4. Nivelul transport

4.a. Caracteristici ale nivelului Transport

Adresarea

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

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

1. Un proces server care furnizează ora exactă și care rulează pe gazda 2 se atașează la TSAP 122 așteptând un apel. Felul în care un proces se atașează la un TSAP nu face parte din modelul de rețea și depinde numai de sistemul de operare local. Poate fi utilizat un apel de tip LISTEN din capitolul precedent.
2. Un proces aplicație de pe gazda 1 dorește să afle ora exactă; atunci el generează un apel CONNECT specificând TSAP 1208 ca sursă și TSAP 1522 ca destinație. Această acțiune are ca rezultat în cele din urmă stabilirea unei conexiuni la nivel transport între procesele aplicație de pe gazda 1și serverul 1 de pe gazda 2.
3. Procesul aplicație 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 și să aștepte replica CONNECTION ACCEPTED . Problema apare deoarece rețeaua poate pierde, memora sau duplica pachete. Acest comportament duce la complicații serioase. Se poate imagina o subrețea care este atât de congestionată încât confirmările ajung greu înapoi, și, din această cauză, fiecare pachet ajunge să fie retransmis de câteva ori.

Punctul crucial al problemei este existența 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 secvență incrementat pentru fiecare conexiune stabilită), ales de cel care inițiază conexiunea, și pus în fiecare TPDU, inclusiv în cel care inițiază conexiunea. Se poate încerca și o altă soluție. În loc să se permită pachetelor să trăiască la nesfârșit în subrețea, se poate inventa un mecanism care să elimine pachetele îmbătrânite.

Eliberarea conexiunii

Există două moduri de a termina o conexiune: eliberare simetrică și eliberare asimetrică. Eliberarea asimetrică este bruscă și 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 destinație, gazda 2 trimite DISCONNECT REQUEST . În acest caz, conexiunea va fi eliberată și vor fi pierdute date.

Eliberarea simetrică este utilă atunci când fiecare proces are o cantitate fixă de date de trimis și știe bine când trebuie să transmită și când a terminat. În alte situații însă, nu este deloc ușor de determinat când trebuie eliberată conexiunea și 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 și tu? Dacă gazda 2 răspunde: Da, am terminat. Închidem! conexiunea poate fi eliberată în condiții bune. Din nefericire, acest protocol nu merge întotdeauna. Binecunoscuta problemă a celor două armate este similară acestei situați.

Controlul fluxului

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

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

Pe scurt, dacă serviciul rețea nu este sigur, emițătorul va trebui să memoreze toate TPDU-urile trimise, la fel ca la nivel legătură de date. Totuși, folosind un serviciu la nivel rețea sigur sunt posibile unele compromisuri. În particular, dacă emițătorul știe că receptorul are întotdeauna tampoane disponibile, atunci nu trebuie să păstreze copiile TPDU-urilor trimise. Totuși, dacă receptorul nu poate garanta că orice TPDU primit va fi acceptat, emițătorul va trebui să păstreze copii. În ultimul caz, emițătorul nu poate avea încredere în confirmarea primită la nivel rețea, deoarece aceasta confirmă sosirea TPDU-ului la destinație, dar nu și 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 spațiu 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 operației va crește.

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

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

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

4.B. PROTOCOLUL TCP

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

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

Entitățile TCP de transmisie și de recepție interschimbă informație sub formă de segmente. Un segment TCP constă dintr-un antet de exact 20 de octeți (plus o parte opțională) urmat de zero sau mai mulți octeți de date. Programul TCP este cel care decide cât de mari trebuie să fie aceste segmente.

Protocolul de bază utilizat de către entitățile TCP este protocolul cu fereastră glisantă. Atunci când un emițător transmite un segment, el pornește un cronometru. Atunci când un segment ajunge la destinație, entitatea TCP receptoare trimite înapoi un segment (cu informație utilă, dacă aceasta există sau fără, în caz contrar) care conține totodată și numărul de secvență următor pe care aceasta se așteaptă să-l recepționeze. Dacă cronometrul emițătorului depășește o anumită valoare înaintea primirii confirmării, emițătorul retransmite segmentul neconfirmat.

Deși acest protocol pare simplu, pot apărea multe situații particulare care vor fi prezentate mai jos. Segmentele pot ajunge într-o ordine arbitrară, deci octeții 3072-4095 pot fi recepționați, dar nu pot fi confirmați datorită absenței octeților 2048-3071. Segmentele pot de asemenea întârzia pe drum un interval de timp suficient de mare pentru ca emițătorul să detecteze o depășire a cronometrului și să le retransmită. Retransmisiile pot include porțiuni de mesaj fragmentate altfel decât în transmisia inițială, ceea ce impune o tratare atentă, astfel încât să se țină evidența octeților primiți corect. Totuși, deoarece fiecare octet din flux are un deplasament unic față de începutul mesajului, acest lucru se poate realiza. TCP trebuie să fie pregătit să facă față unor astfel de situații și să le rezolve într-o manieră eficientă. Un efort considerabil a fost dedicat optimizării performanțelor fluxurilor TCP, ținându-se cont inclusiv de probleme legate de rețea.