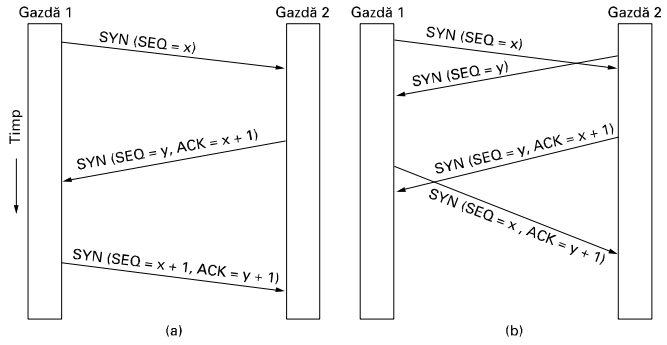
8. Nivelul transport
(completare)

Sursa: A.S. Tanenbaum, *Retele de Calculatoare*, Ed. 4, BYBLOS 2003

Stabilire Conexiune TCP

Se face dupa modelul general de intelegere in trei pasi dar folosind mesajele si conventiile specifice protocolului TCP. Figura prezinta mesajele schimbate intre ele de **entitatile de protocol**, reprezentate prin Gazda 1 si Gazda 2, ca urmare a executiei unor primitive de serviciu, a receptiei unor mesaje de la entitatile pereche sau a unor evenimente interne (de exemplu timeout-uri).

In cazul (a), la executia unei primitive CONNECT, Gazda 1 trimite un segment cu SYN=1, ACK=0, in care specifica un numar de secventa initial SEQ=x. Situatia reprezentata presupune ca la Gazda 2 s-a executat deja primitiva LISTEN si deci un proces utilizator este in asteptarea unei cereri de conectare. La receptia segmentului SYN, Gazda 2 verifica daca procesul utilizator accepta conexiunea. In caz afirmativ, Gazda 2 va trimite un segment cu SYN=1, propriul numar de secventa initial SEQ=y si cu ACK=x+1 (deoarece SYN consuma un octet, datele asteptate incep de la x+1 nu de la x). La receptia acestui segment, Gazda 1 trimite un nou segment SYN, care incheie protocolul de conectare. Segmentul are date de la SEQ=x+1 si confirma receptia corecta a segmentului de la Gazda 2 prin ACK=y+1.



(a) Cazul normal. (b) Coliziune.

Cazul (b) corespunde executiei unui CONNECT in ambele sisteme. Cele doua entitati de protocol trimit mesaje SYN cu numerele de secventa initiale si apoi raspunsuri "simetrice" ca urmare a carora conexiunea se stabileste corect.

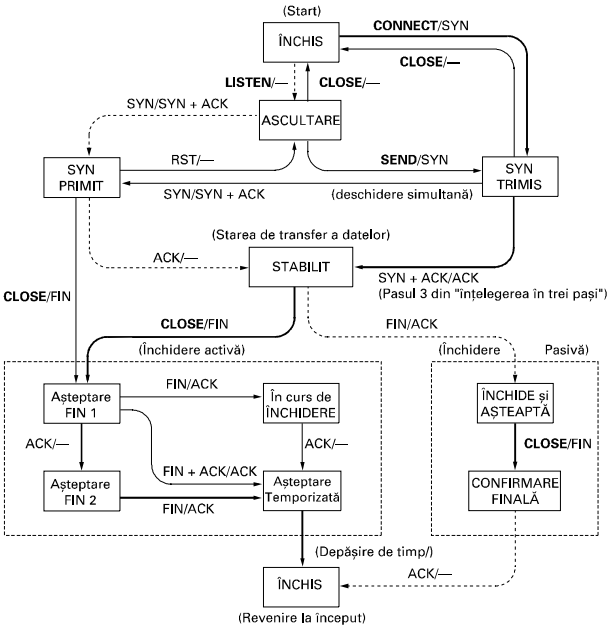
Management Conexiune TCP

Starile folosite in managementul conexiunii TCP

Stare	Descriere
CLOSED (ÎNCHIS)	Nici o conexiune nu este activă sau în aşteptare
LISTEN (ASCULTARE)	Serverul aşteaptă recepţionarea unui apel
SYN RCVD (Recepţie SYN)	S-a recepţionat o cerere de conexiune; aştept ACK
SYN SENT (Transmisie SYN)	Aplicaţia a început deschiderea unei conexiuni
ESTABLISHED (STABILIT)	Starea normală de transfer a datelor
FIN WAIT 1 (Aşteptare FIN 1)	Aplicaţia a anunţat că termină
FIN WAIT 2 (Aşteptare FIN 2)	Partenerul este de acord cu eliberarea conexiunii
TIMED WAIT (Aşteptare Temporizată)	Se aşteaptă „moartea” tuturor pachetelor
CLOSING (În curs de INCHIDERE)	Ambele părţi încearcă simultan închiderea
CLOSE WAIT (ÎNCHIDE şi AŞTEAPTĂ)	Partenerul a iniţiat eliberarea conexiunii
LAST ACK (CONFIRMARE FINALĂ)	Se aşteaptă „moartea” tuturor pachetelor

Stabilirea si desfiintarea Conexiunilor TCP

Stabilirea unei conexiuni poate fi urmarita si pe diagrama de stari din figura urmatoare, care prezinta modelul tranzitional (partial) al protocolului TCP. Ea descrie starile si tranzitiile unei entitati de protocol de transport. Intelegerea functionarii se bazeaza pe ideea ca cele doua entitati de protocol se pot afla la un moment dat in stari diferite, in functie de rolul pe care-l joaca, acela de client sau de server. Pentru a evidentia aces lucru, tranzitiile realizate de client sunt reprezentate cu linie ingrosata continua, iar succesiunea de stari a unui server este marcata cu linie ingrosata intrerupta. Liniile subtiri corespund unor tranzitii exceptionale. Fiecare tranzitie este etichetata cu o pereche de simboluri despartite de slash si semnificand evenimentul care declanseaza tranzitia (intrarea automatului) respectiv actiunea executata (iesirea automatului).



Pentru ca stabilirea unei conexiuni TCP a fost detaliata mai inainte, ne referim la inchiderea ei. Descriem mai intai **inchiderea activa** realizata de client, apoi **inchiderea pasiva** realizata de server. Asa cum se va vedea, cele doua sensuri de transmisie ale unei conexiuni sunt inchise succesiv.

Cand aplicatia client executa o primitiva CLOSE (Inchidere activa in figura), entitatea de protocol transmite un segment FIN, anuntand prin aceasta ca a terminat de transmis mesaje si doreste sa inchida sensul de transmisie al conexiunii. Apoi trece in starea Asteptare FIN 1. Receptia unei confirmari ACK inseamna ca entitatea de protocol pereche este de acord cu inchiderea unui sens al conexiunii (ea a receptionat corect toate segmentele transmise anterior). Entitatea trece in starea Asteptare FIN 2, care semnifica inchiderea unui sens. Receptia unui FIN anunta ca si partenerul a terminat de transmis mesaje si vrea sa inchida al doilea sens de transmisie. Mesajul este confirmat prin ACK (evident, dupa verificarile de rigoare ca receptia segmentelor transmise de partener s-a facut corect) si entitatea de protocol trece in starea Asteptare Temporizata. In aceasta stare, entitatea de protocol asteapta un timp egal cu dublul duratei de viata a unui pachet pentru ca sa moara toate copiile mesajelor si

raspunsurilor trimise pe conexiune si duplicate (din eroare) de retea. La expirarea timpului, entitatea trece in starea INCHIS si sterge inregistrarea asociata conexiunii.

Daca entitatea de protocol primeste un segment FIN (Inchidere pasiva) care informeaza ca partenerul nu mai are de transmis, ea confirma prin ACK (dupa verificarile de rigoare ca receptia segmentelor transmise de partener s-a facut corect) si trece in starea INCHIDE si ASTEAPTA. Un sens al transmisiei (cel de la client la server) este astfel inchis. Cand aplicatia server a terminat de transmis si executa CLOSE, entitatea de protocol (de la server) transmite FIN si comuta in starea CONFIRMARE FINALA. La receptia unei confirmari ACK (care inseamna ca toate segmentele transmise anterior au fost primite bine de client) se inchide si al doilea sens al conexiunii si se trece in starea INCHIS.

Transmiterea de date se face atunci cand ambele entitati de protocol ale unei conexiuni TCP sunt in starea STABILIT. Printre altele, in aceasta stare, TCP trebuie sa faca controlul fluxului de date si controlul congestiei retelei. La ele ne referim in continuare.

Controlul fluxului de date

Controlul fluxului asigura ca segmentele ajunse la entitatea receptoare gasesc spatiu suficient in tamponul receptorului si sunt acceptate. Rejectarea lor ar insemna ca s-au consumat inutil timp si resurse importante din retea pentru aducerea segmentelor de la sursa la destinatar (foarte probabil pe un drum lung si cu multe noduri intermediare). Solutia adoptata in TCP este informarea transmitatorului asupra spatiului tampon disponibil la receptor, prin campul "Dimensiunea ferestrei" din antetul TCP.

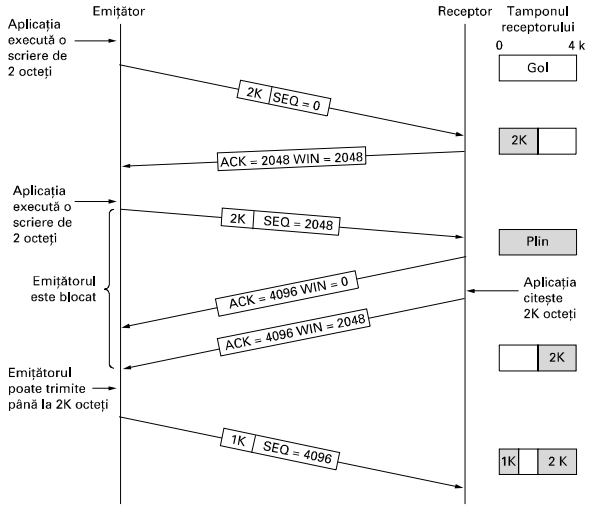


Figura arata cum este folosit acest camp. Pe langa informatiile incluse in figura, putem evidentia urmatoarele elemente:

- transmitatorul va tine evidenta spatiului disponibil in tampon pentru a nu depasi limita sa;
- "Numarul de secventa" dintr-un segment va fi folosit la receptie pentru plasarea datelor intr-o pozitie corespunzatoare in tamponul receptorului; se reface astfel ordinea datelor eventual alterata la transmiterea prin retea;
- politica de livrare a datelor din tampon catre aplicatie tine de implementare: se poate face livrarea intregului tampon sau a unei parti; livrarea se poate face imediat ce datele sunt primite sau cand se umple tamponul etc.
- in exemplul dat, tamponul este circular.

Controlul congestiei

Controlul fluxului prin metoda prezentata anterior nu garanteaza transmiterea corecta a unui segment. Acesta ar putea fi eliminat pe traseu de un ruter congestionat. Pentru a elimina astfel de evenimente, transmitatorul gestioneaza, pe langa fereastra de receptie, o a doua fereastra, dependenta de capacitatea retelei, numita **fereastra de congestie**. Fluxul de segmente va fi astfel limitat de valoarea minima dintre fereastra receptorului si fereastra de congestie. Mecanismele folosite la stabilirea ferestrei de congestie si la controlul congestiei sunt diferite de cele pentru fereastra receptorului.

Algoritmul de **stabilire a fereastrei** de congestie este urmatorul:

- transmitatorul trimite un segment de dimensiunea maxima convenita (sau implicita) pe conexiunea stabilita;
- transmitatorul dubleaza dimensiunea (prin transmiterea unei rafale de segmente) la fiecare transmisie confirmata la timp; in acest fel, cresterea dimensiunii ferestrei este exponentiala;
- la primul timeout se opreste procededeul si fereastra de congestie se stabilieste la valoarea ultimei transmisii confirmate la timp (fara timeout).

Daca pe parcursul transmisiei apar fenomenul de congestie, se foloseste urmatorul **algoritm de control**:

- algoritmul foloseste un **prag** (threshold)
- la producerea unui timeout, pragul este setat la jumatate din fereastra de congestie;
- se aplica apoi procedeul de **crestere exponentiala** a fereastrei de congestie pana se atinge pragul stabilit in pasul anterior;
- peste prag, se aplica in continuare o **crestere liniara** (cu cate un segment maxim la fiecare transmisie confirmata la timp) pana se obtine noua valoare a ferestrei de congestie.

Probleme dimensiuni campuri antet

Ca si in cazul adreselor IP, dimensiunea fixa a unor campuri din antetul TCP a devenit un factor de limtare. Ne referim in continuare la doua dintre campuri.

Campul **Numar de secventa** are 32 de biti putand reprezenta valori dintr-un spatiu destul de larg. El este totusi finit astfel incat, la sfarsitul unui ciclu, numararea se reia. Durata unui ciclu de numarare depinde de viteza transmisie. Ca urmare se poate calcula durata pentru diferite viteze de transfer:

- 1 saptamana pentru 56kbps
- 57 min pentru 10Mbps
- 34 sec pentru 1Gb **(sub 120 sec care este timpul viata maxim in Internet)**

Problema: cum se poate face diferenta intre un duplicat vechi si un segment nou care poarta acelasi numar de secventa?

Un al doilea factor limitativ este campul **Dimensiunea ferestrei** care are 16 biti si permite specificarea unui tmpon de pana la 2^{16} octeti. Acest lucru este neconvenabil pentru performanta retelelor de mare capacitate si latenta ridicata. Sa luam exemplul transmiterii unui segment de 500 Kb pe o legatura de 1 Gbps, care "ocupa" nu mai mult de 500 μsec din capacitatea canalului. La o intarziere de 20 ms, confirmarea se primeste dupa 40 ms, lucru ce conduce la o ocupare a canalului de 1.25%.

Ocuparea completa a canalului in ambele directii ar corespunde produsului **bandwidth*delay** care in acest caz este de 40 milioane de biti. Doar ca, atingerea acestui regim de lucru ar cere ca fereastra receptorului sa aiba o valoare cel putin egala cu acest produs.

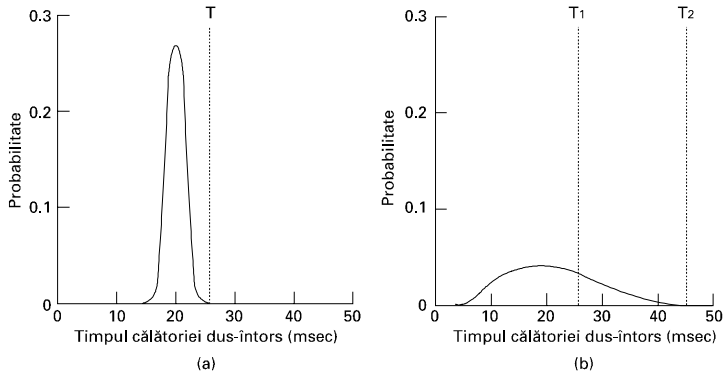
Problema: cum se poate specifica o fereastra mai mare ca 2^{16} octeti?

Solutiile acestor probleme se gasesc in folosirea optiunilor TCP specificate in RFC 1323. In principiu:

- Optiunea **TCP Timestamps** rezolva numere de secventa duplicate, deoarece doua segmente diferite cu acelasi Numar de secventa ar avea amprente de timp diferite, iar
- **Window Scale** permite definirea unui factor de scalare a campului **Window size** cu pana la 2**14 pozitii binare, ceeace da posibilitatea definirii unor ferestre de pana la 2**30 octeti.

Gestiunea ceasurilor in TCP

Setarea ceasului (durata de asteptare a unei confirmari) pentru o conexiune TCP are o importanta foarte mare pentru performanta conexiunii. O setare proasta conduce la performante slabe: daca timpul de timeout este prea lung, transmitatorul asteapta mult pentru retransmisie; daca este prea scurt, retransmiterea unui segment gereneaza un trafic posibil inutil daca segmentul a fost receptat corect dat mai tarziu decat a prevazut transmitatorul.



(a) Densitatea de probabilitate a timpilor de sosire ACK in nivelul legatura de date.

(b) Densitatea de probabilitate a timpilor de sosire ACK pentru TCP.

Stabilirea timeout-ului pentru o conexiune de transport este mai grea decat la o legatura de date, datorita dispersiei mai mari a timpilor de transmitere a segmentelor in cazul transportului (vezi figura). Evident, se poate folosi ca baza de calcul timpul dus-intors (*round trip time* - RTT) pentru un segment pe o legatura de transport. Doar ca, pentru conexiunea de transport, RTT variaza destul de mult in timp.

Solutia (datorata lui Jacobson) este pastrarea unei estimari cat mai bune a RTT si actualizarea ei in functie de timpul masurat M pana la primirea confirmarii unui segment transmis pe conexiune, conform formulei:

$$\mathbf{RTT = \alpha * RTT + (1- \alpha) * M}$$

cu $\mathbf{\alpha = 7/8}$ ales empiric (acest coeficient arata ponderea data vechii valori RTT in formula noii valori).

Cu aceasta, timeout-ul de retransmisie (*retransmission timeout* - RTO) se face cu formula

$$\mathbf{RTO = \beta * RTT}$$

cu $\mathbf{\beta = 2}$ ales, de asemenea, experimental.

O alegere mai buna foloseste *deviatia standard (DS)* a functiei de densitate a probabilitatii timpului de primire a confirmarilor. Pentru un calcul mai simplu, se foloseste **deviatia medie** D ca o aproximare a DS. Deviatia medie se poate actualiza conform formulei:

$$\mathbf{D = \alpha * D + (1- \alpha) * |RTT-M|}$$

(unde $\mathbf{\alpha}$ poate fi diferita de valoarea anterioara) iar pe baza ei se poate estima RTO:

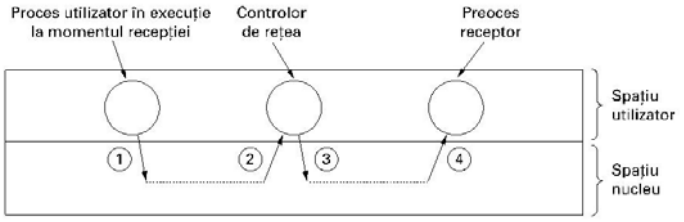
$$\mathbf{RTO = RTT + 4 * D}$$

unde factorul 4 este ales, de asemenea, empiric.

Proiectarea pentru performanta

Cresterea performantei canalelor de comunicatie (1 Gbps a devenit o realitate frecventa) a facut ca gatuirile in performanta retelelor sa se mute la nivelul calculatoarelor gazda. Tratarea lor reclama modificari (uneori importante) in solutia de sistem adoptata sau in implementarea protocoalelor.

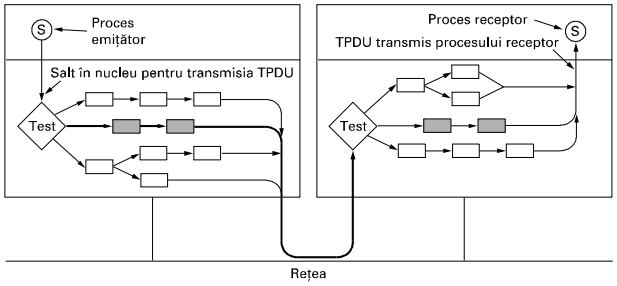
Un exemplu din prima categorie este alegerea plasamentului pentru anumite module de retea, in spatiul utilizator sau in spatiul nucleu. De regula, sosirea unui pachet provoaca o comutare de la procesul aflat in executie la nucleu unde este tratata receptia si o a doua comutare de la nucleu la procesul receptor pentru livrarea datelor. Figura arata cresterea numarului de comutari de context daca procesul Controlor de retea (care trateaza receptia) este plasat in Spatiul utilizator.



Patru schimbari de context pentru a manevra un pachet cu un Controlor de retea in spatiul utilizatorului.

Prelucrare Fast TPDU

Al doilea exemplu se refera la prelucrarea rapida a unei succesiuni de unitati de date de transport. In acest caz, antetele TCP si IP difera putin de la un segment la altul, ceeace permite compunerea pachetului de trimis pornind de la un **antet prototip**.



Calea rapida intre transmitator si receptor este cu line groasa.

Entitatea de transport va verifica daca transmisia este intr-un caz normal, adica:

- starea este ESTABLISHED,
- nu se incearca inchiderea conexiunii,
- exista suficient spatiu la receptor.

In acest caz se poate aplica procedura de prelucrare rapida.

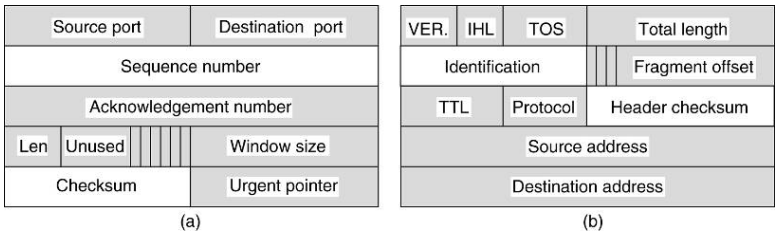
Prelucrare Fast TPDU (2)

In acest caz, entitatea de protocol va face urmatoarele operatii:

- copiaza antetul TCP din prototip in tamponul de transmisie - campurile gri in figura (a)
- completeaza Numarul de secventa (il copiaza din variabila contor pe care o incrementeaza apoi)
- calculeaza Suma de control
- paseaza antetul si datele entitatii de retea

Entitatea de retea:

- copiaza antetul de retea din prototipul corespunzator -campurile gri in figura (b)
- completeaza Identificatorul si calculeaza Suma de control.



(a) Antet TCP. (b) Antet IP.

Receptorul realizeaza urmatoarele:

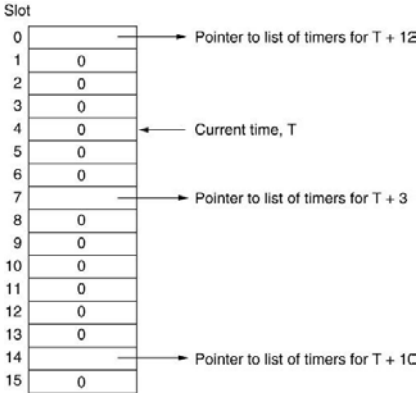
- Localizeaza **inregistrarea conexiunii**; la TCP ea este memorata intr-o tabela hash, pe baza adreselor IP si porturilor sursa si destinatie
- Testeaza daca se poate apllica procedura rapida - cazul normal (similar cu transmisia)
  - starea este ESTABLISHED,
    - nu se incearca inchiderea conexiunii,
    - unitatea de date este intreaga
    - nu sunt setati indicatori speciali
    - numarul de secventa este corect.
- Actualizeaza inregistrarea conexiunii
- Copiaza datele la utilizator si calculeaza suma de control
- Transmite confirmarea.

Prelucrare Fast TPDU (timer management)

"Timing wheel."

De regula, evenimentele de timeout sunt pastrate intr-o coada de prioritati. Fiecare element pastreaza numarul de impulsuri de ceas fata de elementul precedent. Elementul din capat pastreaza numarul de impulsuri pana la declansarea timeout-ului. La fiecare impuls, valoarea lui este decrementata cu o unitate, iar cand ajunge al zero se declanseaza evenimentul de timeout. In functie de implementarea adoptata, inserarea unui eveniment sau extragerea unui eveniment reprezinta operatii costisitoare.

Deoarece, in cazul TCP, majoritatea evenimentelor de timeout sunt anulate inainte de producere, este utila o implementare mai simpla care sa exploateze aceasta proprietate. In implementarea mentionata aici se foloseste un vector (numit roata timpului). Fiecare slot corespunde unui moment de timp. Un pointer arata timpul curent si este avansat cu un slot, circular, la fiecare impuls de ceas. Fiecarui slot ii este atasata o lista de evenimente care trebuie sa se produca la momentul respectiv. Un nou eveniment este programat prin simpla adaugare la lista slotului corespunzator, In figura se arata pozitiile listelor de la momentele T+3, T+10 si T+12, unde T=4 este timpul curent.

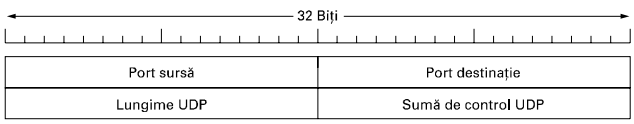


Pentru anulare se cauta in lista de la slotul corespunzator si se elimina evenimentul din acea lista. Daca lista unui slot este nevida, se proceseaza toate evenimentele din lista.

UDP - User Datagram Protocol

UDP livreaza datagrame utilizator - *user datagrams*, numite astfel pentru a face diferenta fata de datagrame, care identifica unitati de date folosite pentru a caracteriza o anumita organizare interna a nivelului retea.

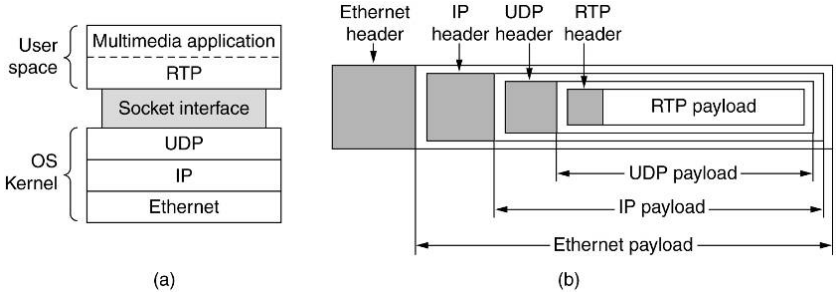
Livrarea datagramelor se bazeaza pe capacitatea retelei, conform principiului "Best effort". Asta inseamna ca datagramele pot fi pierdute, primite in alta ordine etc. Facilitatea remarcabila adaugata de UDP nivelului retea este adresarea punctelor de capat UDP, pentru care se folosesc numere de porturi si adresele IP. UDP include in antet doar numerele de porturi. Antetul mai contine un camp pentru suma de control, dar de regula acesta este ignorat. De asemenea, are un camp pentru lungimea datagramei, care include antetul si datele.



UDP nu face controlul fluxului, al erorilor si nu retransmite datagramele. Este folosit in aplicatii client-server (precum DNS), care transmit unitati mici de date si fac corectiile prin mecanisme proprii.

The Real-Time Transport Protocol

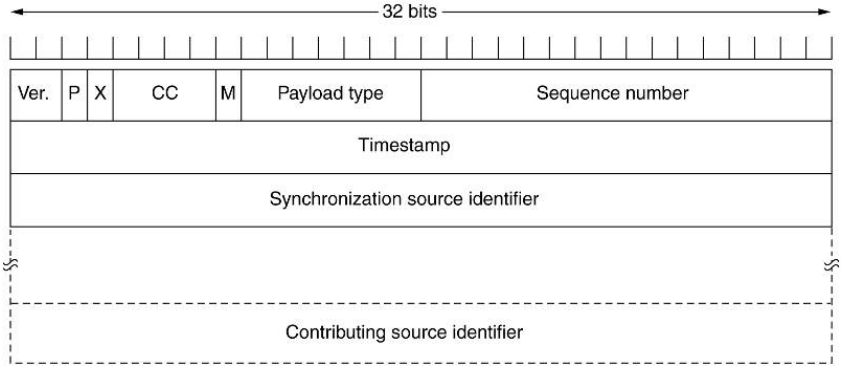
RTP (**Real-time Transport Protocol**) este un protocol folosit pentru suportul aplicatiilor multimedia. In stiva de protocoale, el este pozitionat intre nivelele utilizator is UDP si are ca functie principala multiplexarea mai multor fluxuri RTP intr-un flux UDP, care este trimis catre una sau mai multe destinatii (unicast sau multicast).



(a) Pozitia RTP in stiva de protocoale. (b) Format pachet

Antet RTP

RTP nu face retransmiteri, solutia receptorului fiind aceea de a obtine pachetul absent prin interpolarea pachetelor vecine. Pentru evidenta lor, TCP foloseste in antet un camp **Numar de secventa**. De asemenea, fiecare pachet are o **Amprenta de timp**, utila in multe aplicatii multimedia pentru a reda esantioanele la momentele de timp potrivite. Amprenta de timp este relativa fata de inceputul fluxului. De asemenea, amprentele de timp permit redarea sincronizata a mai multor fluxuri (de ex. audio cu video).



Antetul contine si alte informatii. Bitul *P* (padding) este setat daca pachetul a fost extins la un multiplu de 4 octeti. Ultimul octet extins ne spune câți octeți au fost adăugați.

Bitul *X* (extension) indică prezența unui antet extins. Formatul şi semnificația antetului extins nu sunt definite. Singurul lucru care este definit este acela că primul cuvânt al extensiei dă lungimea. Aceasta este o cale de scăpare pentru orice cerințe neprevăzute.

Campul *CC* arata numarul de identificatori de surse contribuabile (Contributing source identifier) prezente in antet.

Bitul *M* (Marker) este un bit de marcare specific aplicației. Poate fi folosit pentru a marca începutul unui cadru video, începutul unui cuvânt într-un canal audio sau altceva ce aplicația înțelege.

Tip continut (Payload type) indica formatul (codificarea) continutului.

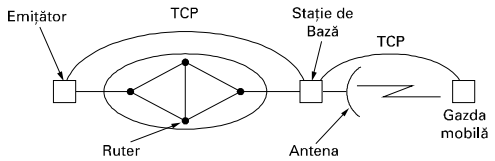
Identificatorul sursei de sincronizare - Synchronization Source (SSRC) este ales aleator si spune carui flux ii apartine pachetul. Toate pachetele de la aceeasi sursa de sincronizare folosesc acelasi spatiu de numere de secventa si amprente de timp, permitand receptorului sa grupeze pachetele dupa aceeasi sursa de sincronizare pentru redare.

I*dentificatorii surselor contributoare* - Contributing source (CSRC) arata sursele care au contribuit la continutul din pachetul curent. De exemplu, la o audioconferinta arata toti vorbitorii ale caror cuvantari au fost mixate in pachet pentru a permite receptorului sa identifice vorbitorul curent, chiar daca toate pachetele audio au acelasi identificator al sursei de sincronizare.

TCP si UDP fara fir

Metodele de tratare a congestiei pomenite anterior pot conduce la probleme nedorite in cazul unor retele bazate pe alte tehnologii decat cele din retelele fixe (cu cabluri). Reactia unei entitati TCP la aparitia de timeout-uri este incetinirea ritmului de transmisie. Ea se bazeaza pe presupunerea ca timeout-urile se datoreaza congestiei si ca aceasta solutie va duce la rezolvarea congestiei. Acest ratinament un este valabil si in cazul retelelor fara fir, care sunt mai putin fiabile. La aparitia unor timeout-uri pentru o astfel de retea, entitatea de protocol ar trebui sa mareasca ritmul, in ideea cresterii sansei mesajelor de a ajunge la destinatie in conditi de fiabilitate proasta. In cazul unor inter-retele eterogene, cele doua mecanisme de solutionare devin contradictorii. Prezentam doua cai de solutionare a acestei probleme.

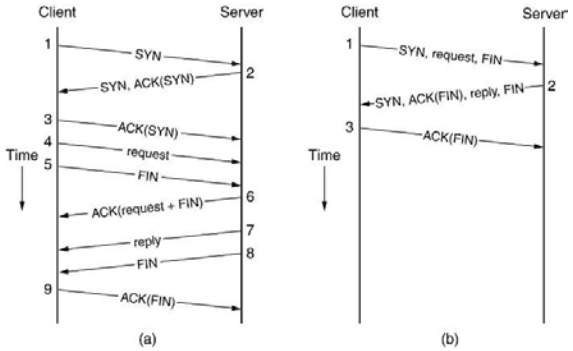
Prima (propusa de Bakne şi Badrinath) imparte conexiunea TCP in doua conexiuni separate, una pentru reteaua cablata si una pentru reteaua fara fir, si aplica politici diferite de tratare a timeout-urilor pe fiecare conexiune (vezi figura). Inconvenientul acestei solutii este violarea semanticii TCP, care un mai este o conexiune capat-loa-capat, ci una care include Statia de Baza ca punct intermediar al conexiunii.



O alta solutie (Balakrishnan et al.) consta in adaugarea unui agent de supraveghere care intercepteaza pachetele TCP care pleaca spre gazda mobila precum si confirmarile care se intorc de la aceasta. Agentul va retransmite catre gazda mobila, fara stirea sursei, segmentul pentru care nu se primeste la timp o confirmare a receptiei acestuia si va elimina confirmarile duplicate pentru a nu fi interpretate de sursa drept semne de congestie. Agentul poate, de asemenea, sa ceara retransmisia de catre gazda mobila a segmentelor pierdute generate de ea, inainte ca aceste pierderi sa fie detectate de partenerul din reteaua cablata.

TCP Tranzactional

UDP este folosit pentru schimburi scurte (gen "intrebare-raspuns") de segmente de mici dimensiuni. Cand volumul de date transmis este mai mare apar probleme carora UDP nu le face fata. De exemplu, poate fi necesara segmentarea mesajelor si asigurarea ca ele ajung corect si in ordine. Dar la asta se pricepe TCP.



(a) RPC folosind TPC normal. (b) RPC folosind T/TCP.

Dezavantajul ar fi totusi supraincarcarea ceruta de stabilirea unei conexiuni si de desfiintarea ei, doar pentru a face un schimb de doua mesaje (unul de "dus" si unul de "intors"). Situatia este ilustrata in figura (a), in care se arata schimbul de segmente complet, inclusiv cele de management al conexiunii.

1. Clientul trimite un pachet SYN pentru a stabili o conexiune.
2. Serverul trimite un pachet ACK pentru a recunoaşte pachetul SYN.
3. Clientul finalizează cu ACK înțelegerea în trei paşi.
4. Clientul trimite cererea.
5. Clientul trimite FIN pentru a arata ca s-a terminat trimiterea.
6. Serverul confirmă cererea şi FIN-ul.
7. Serverul trimite răspunsul înapoi clientului.
8. Serverul trimite un pachet FIN pentru a incheia transmiterea.
9. Clientul confirmă prin ACK si conexiunea se inchide.

TCP tranzactional permite transferul de date odata cu stabilirea conexiunii. Schimbul de segmente este ilustrat in figura (b).

1. Clientul deschide conexiunea cu SYN, transmite cererea si incheie transmiterea cu FIN, toate intr-un singur segment.
2. Dupa ce calculeaza rezultatul, serverulraspunde la deschiderea conexiunii cu SYN, confirma deschiderea de la client (ACK), transmite raspunsul si inchide conexiunea cu FIN, torul intr-un singur segment.
3. Clientul face al treilea pas si confirma inchiderea conexiunii.