care iniţiază conexiunea, şi pus în fiecare TPDU, inclusiv în cel care iniţiază conexiunea. Se poate încerca şi o altă soluţie. În loc să se permită pachetelor să trăiască la nesfârşit în subreţea, se poate inventa un mecanism care să elimine pachetele îmbătrânite.

# Eliberarea conexiunii

Există două moduri de a termina o conexiune: eliberare simetrică şi eliberare asimetrică. Eliberarea asimetrică este bruscă şi poate genera pierderi de date. După stabilirea conexiunii, gazda 1 trimite un TPDU care ajunge corect la gazda 2. Gazda 1 mai trimite un TPDU dar, înainte ca acesta să ajungă la destinaţie, gazda 2 trimite DISCONNECT REQUEST . În acest caz, conexiunea va fi eliberată şi vor fi pierdute date.

Eliberarea simetrică este utilă atunci când fiecare proces are o cantitate fixă de date de trimis şi ştie bine când trebuie să transmită şi când a terminat. În alte situaţii însă, nu este deloc uşor de determinat când trebuie eliberată conexiunea şi când a fost trimis tot ce era de transmis. S-ar putea avea în vedere un protocol de tipul următor: atunci când 1 termină, trimite ceva de tipul:

Am terminat. Ai terminat şi tu? Dacă gazda 2 răspunde: Da, am terminat. Închidem! conexiunea poate fi eliberată în condiţii bune. Din nefericire, acest protocol nu merge întotdeauna. Binecunoscuta problemă a celor două armate este similară acestei situaţi.

# Controlul fluxului

Una din problemele cheie a apărut şi până acum: controlul fluxului. La nivel transport există asemănări cu problema controlului fluxului la nivel legătură de date, dar există şi deosebiri.

La nivel legătură de date, emiţătorul trebuie să memoreze cadrele transmise, pentru că poate fi necesară retransmiterea acestora. Dacă subreţeaua oferă un serviciu datagramă, atunci entitatea de transport emiţătoare va trebui să memoreze pachetele trimise din aceleaşi motive.

Pe scurt, dacă serviciul reţea nu este sigur, emiţătorul va trebui să memoreze toate TPDU-urile trimise, la fel ca la nivel legătură de date. Totuşi, folosind un serviciu la nivel reţea sigur sunt posibile unele compromisuri. În particular, dacă emiţătorul ştie că receptorul are întotdeauna tampoane disponibile, atunci nu trebuie să păstreze copiile TPDU-urilor trimise. Totuşi, dacă receptorul nu poate garanta că orice TPDU primit va fi acceptat, emiţătorul va trebui să păstreze copii. În ultimul caz, emiţătorul nu poate avea încredere în confirmarea primită la nivel reţea, deoarece aceasta confirmă sosirea TPDU-ului la destinaţie, dar nu şi acceptarea lui.

Chiar dacă receptorul va realiza memorarea temporară a mesajelor primite, mai rămâne problema dimensiunii tamponului. Dacă dimensiunea tampoanelor ar fi constantă, egală cu cel mai mare TPDU posibil, atunci va apărea o risipă de spaţiu ori de câte ori este primit un TPDU mai scurt. Dacă dimensiunea tampoanelor este aleasă mai mică decât cel mai mare TPDU posibil, atunci pentru memorarea unui TPDU mai lung vor fi necesare mai multe tampoane, iar complexitatea operaţiei va creşte.

O altă soluţie este utilizarea unor tampoane de dimensiune variabilă. Avantajul este o mai bună utilizare a memoriei, cu preţul unei gestiuni a tampoanelor mai complicată. O a treia posibilitate este alocarea unui singur tampon circular pentru fiecare conexiune. Această soluţie are de asemenea avantajul unei utilizări eficiente a memoriei, dar numai în situaţia în care conexiunile sunt relativ încărcate.

# Multiplexarea

Multiplexarea mai multor conversaţii pe conexiuni, circuite virtuale şi legături fizice joacă un rol important în mai multe niveluri ale arhitecturii reţelei. În cazul nivelului transport, multiplexarea poate fi necesară din mai multe motive. De exemplu, dacă doar o singură adresă de reţea este disponibilă pe o gazdă, toate conexiunile transport de pe acea maşină trebuie să o folosească. Când un TDPU soseşte este necesar un mod de a spune cărui proces trebuie dat. Această situaţie numită multiplexare în sus.

Multiplexarea poate să fie utilă nivelului transport şi din alt motiv, legat de deciziile tehnice şi nu de politica de preţuri ca până acum. Să presupunem, de exemplu, că o subreţea foloseşte intern circuite virtuale şi impune rată de date maximă p fiecare dintre ele. Dacă un utilizator are nevoie de mai multă lăţime de bandă decât poate oferi un circuit virtual, o soluţie este ca nivelul transport să deschidă mai multe conexiuni reţea şi să distribuie traficul prin acestea (într-un sistem round-robin), .Acest mod de operare se numeşte multiplexare în jos.

4.b. Protocolul tcp

**TCP** (**Transport Communication Protocol** - protocol de comunicaţie de nivel transport) a fost proiectat explicit pentru a asigura un flux sigur de octeţi de la un capăt la celălalt al conexiunii într-o inter-reţea nesigură.

O caracteristică importantă a TCP, care domină structura protocolului, este aceea că fiecare octet al unei conexiuni TCP are propriul său număr de secvenţă, reprezentat pe 32 biţi.

Entităţile TCP de transmisie şi de recepţie interschimbă informaţie sub formă de segmente. Un segment TCP constă dintr-un antet de exact 20 de octeţi (plus o parte opţională) urmat de zero sau mai mulţi octeţi de date. Programul TCP este cel care decide cât de mari trebuie să fie aceste segmente.

Protocolul de bază utilizat de către entităţile TCP este protocolul cu fereastră glisantă. Atunci când un emiţător transmite un segment, el porneşte un cronometru. Atunci când un segment ajunge la destinaţie, entitatea TCP receptoare trimite înapoi un segment (cu informaţie utilă, dacă aceasta există sau fără, în caz contrar) care conţine totodată şi numărul de secvenţă următor pe care aceasta se aşteaptă să-l recepţioneze. Dacă cronometrul emiţătorului depăşeşte o anumită valoare înaintea primirii confirmării, emiţătorul retransmite segmentul neconfirmat.

Deşi acest protocol pare simplu, pot apărea multe situaţii particulare care vor fi prezentate mai jos. Segmentele pot ajunge într-o ordine arbitrară, deci octeţii 3072-4095 pot fi recepţionaţi, dar nu pot fi confirmaţi datorită absenţei octeţilor 2048-3071. Segmentele pot de asemenea întârzia pe drum un interval de timp suficient de mare pentru ca emiţătorul să detecteze o depăşire a cronometrului şi să le retransmită. Retransmisiile pot include porţiuni de mesaj fragmentate altfel decât în transmisia iniţială, ceea ce impune o tratare atentă, astfel încât să se ţină evidenţa octeţilor primiţi corect. Totuşi, deoarece fiecare octet din flux are un deplasament unic faţă de începutul mesajului, acest lucru se poate realiza. TCP trebuie să fie pregătit să facă faţă unor astfel de situaţii şi să le rezolve într-o manieră eficientă. Un efort considerabil a fost dedicat optimizării performanţelor fluxurilor TCP, ţinându-se cont inclusiv de probleme legate de reţea.