

TRANSMISSION CONTROL PROTOCOL

PROGRAM INTERNET DARPA

SPECIFICATIILE PROTOCOLULUI

traducere realizata de Catalin Bulancea, Info III, gr4A

Septembrie 1981

realizat pentru

Agentia de Proiecte Avansate in Cercetarea pentru Aparare
Biroul de Tehnici de Procesare a Informatiei
Bulevardul Wilson 1400
Arlington, Virginia 22209

de
Institutul de Stiinte ale informatiei
Universitatea South Carolina
Calea Admiralty 4676
Marina del Rey, California 90291

Septembrie 1981

Transmission Control Protocol

Cuprins:

PREFATA.....	iii
1. INTRODUCERE.....	1
1.1 Motivatie.....	1
1.2 Scop	2
1.3 Despre Document	2
1.4 Interfete	3
1.5 Operatii.	3
2. FILOSOFIE	7
2.1 Elemente ale Sistemului Inter-retele	7

2.2	Model de Operatie	7
2.3	Mediul Gazda	8
2.4	Interfete	9
2.5	Relatie cu alte Protocoale	9
2.6	Comunicatie de incredere	9
2.7	Stabilirea Conexiunii si eliberarea ei.....	10
2.8	Comunicatii de Date	12
2.9	Precedenta si Securitate	13
2.10	Principiul Robustetii	13
3.	SPECIFICATII FUNCTIONALE	15
3.1	Formatul Headerului	15
3.2	Terminologie	19
3.3	Numere de secventa.....	24
3.4	Stabilirea unei conexiuni	30
3.5	Inchiderea unei Conexiuni.....	37
3.6	Precedenta si Securitate	40
3.7	Comunicatii de Date.....	40
3.8	Interfete	44
3.9	Procesare de evenimente.....	52
GLOSAR	79
REFERINTE	85

PREFATA

Acest document descrie Standardul DOD Transmission Control Protocol(TCP). Au fost deja 9 editii anterioare de specificatii ARPA TCP pe care se bazeaza acest standard si textul de fata se bazeaza puternic pe acestea. Au fost o multime de coautori ai acestei lucrari atat in definirea conceptelor cat si in realizarea textului. Aceasta editie clarifica cateva detalii si inlatura ajustarile de dimensiune a buffer-ului de tip sfarsit-de-litera si redescrie mecanismul de litera ca o functie Adauga (Push).

Jon Postel

Editor

RFC: 793
Replaces: RFC 761
IENs: 129, 124, 112, 81,
55, 44, 40, 27, 21, 5

TRANSMISSION CONTROL PROTOCOL

PROGRAMUL DARPA INTERNET SPECIFICATIILE PROTOCOLULUI

1. INTRODUCERE

Transmission Control Protocol (TCP) este folosit ca un protocol gazda catre gazda(host to host) de incredere intre gazde(hosts) in comunicarea prin schimbul de pachete intre computerele in retea si in sistemele inter-conectate de tip retea.

Acest document descrie functiile care trebuie indeplinite de Transmission Control Protocol (TCP) , programul care implementeaza acest protocol si interfata catre programe sau utilizatori care necesita serviciile TCP.

1.1. Motivare

Sistemele de comunicare intre calculatoare joaca un rol din ce in ce mai important in domeniile militar, guvern, civil. Acest document isi concentreaza in mod special atentia pe cerintele de comunicare intre calculatoare in domeniul militar, robuste in prezenta unor comunicatii pe care nu te poti baza, disponibilitatea in cazul congestiei, dar multe dintre aceste probleme se regasesc si in sectorul guvernului sau in cel civil.

Privite ca o modalitate de comunicare strategica si tactica intre calculatoare, retelele sunt dezvoltate si plasate fiind esentiala procurarea de mijloace de a le inter-conecta si de a produce protocoale standard de comunicare intre procese care sa poata suporta o banda larga de aplicatii. In anticiparea nevoilor pentru asemenea standarde, asistentul sub-secretar de stat in Cercetari si Inginerie pentru Aparare a declarat Transmission Control Protocol (TCP) descris in specificatia de fata ca fiind baza pentru standardizarea protocolului DOD-wide de comunicare inter-procese.

TCP un protocol de incredere orientat pe conexiuni, capat-la-capat realizat pentru o ierarhie in crestere de protocoale care suporta aplicatii multi-retea. TCP este folosit pentru comunicarea de incredere intre perechi de procese apartinand calculatoarelor gazda care la randul lor apartin unor retele distincte, inter-conectate. Foarte putine presupuneri au fost facute in privinta increderii in protocolul de comunicare sub stratul TCP . TCP isi propune sa obtina un serviciu simplu de transmitere de datagrame, posibil inconsistent(lipsit de incredere) pentru nivel inferior de protocoale. In principiu, TCP ar trebui sa fie in stare sa opereze pe un spectru larg de sisteme de comunicatie de la conexiunile prin fir pana la retelele cu trimitere de pachete sau de circuite.

TCP se bazeaza pe concepte descrise intaia oara de Cerf si Kahn in [1]. TCP se

potriveste intr-o arhitectura stratificata pe protocol imediat superioara unui Protocol de Internet [2] care da posibilitatea TCP-ului sa trimita si sa primeasca segmente de informatii de lungime variabila inscrite in datagrame internet numite si "plicuri". Datagrama internet ofera mijlocul de adresare a TCP-urilor sursa si dstinatie in diferite retele. Protocolul internet are de asemenea in grija orice fragmentare sau asamblare a segmentelor TCP necesare transportului si livrarii prin retele multiple si porti inter-conectate. Protocolul internet transporta informatii cu privire la precedenta , securitatea si compartimentarea segmentelor TCP, asa incat aceasta informatie poate fi comunicata de la capat-la-capat prin mai multe retele.

Stratificarea Protocoalelor

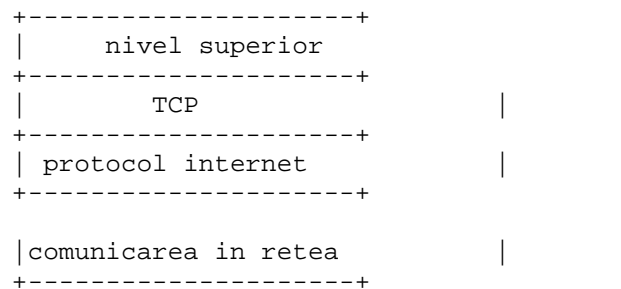


Figura 1

O mare parte a acestui document este scrisa in contextul implementarilor TCP care se regasesc in protocoale de nivel inalt in calculatorul gazda. Unele calculatoare la retea prin calculatoare front-end care gazduiesc straturile de protocoale TCP si internet, ca si soft-ul de retea specific. Specificatiile TCP descriu o interfata pentru protocoale de nivel inalt care pot fi implementate chiar si pentru cazul front-end, atata timp cat este implementat un protocol host-to front end potrivit.

1.2. Scopul

TCP isi propune sa ofere o comunicare proces-catre-proces de incredere intr-un mediu multi-retea. TCP doreste sa fie un protocol gazda-la-gazda de uz comun in retele multiple.

1.3 Despre acest document

Acest document reprezinta o specificare a comportamentului cerut de orice implementare TCP, atat in interactiunea cu un nivel mai inalt de protocoale cat si in interactiunea cu alte TCP-uri. Restul acestei sectiuni ofera o foarte scurta privire asupra interfetelor si operatiilor protocol. A doua sectiune sumeaza baza filosofiei designului TCP . Sectiunea 3 ofera o descriere detaliata a actiunilor cerute de TCP cand apar evenimente variate (ajungerea de noi segmente, apeluri din partea utilizatorului, erori, etc.) si detaliile formatului segmentelor TCP.

1.4 Interfete

Interfetele TCP sunt pe de o parte catre utilizator sau procese de aplicatii si pe de alta parte catre nivele inferioare de protocoale cum ar fi Protocolul Internet.

Interfata intre procese de aplicatii si TCP este ilustrata in detaliu. Aceasta interfata consta dintr-o serie de apeluri asemanatoare cu apelurile pe care un sistem de operare le ofera pentru un proces aplicatie pentru manipularea de fisiere. De exemplu, exista apeluri de deschidere si de inchidere a conexiunilor si de trimitere sau de primire de date prin conexiuni stabilite. Este de asemenea de asteptat ca TCP-ul sa comunice asincron cu programe aplicatii. Desi cei care implementeaza designul interfetelor se bucura de o libertate considerabila, se cere totusi o functionalitate minima pentru interfata TCP/utilizator pentru orice implementare valida.

Interfata intre TCP si rotocolul de nivel inferior este in mod esential nespecificata cu exceptia ca se presupune ca exista un mecanism prin care cele doua nivele isi pot trimite informatii una alteia. De obicei ne asteptam ca protocolul de nivel inferior sa specifice interfata. TCP este realizata pentru a functiona intr-un mediu foarte general de interfete. Protocolul de nivel inferior presupus in acest document este Protocolul Internet. [2]

1.5 Operatii

Cum am spus mai sus, scopul pricipal al TCP este de a oferi un serviciu de conexiune sau un circuit logic sigur si de incredere intre perechi de procese. Pentru a oferi asemenea serviciu sigur bazat pe un sistem mai putin sigur cum este comunicarea prin internet se cer facilitati in urmatoarele domenii:

- Transfer Primar de Date
- Incredere
- Control al curgerii informatiei
- Multiplexare
- Conexiuni
- Precedenta si Securitate

Operatiile de baza ale TCP in fiecare din aceste domenii sunt descrise in paragrafele urmatoare.

Transfer Primar de Date:

TCP este in stare sa transfere un flux de octeti in fiecare directie intre utilizatori impachetand un nr de octeti in segmente pentru transmiterea prin internet. In general, TCP-urile decid cand blocheaza si cand transmit date.

Uneori utilizatorii trebuie sa fe sigur ca datele trimise de ei catre TCP au fost transmise in intregime. Pentru aceasta se defineste o functie Push. Pentru a se asigura ca datele trimise TCP-ului au fost transmise in intregime expeditorul indica impingerea(push-ul) catre destinatar. Un asemenea push determina TCP-ul sa trimita prompt datele pana la punctul destinatarului. Punctul exact din care se face push-ul nu este vizibil destinatarului si functia

push nu ofera nici o inregistrare ??? a record boundary marker.???

Incredere:

TCP trebuie sa revina la o stare sigura de dinaintea coruperii, pierderii, duplicarii sau transiterii haotice de sistemul de comunicatie prin internet a unor date. Acest lucru se atinge asignand o secventa ordonata de numere fiecarui octet transmis, iar la TCP-ul destinatari se face o recunoastere (ACK - acknowledgement) a datelor primite. Daca ACK nu este primita intr-un anumit interval de timp, datele sunt retransmise. La destinatari, secventa de numere este folosita pentru a ordona in ordine corecta segmentele care ar fi putut fi primite intr-o ordine aleatoare si pentru a elimina duplicatele. Coruperea datelor e preintampinata prin adaugarea unei sume de control la fiecare segment transmis, verificandu-se suma la destinatari si eliminand segmentele corupte.

Atata timp cat TCP continua sa functioneze corespunzator si sistemul de internet nu devine complet partitionat, nici o eroare nu va afecta transmiterea corecta a datelor. TCP isi revine din erorile provenite din sistemul de comunicatie prin internet.

Controlul curgerii informatiei:

TCP ofera mijloace pentru destinatari de a manevra cantitatea de date transmisa de expeditor. Acest lucru se realizeaza returnand o "fereastră" cu fiecare ACK indicand o raza acceptabila de secvente de numere dupa ultimul segment primit cu succes. Fereastră indica un nr de octeti permis pe care expeditorul poate sa-i transmita inaintea primirii permisiunii de transmitere a datelor urmatoare.

Multiplexare:

Pentru a permite mai multor procese dintr-un singur calculator gazda sa foloseasca facilitatile TCP simultan, TCP ofera un set de adrese sau porturi pentru fiecare host. Concatenate cu adresele de retea si cele de host din stratul de comunicatie internet ele formeaza un socket. O pereche de socketuri identifica in mod unic fiecare conexiune. Adica, un socket poate fi folosit simultan in mai multe conexiuni.

Legarea porturilor la procese este manevrata independent de fiecare host. Totusi, se dovedeste utila atasarea proceselor folosite frecvent (de ex un serviciu de timesharing sau "logger") la socketuri care sunt cunoscute publicului. Aceste servicii pot fi accesate prin adresele cunoscute. Stabilirea si invatarea adreselor de port ale altor procese pot implica unor mecanisme mai dinamice.

Conexiuni:

Mecanismele de incredere si cele de control al curgerii informatiei descrise mai sus necesita ca TCP-ul sa initializeze si sa mentina o anumita stare a informatiei pentru fiecare flux de date. Combinarea acestei informatii incluzand socketurile, secventele de numere si dimensiunile ferestrei se numeste conexiune. Fiecare conexiune este in mod unic specificata de o pereche de socketuri ce indentifica cele doua capete.

Cand doua procese doresc sa comunice, TCP-urile lor trebuie mai intai sa stabileasca o conexiune(sa initializeze informatia de start la fiecare capat). Cand comunicatia se termina, conexiunea este intrerupta sau inchisa pentru a elibera resursele pentru alte scopuri.

De cand conexiunile trebuie stabilite intre gazde (hosts) instabile (nesigure) peste un sistem de comunicatie pe internet nesigur, un mecanism de tip handshake bazat pe o secventa de numere cu un ceas este folosit pentru evita initializarile eronate ale conexiunilor.

Precedenta si Securitate

Utilizatorii TCP pot foarte clar semnala precedenta si securitatea comunicarii dintre ei. Se face o dispozitie(clauza) pentru valorile implicite care vor fi utilizate atunci cand nu este nevoie de aceste trasaturi.

2. FILOSOFIE

2.1 Elemente ale sistemului inter-retele

Mediul inter-retele consta din host-uri conectate la retele care la randul lor inter-conectate prin gateways. Se presupune ca retelele pot fi locale (de ex. Ethernet) sau retele mari (de ex. ARPANET), dar in orice caz se bazeaza pe tehnologia de schimb de pachete. Agentii activi care produc si consuma mesaje sunt procesele. Diverse niveluri de protocoale de retea, gateways si gazde suporta sistemul de comunicatie inter-procese care ofera o comunicare de tip duplex (in ambele directii) in conexiunile logice dintre porturile proceselor.

Termenul pachet este aici folosit pentru a specifica tranzactiile de date dintre host si retea. Formatul de blocuri de date schimbat in retea nu constiuieste obiectul specificatiilor de fata.

Gazdele sunt calculatoare atasate la o retea si, din punctul de vedere al comunicarii in retea, sunt privite ca surse si destinatii pentru pachete. Procesele sunt vazute ca elemente active in calculatoarele gazda (in concordanta cu definitia proceselor = programe in executie). Pana si terminalele si fisierele sau alte dispozitive de I/O comunica intre ele prin intermediul proceselor. Astfel, intreaga comunicare e vazuta ca o comunicare intre procese.

Deoarece un proces trebuie sa faca diferenta dintre diferite fluxuri de comunicatie intre el si alte procese , vedem fiecare proces ca avand un nr de porturi prin care comunica cu porturile altor peocese.

2.2 Model de operatie

Procesele transmit date apeland TCP si pasand buffere de date ca argumente. TCP-ul impacheteaza datele din aceste buffere in segmente si apeleaza modulul internet pentru a transmite fiecare segment la TCP-ul destinatie. TCP-ul destinatie pune datele din pachetele primite intr-un buffer de primire si anunta utilizatorul de primirea lor. TCP include informatii de control in segmente folosite pentru a asigura o trimitere a datelor in ordine.

Modelul de comunicare prin internet se definește ca un modul de internet protocol asociat cu fiecare TCP care oferă o interfață în rețeaua locală. Acest modul internet împachetează segmentele TCP în cadrul datagramelor internet și rotează aceste datagrame la un modul internet destinație sau la un gateway intermediar. Pentru a transmite datagrama prin rețeaua locală ea este înglobată într-un pachet al rețelei locale.

Schimbul de pachete poate produce alte împachetări, fragmentări sau alte operații pentru a obține trimiterea pachetelor locale la modulul internet de destinație.

La o gateway între rețele datagrama internet este "despachetată" din pachetul sau local și examinată pentru a determina prin care rețea urmează datagrama să fie trimisă mai departe. Datagrama internet este apoi "împachetată" într-un pachet local potrivit rețelei următoare și rouată către următoarea gateway sau către destinația finală.

O poartă (gateway) poate desface o datagramă de tip internet în fragmente dacă e nevoie pentru a putea fi transmise mai departe în rețeaua următoare. Pentru a face aceasta, poarta produce un set de datagrame internet fiecare purtând un fragment. Fragmentul de datagramă are un format în așa fel încât modulul internet destinație să poată reasambla fragmentele în datagrame internet.

Un modul internet destinație desface segmentul din datagramă (după reasamblarea datagramei, dacă e necesar) și îl trimite la TCP-ul de destinație.

Acest model simplu de operație explică multe detalii. O trăsătură importantă este tipul de serviciu. Acesta oferă informații către poartă (sau către modulul internet) ghidând-o în selectarea parametrilor serviciu care vor fi folosiți în traversarea rețelei următoare. Inclusiv în acest tip de serviciu informație este precedentă datagramei. Datagramele pot de asemenea transporta informații de securitate care permit gazdelor și porturilor care operează în medii securizate pe mai multe nivele să separe datagramele din considerații de securitate.

2.3 Mediul gazda

TCP-ul se presupune a fi un modul în sistemul de operare. TCP poate apela alte funcții ale sistemului de operare, de exemplu pentru administrarea structurilor de date. Interfața către rețea se presupune a fi controlată de un modul driver de dispozitiv. TCP nu apelează driverul de dispozitiv de rețea direct ci mai degrabă apelează modulul de protocol de datagrame internet care poate apela driverul de dispozitiv.

Mecanismul TCP nu înlocuiește implementarea TCP-ului într-un procesor front-end. Totuși, în asemenea implementare, un protocol host-to front-end trebuie să ofere funcționalitate pentru a sprijini tipul de interfață TCP descrisă în acest document.

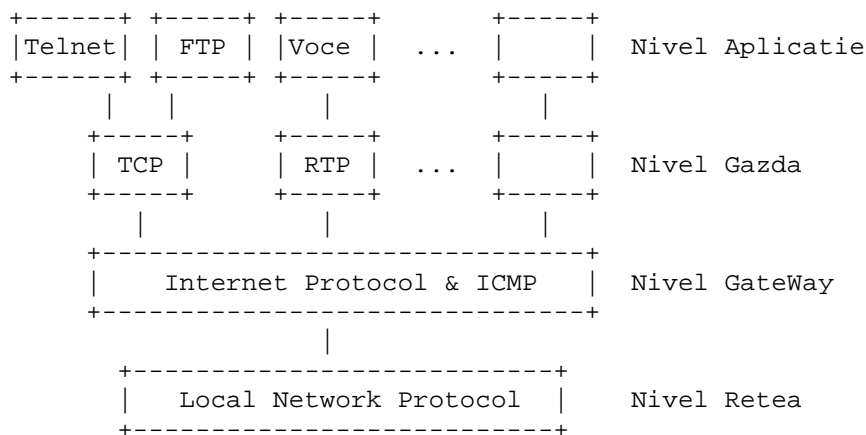
2.4 Interfete

Interfata TCP/utilizator ofera pentru apeluri ale utilizatorului catre TCP functii de OPEN sau CLOSE a conexiunii, SEND sau RECEIVE date, sau sa obtina un STATUS despre conexiune. Aceste apeluri sunt asemanatoare cu celelalte apeluri de la utilizator catre sistemul de operare, de exemplu, apelul de deschidere, citire din fisier si de inchidere a unui fisier.

Interfata TCP/internet ofera apeluri pentru trimiterea si receptionarea de datagrame adresate modulelor TCP in calculatoarele gazda oriunde in sistemul internet. Aceste apeluri au ca parametri adrese, tipuri de servicii, securitate si alte informatii de control.

2.5 Relatii cu alte protocoale

Urmatoarea diagrama ilustreaza locul protocolului TCP in ierarhia de protocoale:



Relatiile intre Protocoale

Figura 2.

Este de asteptat ca TCP sa fie in stare sa suporte un nivel superior de protocoale in mod eficient. Ar trebui sa fie simpla interconectarea protocoalelor ca ARPANET Telnet sau AUTODIN II THP la TCP.

2.6 Comunicatie de incredere

Un flux de date trimis printr-o conexiune TCP este livrat in siguranta si in ordine la destinatie. Transmisia se face in sigutanta prin intermediul seventelor de numere si a confirmarii primirii lor. Conceptual, fiecarui octet de date i se asigneaza un numar. Numarul asignat primului octet de date dintr-un segment este transmis cu acest segment si se numeste numarul de secventa segment. Segmentele transmit de asemenea un numar de confirmare care este numarul secventa a urmatorului octet de date trimis in directia inversa. Cand TCP-ul trimite un segment care contine date el pune o copie intr-o coada de retransmitere si porneste un ceas; cand confirmarea asupra datei e primita,

segmentul este sters din coada. Daca aceasta confirmare nu e primita inainte ca ceasul sa se opreasca, segmentul este retransmis.

O confirmare din partea TCP nu garanteaza ca datele eu fost primite de destinatar, ci doar ca TCP-ul destinatar a primit insarcinarea sa transmita datele.

Pentru a manipula fluxul de date intre protocoalele TCP , este intrebuintat un mecanism de control a fluxului. TCP-ul destinatie raporteaza o "fereastră" TCP-ului expeditor. Aceasta fereastră specifica numarul de octeti incepand cu numarul de confirmare pe care TCP-ul destinatie este gata sa il primeasca .

2.7 Stabilirea conexiunii si eliberarea conexiunii

Pentru a identifica fluxurile de date separate pe care TCP-ul le manipuleaza, TCP-ul ofera un identificator de port. Deoarece identificatorii de port sunt selectati independent de fiecare TCP este posibil ca ei sa nu fie unici. Pentru a oferi adrese unice in cadrul fiecarui TCP, vom concatena adresa de internet care identifica TCP-ul cu un identificator de port pentru a crea un socket care va fi unic in intreaga retea.

O conexiune este specificata de o pereche de socketuri la capete. Un socket local poate participa in mai multe conexiuni cu diferite socketuri "straine". O conexiune poate transmite date in ambele directii. Ea se numeste "full duplex".

TCP-urile sunt libere sa asocieze porturi cu procese in ordinea in care doresc. Totusi se impun cateva concepte de baza in fiecare implementare. Trebuie sa existe socketuri bine cunoscute pe care TCPul le asociaza doar proceselor "potrivite" unor scopuri. Procesele pot chiar "detine" propriile lor porturi. (Mijloacele de implementare a detinerii constituie o problema locala, dar ne putem imagina un Port de Cerere de tip comanda utilizator, sau o metoda de alocare unica a unui grup de porturi unui anumit proces, adica posibilitatea de a asocia bitii semnificativi ai unui port cu un anumit proces dat.).

O conexiune este specificata in apelul OPEN de portul local si argumentele socket straine. In schimb, TCP-ul specifica un scurt nume de conexiune locala prin care utilizatorul se poate referi prin apeluri. Sunt cateva lucruri care trebuie amintite in cadrul unei conexiuni. Pentru a stoca o informatie ne imaginam o structura de date numita Transmission Control Block (TCB). O strategie de implementare presupune ca numele conexiunii sa fie un pointer catre TCB. Apelul functiei OPEN specifica daca se urmareste stabilirea activa a conexiunii sau una pasiva.

Un apel pasiv al functiei OPEN semnifica faptul ca procesul doreste mai degraba sa accepte cereri de conexiune decat sa initializeze o conexiune. Deseori procesul care face o cerere OPEN pasiva va accepta o conexiune cu orice proces apelant. In acest caz un socket ce are bitii pe 0 este utilizat pentru a specifica un socket nespecificat. Socketurile straine nespecificate sunt permise doar in apeluri OPEN pasive.

Un proces serviciu care doreste sa specifice servicii catre alte procese necunoscute va face un apel OPEN pasiv catre acel socket strain necunoscut. Ar fi de ajutor daca acest socket ar fi asociat cu serviciul respectiv.

Socketurile cunoscute constituie un mecanism convenabil pentru asocierea a priori a adresei socketului cu un serviciu standard. De exemplu, Serverul Telnet este permanent asignat unui anumit socket, in timp ce alte socketuri sunt asociate pentru procese care se ocupa cu transferul de fisiere(File Transfer), Remote Job Entry, Text Generator, Echoer si Sink (ultimele 3 sunt folosite pentru testare). O adresa de socket poate fi rezervata pentru a accesa serviciul "Look-Up" care va returna un anumit socket caruia i se va oferi un serviciu nou creat. Conceptul de socket cunoscut este o parte a spcificatiei TCP, dar asignarea de servicii socketurilor nu face parte din aceasta specificatie.

Procesele pot emite apeluri OPEN pasive si pot astepta apeluri OPEN active din parte altor procese, fiind informate de TCP cand are loc stabilirea legaturii. Doua procese care emit apeluri OPEN active unul spre celalalt in acelasi timp se vor conecta in mod corect. Aceasta flexibilitate este esentiala in calculul distribuit(distributed computing) in care componentele se comporta asincron.

Exista doua cazuri pricipale in potrivirea socketurilor in privinta apelurilor OPEN pasive locale cat si a apelurilor OPEN active straine. In primul caz, apelul OPEN pasiv local specifica in intregime socketul strain. In acest caz, potrivirea trebuie sa fie exacta. In al doilea caz, apelurile OPEN pasive locale nu specifica socketurile straine. In acest caz orice socket strain este acceptat atata timp cat se potriveste cu socketul local. Alte posibilitati includ potriviri parțiale restrictionate.

Daca au loc apeluri de OPEN pasive(inregistrate in TCB) in acelasi timp cu apelul socketului local, un OPEN activ strain va fi potrivit cu un TCB cu socketul strain specificat in OPEN-ul activ strain.

Procedurile care stabilesc conexiunile folosesc flag-ul de control synchronize (SYN) si implica schimbul de trei mesaje. Acest schimb a fost numit strangere de mana in trei (three-way hand shake) [3].

O conexiune este initiata de intalnirea unui segment continand un SYN si o intrare TCB in asteptare, fiind fiecare creata de o comanda OPEN a utilizatorului. Potrivirea socketurilor locale si straine determina cand o conexiune a fost initiata. Conexiunea e stabilita cand o secventa de numere au fost sincronizate in ambele directii.

Eliberarea unei conexiuni implica de asemenea schimbul de segmente, care poarta in acest caz flagul de control FIN.

2.8 Comunicarea de date

Datele care circula printr-o conexiune pot fi gandite ca un flux de octeti. Utilizatorul expeditor specifica in fiecare apel SEND daca datele trimise de el prin acest apel(sau oricare alt apel) ar trebui trimise imediat la destinatar setand pe true flagul PUSH.

Un TCP expeditor are voie sa colecteze date de la utilizatorul expeditor si sa transmita date la propria-i alegere, pana cand se apeleaza functia de push; in acest moment TCP-ul trebuie sa transmita toate datele netransmise inca. Cand un

TCP destinatar vede flagul PUSH nu trebuie sa mai astepte date de la TCP-ul expeditor inainte de a pasa datele catre procesul destinatar.

Nu exista o legatura necesara intre functiile de push si limitele segmentelor. Datele din fiecare segment pot fi obtinute printr-un singur apel SEND sau prin mai multe apeluri SEND.

Scopul functiei push si a flagului PUSH este de a transmite date de la utilizatorul expeditor la cel destinatar. Dar nu ofera un serviciu de inregistrari.

Intre functia push si utilizarea de buffere de date care intersecteaza interfata TCP/utilizator exista o cuplare(mufare). De fiecare data cand un flag PUSH este asociat cu o serie de date plasate in bufferul utilizatorului destinatar, bufferul este returnat catre utilizator pentru procesare chiar daca acesta nu s-a umplut. Daca in schimb datele umplu bufferul destinatarului inainte ca sa fie apelata functia PUSH, datele sunt transmise catre utilizator in unitati de dimensiunea bufferului.

TCP-ul ofera de asemenea mijloace de a comunica destinatarului ca intr-un punct ulterior fata de citirea datelor prezente se afla date esentiale . TCP nu incearca sa defineasca ce face utilizatorul atunci cand este instiintat asupra datelor urgente(esentiale) pe care urmeaza sa le citeasca, dar ideea generala e aceea ca procesul destinatar va lua masuri in vederea procesarii rapide a datelor esentiale.

2.9 Precedenta si securitate

TCP-ul face uz de campurile de servicii si de optiunile de securitate ale protocolul internet pentru a oferi precedenta si securitate per conexiune tutuor utilizatorilor TCP. Nu toate modulele TCP vor functiona neaparat intr-un mediu multinivel sigur; cateva pot fi limitate doar la o utilizare nesecreta, in timp ce altele pot opera doar intr-un singur compartiment la un singur nivel de securitate. Prin urmare cateva implementari si servicii TCP catre utilizatori vor fi limitate la o submultime a cazului securitatii multinivel.

Modulele TCP care opereaza intr-un mediu multinivel sigur trebuie neaparat sa marcheze segmentele ce vor fi transmise in afara din punct de vedere al securitatii, compartimentarii si precedentei. Aceste module TCP trebuie de asemenea sa ofere o interfata utilizatorilor sau protoalelor de nivel superior ca Telnet sau THP pentru a le permite acestora sa specifice nielul de securitate, compartimentare si precedenta a conexiunilor.

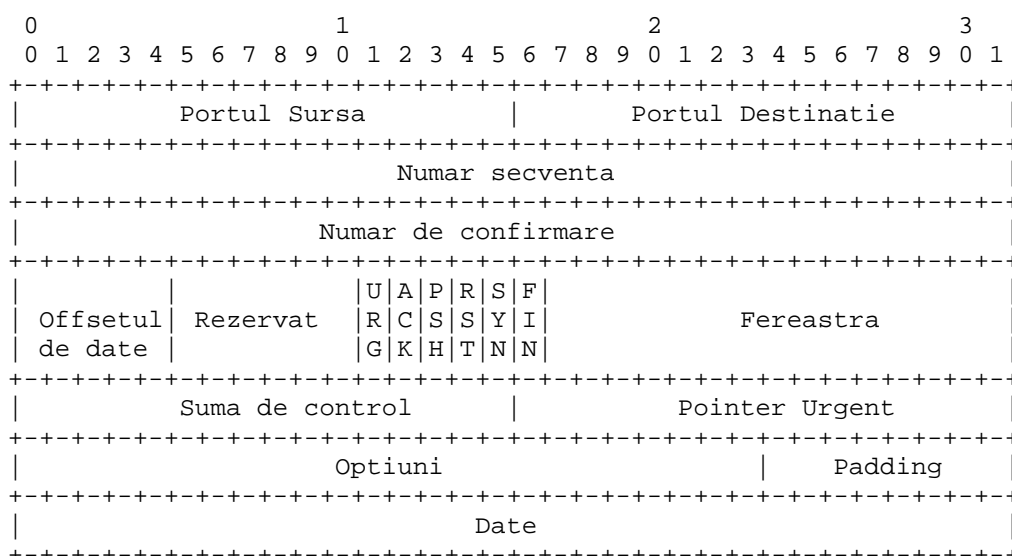
2.10 Principiul robustetii

Implementarile TCP vor urma un principiu general al robusteii: fii conservator in ceea ce faci si liberal in ceea ce accepti de la altii.

3. SPECIFICATII FUNCTIONALE

3.1 Formatul Headerului(antetului)

Segmentele TCP sunt trimise ca datagrame internet. Headerul Protocolului Internet poarta cateva campuri de informatii , incluzand adresele calculatorului sursa si destinatie. Un header TCP urmeaza headerului internet oferind informatii specifice protocolului TCP. Aceasta diviziune permite existenta protocoalelor de nivel altele decat TCP.



Formatul antetului TCP

Notam ca un insemn reprezinta o pozitie de un bit

Figura 3.

Portul sursa: 16 biti
 Numarul portului sursa.

Portul destinatie: 16 biti
 Numarul portului destinatie.

Numarul secventa: 32 biti
 Numarul secventa reprezinta primul octet de date din acest segment(exceptand cazul cand e prezent SYN). Daca SYN este prezent atunci numarul secventa este numarul secventa initial (ISN) si primul octet de date este ISN+1.

Numarul de confirmare: 32 biti
 Daca bitul de control ACK este setat acest camp va contine urmatorul numar secventa pe care expeditorul segmentului de date se asteapta sa-l primeasca. Odata ce se stabileste o conexiune, numarul de confirmare este intodeauna trimis.

Offsetul data: 4 biti

Numarul de cuvinte de 32 de biti din antetul TCP. Indica locul de unde incep datele. Antetul TCP (chiar si cel care include optiuni) este un numar de 32 biti.

Rezervat: 6 biti

Rezervat pentru o viitoare utilizare. Trebuie sa fie 0.

Bitii de control : 6 biti (de la stanga la dreapta)

URG: un pointer la un pachet de date urgente

ACK: un numar corect de confirmare

PSH: functia Push

RST: Reseteaza conexiunea

SYN: sincronizeaza numerele de secventa

FIN: semnifica inchiderea conexiunii(nu mai sunt date de transmis)

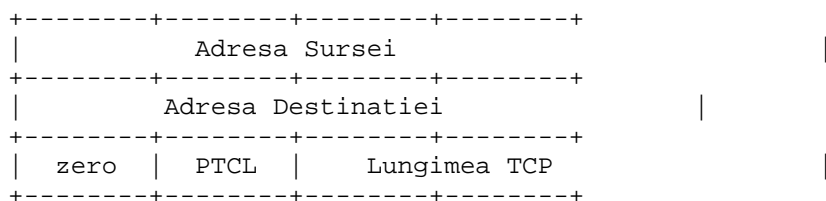
Fereastra: 16 biti

Numarul de octeti de date incepand cu cel indicat in campul de confirmare pe care expeditorul segmentului este gata sa-l accepte.

Suma de control: 16 biti

Campul sumei de control este complementul de 16 biti a sumei complement a tuturor cuvintelor de 16 biti din header si din text. Daca un segment contine un numar impar de octeti de antet si text care trebuie insumati , ultimul octet este completat la dreapta cu zerouri pentru a alcatui un cuvint de 16 biti in scopul realizarii sumei de control. Zerourile adaugate nu sunt transmise insa ca parte a segmentului. In timpul calcularii sumei de control, campul sumei de control este inlocuit cu valoarea 0.

Suma de control acopera de asemenea un pseudo antet de 96 de biti care prefixeaza antetul TCP. Acest pseudo antet contine adresa sursa, adresa destinatie , protocolul si lungimea TCP-ului. Acesta furnizeaza TCP-ului protectie impotriva rutarii eronate a segmentelor. Aceasta informatie este transportata de Protocolul Internet si e transferata in interfata TCP/Retea prin intermediul argumentelor sau a rezultatelor apelului TCP asupra IP-ului.



Lungimea TCP-ului este lungimea antetului TCP plus lungimea datelor in octeti(aceasta nu e o cantitate transmisa explicit ci e calculata); nu fac parte cei 12 octeti ai pseudo antetului.

Pointer de urgenta: 16 biti

Acest camp comunica valoarea curenta a pointerului de urgenta ca un offset pozitiv al numarului secventa din acest segment. Pointerul urgent pointeaza spre numarul secventa din octet fiind positionat dupa pachetele de date urgente.

Acset camp este interpretat in segmente prin setarea pe true a bitului de control URG.

Optiuni: variabile

Optiunile pot ocupa de la sfarsitul antetului TCP si au o lungime multiplu de 8 biti. Toate optiunile sunt incluse in suma de control. O optiune poate incepe la sfarsitul oricarui octet. Sunt doua tipuri de formate de optiuni:

Tip 1: Un singur octet de optiune a tipului.

Tip 2: Un octet de optiune a tipului, un octet de optiune a lungimii octetii de optiune a datelor.

Optiunea lungimii numara cei doi octeti ai a optiunii de tip si ai optiunii de date.

Notam faptul ca lista de optiuni poate mai scurta decat campul offsetului data. Continutul antetului dicolo de optiunea Sfarsit-de-Optiune trebuie sa fie o completare a antetului.

TCP-ul trebuie sa implementeze toate optiunile.

Optiunile definite in mod curent includ(tipul e indicat in octal):

Tip	Lungime	Explicatie
0	-	Sfarsitul listei de optiuni.
1	-	Nici o operatie.
2	4	Dimensiunea maxima a segmentului.

Definitii ale optiunilor specifice

Lista de Sfarsit-de-Optiune

```
+-----+
|00000000 |
+-----+
tip=0
```

Acest cod optiune indica sfarsitul listei de optiuni. Aceasta nu inseamna neaparat ca sfarsitul listei de optiuni coincide cu sfarsitul antetului TCP conform campului de offset de date. Acest cod este folosit la sfarsitul tuturor optiunilor, nu la sfarsitul fiecarei optiuni in parte si este folosit dar atunci cand sfarsitul listei de optiuni nu coincide cu sfarsitul antetului TCP.

Nici o operatie

```
+-----+
|00000001 |
+-----+
tip=1
```

Acest cod optiune poate fi folosit intre optiuni, de exemplu, pentru a alinia inceputul unei optiuni ulterioare la marginea unui cuvnt. Nu exista nici o garantie ca expeditorii vor folosi aceasta optiune, deci destinatarii trebuie sa fie pregatiti sa proceseze optiunile chiar daca ele nu incep la marginea unui cuvnt.

Dimensiunea maxima a segmentului

```

+-----+-----+-----+-----+
|00000010|00000100|   dim max seg   |
+-----+-----+-----+-----+
tip=2   Lungime=4

```

Optiunea de Date a segmentului de dimensiune maxima : 16 biti

Daca aceasta optiune e prezenta , atunci ea comunica dimensiunea maxima a segmentului primit de TCP-ul care trimite acest segment. Acest camp trebuie trimis in cererea initiala de conexiune(adica in segmentele cu bitul SYN setat). Daca aceasta optiune nu e folosita atunci segmentului ii este permisa orice dimensiune.

Completare: variabila

Completarea antetului este folosita pentru a se asigura ca antetul TCP se termina si datele incep la limita de 32 biti. Completarea se face cu zerouri.

3.2 Terminologie

Inainte de a putea discuta despre operatiile TCP trebuie sa introducem o terminologie detaliata. Pentru a intretine conexiunea TCP trebuie sa tinem minte cateva variabile. Concepem aceste variabile ca fiind stocate intr-o inregistrare conexiune numita Transmission Control Block sau TCB. Printre variabilele stocate in TCB se afla si numere de socketuri locale si straine, securitatea si precedenta conexiunii, pointere la bufferele sursa si destinatie ale utilizatorului, pointere catre coada de retransmitere si catre segmentul curent. Ca o completare, cateva variabile care sunt in legatura cu transmiterea si primirea numerelor de secventa sunt stocate de asemenea in TCB.

Variabile de secventa de trimitere (Send)

SND.UNA - send unacknowledged- trimite fara confirmare

SND.NXT - send next- trimite urmatorul

SND.WND - send window- trimite fereastra

SND.UP - send urgent pointer- trimite pointer urgent

SND.WL1 - segment sequence number used for last window update- numar de secventa segment folosit pentru ultima actualizare a ferestrei

SND.WL2 - segment acknowledgment number used for last window update- numar de confirmare segment folosit pentru ultima actualizare a ferestrei

ISS - initial send sequence number- numarul de secventa de trimitere initiala

Variabile de secventa de primire(Receive)

RCV.NXT - receive next- primeste urmatorul

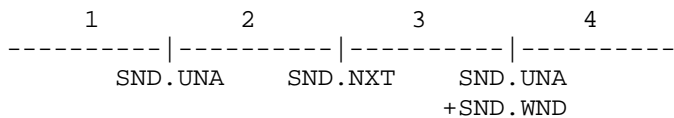
RCV.WND - receive window- primeste fereastra

RCV.UP - receive urgent pointer- primeste pointerul urgent

IRS - initial receive sequence number- numarul de secventa de primire initiala

Urmatoarele diagrame ajuta la legarea catorva din aceste variabile la spatiul de secventa.

Spatiul de secventa de trimitere



1 - numere de secventa vechi care au fost confirmate

2 - numere de secventa cu date neconfirmate

3 - numere de secventa permise pentru transmiterea de date noi

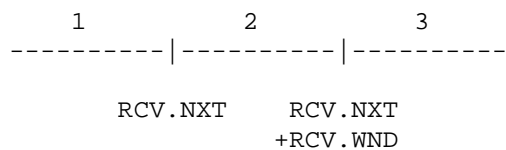
4 - numere de secventa viitoare care inca nu sunt permise

Spatiul de secventa de trimitere

Figura 4.

Fereastra expeditor este portiunea din spatiul de secventa etichetat cu 3 in figura 4.

Spatiul de secventa de primire



1 - numere de secventa vechi care au fost confirmate

2 - numere de secventa permise pentru primiri de date noi

3 - numere de secventa viitoare care inca nu sunt permise

Spatiul de secventa de primire

Figura 5.

Fereastra destinatar este portiunea din spatiul de secventa etichetat cu 2 in figura 5.

Sunt de asemenea cateva variabile folosite in mod frecvent in discutii care isi iau valorile din campurile segmentului curent.

Variabilele de segment curente

SEG.SEQ - segment sequence number - numarul de secventa al segmentului

SEG.ACK - segment acknowledgment number - numarul de confirmare al segmentului

SEG.LEN - segment length - lungimea segmentului

SEG.WND - segment window - fereastra segmentului

SEG.UP - segment urgent pointer - pointerul urgent al segmentului

SEG.PRC - segment precedence value - valoare de precedenta a segmentului

O conexiune trece printr-o serie de stari in timpul duratei sale de viata. Aceste stari sunt: LISTEN(Asculta), SYN_SENT, SYN_RECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT si starea fictiva CLOSED. CLOSED este fictiva pentru ca reprezinta starea in care nu exista nici un TCB, si deci nici o conexiune. Pe scurt, sensul acestor stari este:

LISTEN - reprezinta asteptarea unei cereri de conexiune din partea unui TCP sau port indepartat

SYN-SENT - reprezinta asteptarea unei cereri de conexiune potrivita dupa ce a trimis o cerere de conexiune

SYN-RECEIVED - reprezinta asteptarea unei cereri de confirmare a conexiunii dupa ce a trimis si a primit o cerere de conexiune

ESTABLISHED - reprezinta o conexiune deschisa, datele primite pot fi transmise utilizatorului. Este starea normala pentru faza de transfer de date in conexiune.

FIN-WAIT-1 - reprezinta asteptarea unei cereri de terminare de la TCP-ul indepartat, sau o confirmare a cererii de terminare a conexiunii trimise anterior.

FIN-WAIT-2 - reprezinta asteptarea unei cereri de terminare a conexiunii din partea TCP-ului indepartat.

CLOSE-WAIT - reprezinta asteptarea unei cereri de terminare a conexiunii din partea utilizatorului local.

CLOSING - reprezinta asteptarea unei confirmari a cererii de terminare a conexiunii din partea TCP-ului indepartat.

LAST-ACK - reprezinta asteptarea unei confirmari a cererii de terminare a conexiunii trimisa anterior TCP-ului indepartat(include si confirmarea cererii de confirmare a propriei conexiuni).

TIME-WAIT - reprezinta asteptarea unui anumit timp pentru a fi sigur faptul ca TCP-ul indepartat a primit confirmarea cererii de terminare a conexiunii .

SND.UNA = cel mai vechi numar de secventa oldest neconfirmat

SND.NXT = urmatorul numar de secventa care va fi trimis

SEG.ACK = confirmare din partea TCP-ului destinatar(urmatorul numar de secventa asteptat de TCP-ul destinatar)

SEG.SEQ = primul numar de secventa al unui segment

SEG.LEN = numarul de octeti ocupati de datele din segment (in acestia intra si SYN si FIN)

SEG.SEQ+SEG.LEN-1 = ultimul numar de secventa intr-un segment

O noua confirmare (numita "confirmare acceptabila") este aceea pentru care e valabila urmatoarea inegalitate:

$$\text{SND.UNA} < \text{SEG.ACK} \leq \text{SND.NXT}$$

Un segment din coada de retransmitere este confirmat in intregime daca suma dintre numarul sau de secventa si lungimea sa este mai mica sau egala decat valoarea de onfirmare a segmentului ce urmeaza sa fie primit.

Cand se primesc date este nevoie de urmatoarele comparatii:

RCV.NXT = urmatorul numar de secventa asteptat cu un segment ce urmeaza sa fie primit; reprezinta marginea din stanga sau marginea inferioara a ferestrei destinatie

RCV.NXT+RCV.WND-1 = ultimul numar de secventa asteptat cu un segment ce urmeaza sa fie primit; reprezinta marginea din dreapta sau marginea superioara a ferestrei destinatie

SEG.SEQ = primul numar de secventa ocupat de segmentul ce urmeaza sa fie primit

SEG.SEQ+SEG.LEN-1 = primul numar de secventa ocupat de segmentul ce urmeaza sa fie primit

Un segment ocupa un spatiu de secventa de primire valid daca

$$\text{RCV.NXT} \leq \text{SEG.SEQ} < \text{RCV.NXT} + \text{RCV.WND}$$

sau daca

$$\text{RCV.NXT} \leq \text{SEG.SEQ} + \text{SEG.LEN} - 1 < \text{RCV.NXT} + \text{RCV.WND}$$

Prima parte a acestui test verifica daca inceputul segmentului se intrepatrunde cu fereastra ; a doua parte a testului verifica daca sfarsitul segmentului se intrepatrunde cu fereastra; daca inceputul sau sfarsitul segmentului se intrepatrunde cu fereastra spunem ca el contine date din fereastra.

De fapt, este putin mai complicat. Datorita segmentelor si ferestrelor de lungime zero, avem 4 cazuri de a accepta un segment ce urmeaza sa fie primit:

Lungimea Segmentului	Fereastra destinatie	Test
-----	-----	-----
0	0	SEG.SEQ = RCV.NXT
0	>0	RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND
>0	0	not acceptable
>0	>0	RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND or RCV.NXT =< SEG.SEQ+SEG.LEN-1 < RCV.NXT+RCV.WND

De notat ca atunci cand fereastra destinatie este zero nu exista segmente acceptabile exceptie facand segmentele ACK. Astfel, este posibil pentru TCP sa mentina o fereastra destinatie zero in timp ce transmite date si primeste confirmari(ACK). Totusi, chiar daca fereastra destinatie este zero, TCP trebuie sa proceseze campurile RST si URG pentru toate segmentele ce urmeaza sa fie primite.

Am profitat si de schema de numerotare pentru a proteja anumite informatii de control. Acest lucru e realizat prin includerea implicita a unor flaguri de control in spatiul de secventa deci aceste informatii pot fi retransmise sau confirmate fara a se face confuzii (se va actiona la un moment dat numai asupra unei copii de control). Informatiile de control nu sunt transportate fizic in spatiul de date al segmentului. Drept consecinta, trebuie sa adoptam o serie de reguli pentru a putea asigna implicit controale numerelor de secventa. SYN si FIN sunt singurele controale care necesita astfel de protectie si sunt folosite la deschiderea sau inchiderea unei conexiuni. In privinta numerelor de secventa, se considera ca SYN apare inaintea primului octet de date efectiv din segmentul in care se afla, iar FIN apare dupa ultimul octet de date efectiv in segmentul in care se afla. Lungimea segmentului(SEG.LEN) include controale ce se ocupa atat cu date cat si cu spatiul de secventa. Cand un SYN este prezent, SEG.SEQ este numarul de secventa al lui SYN.

Selectia initiala a numerelor de secventa

Protocolul nu impune nici o restrictie pe o conexiune particulara care este folosita de mai multe ori incontinuu. O conexiune este definita de o pereche de socketuri. Noi instante de conexiuni vor fi privite ca intrupari a conexiunii. Problema care se ridica este --" cum identifica TCP-ul segmentele care se repeta in intrupari(instantieri) anterioare ale conexiunii?". Aceasta problema devine aparenta daca o conexiune este deschisa si inchisa intr-o succesiune rapida., sau daca o conexiune se intrerupe cu pierderi de memorie si apoi e restabilita.

Pentru a evita confuziile trebuie sa prevenim utilizarea unor segmente care apartin unei instante a unei conexiuni in timp ce aceleasi numere de secventa sunt inca prezente in retea din instante anterioare. Vrem sa ne asiguram de acest lucru chiar daca TCP-ul pica si pierde toate cunostintele despre numerele de secventa cu care lucra. Cand noi conexiuni sunt create, este intrebuintat un generator de numar de secventa initial (ISN) care selecteaza un nou iSN de 32 de biti. Acest generator este legat de un ceas de 32 biti (posibil fictiv), a carui bit inferior este incrementat la fiecare 4 microsecunde. Astfel, ISN-ul cicleaza

la fiecare 4.55 ore. Deoarece ne asiguram ca segmentele raman in retea nu mai mult de timpul de viata maxima a unui segment(MSL) si ca acest MSL este mai mic de 4.55 putem pe buna dreptate sa presupunem ca ISN-ul va fi unic.

Pentru fiecare conexiune exista un numar de secventa de trimitere si unul de primire. Numarul de secventa de trimitere initial (ISS) este ales de TCP-ul ce transmite datele, iar numarul de secventa de primire initial (IRS) este retinut in timpul procedurii de stabilire a conexiunii.

Pentru ca o conexiune sa fie stabilita sau initializata, cele doua TCP-uri trebuie sa-si sincronizeze numerele de secventa initiale. Acest lucru se realizeaza intr-un schimb de segmente ce stabilesc conexiunea care poarta un bit de control numit "SYN" (pentru sincronizare) si numerele de secventa initiale. O alta denumire pentru segmentele care transporta bitul de control SYN este "SYNs". De acum incolo solutia necesita un mecanism potrivit de alegere a numarului de secventa initial si o usoara comunicare pentru a face schimb de ISN-uri.

Sincronizarea necesita ca ambele parti sa-si trimita unul altuia numarul de secventa initial si sa primeasca o confirmare a acestuia ca o recunoastere din partea cealalta. De asemenea fiecare parte trebuie sa primeasca numarul de secventa initial din cealalta parte si sa trimita o confirmare a recunoasterii.

- 1) A --> B SYN numarul meu de secventa este X
- 2) A <-- B ACK numarul tau de secventa este X
- 3) A <-- B SYN numarul meu de secventa este Y
- 4) A --> B ACK numarul tau de secventa este Y

Specificatii functionale

Deoarece pasii 2 si 3 pot fi combinati intr-un singur mesaj, aceasta schema se mai numeste three way handshake (negociere in trei pasi).

Aceasta este necesara pentru ca numerele de secventa nu sunt legate de un ceas global in retea, iar TCP-urile pot avea mecanisme siferite de alegere a ISN-urilor. Destinatarul primului ISN nu poate sti daca segmentul pe care urmeaza sa-l primeasca este unul intarziat sau nu, decat daca tine minte ultimul numar de secventa utilizat in conexiune(lucru care nu e intodeaua posibil), deci trebuie sa cerem expeditorului sa verifice acest SYN. Strangerea de mana in trei si avantajele unei scheme cu ceas sunt dscutate in partea [3].

Stiinta de a sti sa taci cand trebuie

Pentru a fi siguri ca TCP-ul nu creeaza un segment care sa transporte un numar de secventa care ar putea fi duplicat de un segment mai vechi aflat in retea TCP-ul trebuie sa stea in asteptare pentru o perioada egala cu durata maxima de viata a unui segment (MSL) inainte de a asigna un numar de secventa la pornirea sau la recuperarea dupa o eroare de memorie in care numerele de secventa aflate in lucru au fost pierdute. In aceasta specificatie MSL este considerat ca avand 2 minute. Aceasta alegere depinde de ingineri, dar poate fi schimbata daca experienta dovedeste ca e bine sa o schimbam. De notat ca daca TCP-ul este reinitializat in vreun sens, retine totusi in memorie numerele de secventa utilizate, de aceea nu necesita nici un timp de asteptare; trebuie doar sa foloseasca numere de secventa mai mari decat cele pe care tocmai le-a folosit.

Conceptul TCP de asteptare in liniste

Aceasta specificatie ofera pentru gazdele care "pica" fara a retine cunostinte despre ultimele numere de secventa transmise in conexiunile active(adica cele care nu sunt inchise) o perioada de intarziere a emiterii de segmente TCP pentru macar o perioada de timp egala cu MSL in sistemul internet din care gazda face parte. In paragrafele de mai jos se va da o explicatie a acestei specificatii. Cei care implementeaza TCP-ul pot incalca restrictia de "asteptare in liniste", dar cu riscul ca date vechi sa fie acceptate ca noi si datele noi sa fie respinse fiindca sunt considerate ca fiind vechi.

TCP-ul consuma spatiu de numar de secventa de fiecare data cand se formeaza un segment si acesta intra in coada de iesire a retelei a unei gazde sursa. Algoritmul de detectare a duplicarii si a secventialitatii in protocolul TCP se bazeaza pe legatura unica dintre segmentul de date si spatiul de secventa in masura in care numerele de secventa nu vor cicla prin toate cele 2^{32} valori inainte ca legatura segmentului de date cu numerele de secventa sa fie trimisa si confirmata de destinatar si toate copiile segmentelor duplicate sa se "scurga" afara din internet. Fara aceasta presupunere la doua TCP-uri distincte s-ar putea asigna secvente de numere identice sau suprapuse, creand confuzie la destinatar care nu va mai sti care date sunt noi si care sunt vechi. Va amintim ca fiecare segment este legat la atatea numere de secventa consecutive cati octeti de date sunt in segment.

In conditii normale, TCP-urile tin evidenta urmatorului numar de secventa si a celei mai vechi confirmari pentru a evita folosirea eronata a unui numar de secventa inainte ca prima sa utilizare sa fie confirmata. Acest lucru insa nu e suficient pentru a se asigura ca duplicatele vechi de date se scurg afara din retea, asa ca spatiul de secventa a fost facut foarte mare pentru a reduce posibilitatea ca un duplicat ratacit sa produca probleme la sosire. Cu 2 megabiti/sec timpul in care se vor folosi pana la 2^{32} octeti ai spatiului de secventa este de 4,5 ore. Stiind ca timpul maxim de viata al unui segment in retea nu depaseste cateva zeci de secunde, acest lucru este de o ampla protectie pentru retele cu anticipatie, chiar daca frecventa de trimitere a datelor creste la 10 megabiti/sec. La 100 megabiti/sec, timpul de ciclare este scurt, ??dar totusi just??.

Algoritmul de baza de detectare a duplicarii si secventialitatii in TCP poate fi totusi inselat, daca un TCP sursa nu tine minte numerele de secventa pe care le-a folosit intr-o conexiune. De exemplu, daca TCP-ul ar initializa toate conexiunile cu numerele de secventa 0, atunci in afara de a pica si a se restarta, TCP-ul ar putea stabili o conexiune anterioara(posibil dupa o conexiune deschisa pe jumătate) si sa emita pachete cu numere de secventa identice sau suprapuse cu pachetele aflate inca in retea care au fost trimise de o instanta anterioara a aceleiasi conexiuni. In cazul in care TCP-ul nu tine minte numerele de secventa folosite intr-o anumita conexiune, Specificatiile TCP recomanda ca sursa sa intarzie un timp egal cu MSL inainte ca sa transmita segmente intr-o conexiune pentru a da timp segmentelor dintr-o instanta anterioara a conexiunii sa se scurga afara din sistem.

Chiar calculatoarele gazda care pot memora ora si o folosesc pentru a selecta numere de secventa initiale nu sunt imune la aceasta problema(chiar daca ora din zi e folosita pentru a selecta un numar de secventa initial pentru fiecare noua instanta a conexiunii).

Sa presupunem, de exemplu, ca o conexiune e deschisa si numarul se initializeaza cu S. Sa presupunem de asemenea ca aceasta conexiune nu e utilizata mult timp si ca pana la urma functia numarului de secventa initial ($ISN(t)$) ia o valoarea egala cu cea a numarului de secventa, sa spunem S1, a ultimului segment trimis de TCP intr-o anumita conexiune. Sa presupunem ca in acest moment gazda pica, apoi isi revine si stabeste o noua instanta a conexiunii. Numarul initial de secventa ales este $S1 = ISN(t)$ -- ultimul numar de secventa din instanta anterioara a conexiunii. Daca revenirea are loc suficient de repede, orice duplicate vechi din retea avand numere de secventa apropiate de S1 pot ajunge la destinatarul noii instante a conexiunii si pot fi tratate de acesta ca fiind pachete noi.

Problema e ca gazda care pica nu are cum sa stie cat timp a fost inchisa si nici daca mai sunt in sistem duplicate ale unei conexiuni anterioare.

O solutie la aceasta problema este intarzierea deliberata de transmitere a segmentelor pentru o perioada de timp egala cu MSL, dupa recuperarea de dupa o prabusire - aceasta se numeste specificatia "quite time". Cei care implementeaza pot oferi utilizatorilor TCP abilitatea de a alege, intr-un sistem de tip conexiune-conexiune, sa astepte dupa o prabusire, sau pot implementa neoficial solutia "quite time" pentru toate conexiunile. Evident, chiar atunci cand un utilizator alege sa astepte, acest lucru nu e necesar daca gazda si-a revenit de cel putin MSL secunde.

Ca un rezumat: fiecare segment emis ocupa unul sau mai multe numere de secventa din spatiul de secventa, numerele de secventa folosite de un segment sunt "ocupate" sau "in lucru" pana cand au trecut MSL secunde, dupa prabusire un bloc spatiu-timp este ocupat de ocupat de octetii ultimului segment trimis, daca o conexiune isi revine prea curand si foloseste numere de secventa din amprenta spatiu-timp a ultimului segment dintr-o instanta anterioara a conexiunii, este posibil sa existe o zona in care numerele de secventa se suprapun si produc confuzie la destinatar.

3.4 Stabilirea unei conexiuni

"Strangerea de mana in trei sensuri" este o procedura folosita pentru a stabili o conexiune. Aceasta este de obicei initiata de un TCP si ei ii raspunde celalalt TCP. Procedura functioneaza si daca ambele TCP-uri initiaza procedura simultan. Cand apare o astfel incercare simultana, fiecare TCP primeste un segment SYN care nu transporta nici o confirmare dupa ce a trimis SYN-ul. Desigur, aparitia unui segment SYN vechi duplicat poate semnala destinatarului ca are loc o initiere simultana a 2 conexiuni. Utilizarea corespunzatoare a segmentelor "reset" poate evita ambiguitatea in aceste cazuri.

Urmeaza cateva exemple de initializare simultana a unor conexiuni. Desi aceste exemple nu ilustreaza sincronizarea conexiunilor folosind segmente purtatoare de date, ele sunt complet justificate atata timp cat TCP-ul destinatar nu trimite datele utilizatorului pana nu este sigur ca sunt valide (adica datele trebuie pastrate intr-un buffer la destinatar pana cand conexiunea ajunge in starea ESTABLISHED - stabilita). Strangerea de mana in trei sensuri reduce posibilitatea conexiunilor false. Implementarea schimbului dintre memorie si mesaje ofera informatii pentru aceasta verificare.

Cea mai simpla reprezentare a "strangerii de mana in trei sensuri " e ilustrata in figura 7 de ma jos. Figura trebuie interpretata in felul urmator: Fiecare rand este numerotat pentru a putea face referinta la el. Sagetile dreapta (-->) indica plecarea unui segment TCP de la TCP-ul A la TCP-ul B, sau ajungerea unui segment la B pornit din A. Sagetile stanga (<-->) indica exact contrariul. Elipsa (...) indica un segment care se afla inca in retea(e intarziat). "XXX" indica un segment care e pierdut sau respins. Comentariile apar in paranteze. Starile TCP-ului reprezinta starea de dupa (AFTER) plecarea sau ajungerea segmentelor (al caror continut este ilustrat la mijocul fiecarui rand). Continutul segmentelor este prezentat intr-o forma prescurtata, cu numere de secventa, flaguri de control si campul ACK. Alte campuri ca fereastra, adrese, lungime si text au fost omise pentru claritate.

TCP A	TCP B
1. CLOSED	LISTEN
2. SYN-SENT --> <SEQ=100><CTL=SYN>	--> SYN-RECEIVED
3. ESTABLISHED <-- <SEQ=300><ACK=101><CTL=SYN,ACK>	<-- SYN-RECEIVED
4. ESTABLISHED --> <SEQ=101><ACK=301><CTL=ACK>	--> ESTABLISHED
5. ESTABLISHED --> <SEQ=101><ACK=301><CTL=ACK><DATA>	--> ESTABLISHED

Basic 3-Way Handshake for Connection Synchronization
Negocierea in trei pasi pentru conexiunea sincronizata

Figura 7.

In linia 2 din Figura 7 TCP-ul A incepe trimiterea unui segment SYN indicand faptul ca va folosi numerele de secventa care ince cu numarul initial de secventa 100. In linia 3, TCP-ul B trimite si el un SYN si confirma primirea SYN-ului de la A. Notam ca campul de confirmare indica faptul ca TCP-ul B este gata sa primeasca secventa 101, confirmand primirea SYN-ului care a ocupat secventa 100.

In linia 4, TCP-ul A raspunde cu un segment gol continand un ACK(o confirmare) pentru primirea SYN-ului de la TCP-ul B. Iar in linia 5 TCP-ul A trimite date. De remarcat ca numarul secventa din linia 5 e acelasi cu numarul secventa din linia 4 pentru ca ACK nu ocupa spatiu de numar de secventa (daca ar fi ocupat spatiu am fi ajuns sa facem confirmare la confirmari !).

Initializarea simultana este putin mai complicata, asa cum arat Figura 8. Fiecare TCP trece prin starile de la CLOSED la SYN-SENT la SYN-RECEIVED la Starea ESTABLISHED (adica stabilirea conexiunii).

TCP A	TCP B
1. CLOSED	CLOSED
2. SYN-SENT --> <SEQ=100><CTL=SYN>	...
3. SYN-RECEIVED <-- <SEQ=300><CTL=SYN>	<-- SYN-SENT

```

4.          ... <SEQ=100><CTL=SYN>          --> SYN-RECEIVED
5. SYN-RECEIVED --> <SEQ=100><ACK=301><CTL=SYN,ACK> ...
6. ESTABLISHED <-- <SEQ=300><ACK=101><CTL=SYN,ACK> <-- SYN-RECEIVED
7.          ... <SEQ=101><ACK=301><CTL=ACK>   --> ESTABLISHED

```

Sincronizarea conexiunii simultane

Figura 8.

Scopul principal al strangerii de mana in trei sensuri este de a preveni confuzia creata de initializarile de conexiuni duplicate vechi. Pentru aceasta s-a inventat un mesaj special de control numit reset. Daca TCP-ul destinatar este intr-o stare nesincronizata, (adica SYN-SENT, SYN-RECEIVED), el se intoarce la LISTEN pentru a primi un reset acceptabil. Daca TCP-ul este intr-una din starile sincronizate (ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT), intrerupe conexiunea si informeaza utilizatorul. Vom discuta acest ultim caz in conexiunile "pe jumătate deschise" de mai jos.

TCP A	TCP B
1. CLOSED	LISTEN
2. SYN-SENT --> <SEQ=100><CTL=SYN>	...
3. (duplicate) ... <SEQ=90><CTL=SYN>	--> SYN-RECEIVED
4. SYN-SENT <-- <SEQ=300><ACK=91><CTL=SYN,ACK>	<-- SYN-RECEIVED
5. SYN-SENT --> <SEQ=91><CTL=RST>	--> LISTEN
6. ... <SEQ=100><CTL=SYN>	--> SYN-RECEIVED
7. SYN-SENT <-- <SEQ=400><ACK=101><CTL=SYN,ACK>	<-- SYN-RECEIVED
8. ESTABLISHED --> <SEQ=101><ACK=401><CTL=ACK>	--> ESTABLISHED

Refacerea dupa un SYN vechi duplicat

Figura 9.

Consideram Figura 9 ca un exemplu simplu de refacere dupa duplicate vechi. La linia 3, un duplicat TCP vechi ajunge la TCP-ul B. TCP-ul B nu poate sti ca este un duplicat vechi si raspunde normal (linia 4). TCP-ul A detecteaza vede campul ACK ca fiind incorect si returneaza valoarea de reset RST impreuna cu campul sau SEQ pentru a face segmentul de incredere. TCP-ul B, la primirea lui RST, se intoarce la starea LISTEN. Cand SYN-ul original ajunge in cele din urma la linia 6, sincronizarea se desfasoara normal. Daca segmentul SYN de la linia 6 ar fi ajuns inaintea RST-ului, ar fi avut loc un schimb mai complex cu RST-uri trimise in ambele directii.

Conexiuni deschise pe jumătate și alte anomalii

O conexiune este stabilită pe jumătate dacă unul din TCP-uri a închis sau întrerupt conexiunea fără a înștiința celălalt TCP, sau dacă cele două capete ale conexiunii sau desincronizat datorită unei prabusiri din care a rezultat pierdere de memorie. Acest fel de conexiuni vor fi resetate automat dacă se va încerca trimiterea de date într-o direcție sau alta. Totuși, conexiunile deschise pe jumătate apar rar fiind folosite într-o oarecare măsură procedura de recuperare.

Dacă în punctul A conexiunea nu mai există, atunci o încercare din partea unui utilizator de a trimite date din punctul B va provoca primirea în TCP-ul destinat din punctul B a unui mesaj de control reset. Acest mesaj indică TCP-ului din punctul B că ceva este în neregulă și conexiunea este de așteptat să fie întreruptă.

Să presupunem că două procese utilizator A și B comunică unul cu celălalt și deodată conexiunea pica determinând pierdere de memorie TCP-ului A. În funcție de sistemul de operare care suportă TCP-ul A, este probabil să existe un mecanism de recuperare a datelor. Când TCP-ul își revine, A va porni din nou din punctul de început sau din punctul de revenire. Ca rezultat, A va încerca probabil să deschidă iar conexiunea apelând OPEN, sau va încerca să facă un SEND spre o conexiune pe care o crede deschisă. În ultimul caz, va primi un mesaj de eroare: "conexiunea nu este deschisă", apărut la TCP-ul A local. Într-o încercare de a stabili conexiunea, A va trimite un segment continuând SYN. După ce TCP-ul A pica, utilizatorul încearcă să redeschidă conexiunea. TCP-ul B, într-un timp, crede că conexiunea este deschisă.

TCP A	TCP B
1. (CRASH)	(send 300, receive 100)
2. CLOSED	ESTABLISHED
3. SYN-SENT --> <SEQ=400><CTL=SYN>	--> (??)
4. (!!)	<-- <SEQ=300><ACK=100><CTL=ACK>
5. SYN-SENT --> <SEQ=100><CTL=RST>	--> (Abort!!)
6. SYN-SENT	CLOSED
7. SYN-SENT --> <SEQ=400><CTL=SYN>	-->

Descoperirea unei conexiuni deschise pe jumătate

Figura 10.

Când ajunge SYN, la linia 3, TCB-ul B, fiind într-o stare sincronizată, iar segmentul ce urmează să ajungă în afara ferestrei, răspunde printr-o confirmare indicând că secvența se așteaptă să asculte (ACK 100). TCP-ul A vede că

segmentul nu confirma nimic din ce a trimis, si, fiind nesincronizat, trimite un reset(RST) pentru ca a detectat o conexiune pe jumătate deschisa. La linia 5, TCP-ul B intrerupe conexiunea. TCP-ul A va continua sa incerce sa restabileasca conexiunea; problema e acum redusa la strangerea de mana in trei sensuri din figura 7.

Un caz alternativ apare cand TCP-ul A pica si TCP-ul B incearca sa trimita date spre ceea ce crede el ca este o conexiune sincronizata. Acest lucru este ilustrat in figura 11. In acest caz, datele venite la TCP-ul A de la TCP-ul B (linia 2) nu sunt acceptate pentru ca nu exista nici o astfel de conexiune, deci TCP-ul A trimite un RST. RST-ul este acceptat, deci TCP-ul B il proceseaza si intrerupe conexiunea.

TCP A	TCP B
1. (CRASH)	(trimite 300,primeste 100)
2. (??) <-- <SEQ=300><ACK=100><DATA=10><CTL=ACK>	<-- ESTABLISHED
3. --> <SEQ=100><CTL=RST>	--> (Intrerupere!!)

Partea activa descopera o conexiune deschisa pe jumătate

Figura 11.

In figura 12, cele 2 TCP-uri A si B au conexiuni pasive asteptand SYN. Un duplicat vechi ajungand la TCP-ul B (linia 2) il pune pe B in actiune. Un SYN-ACK este returnat (linia 3) si determina TCP-ul A sa genereze un RST (ACK din linia 3 nu este acceptat). TCP-ul B accepta resetul si se intoarce in starea pasiva de LISTEN.

TCP A	TCP B
1. LISTEN	LISTEN
2. ... <SEQ=Z><CTL=SYN>	--> SYN-RECEIVED
3. (??) <-- <SEQ=X><ACK=Z+1><CTL=SYN,ACK>	<-- SYN-RECEIVED
4. --> <SEQ=Z+1><CTL=RST>	--> (se intoarce la LISTEN!)
5. LISTEN	LISTEN

Un SYN duplicat vechi initializeaza un reset la doua socketuri pasive

Figura 12.

Sunt posibile o multime de alte cazuri, toate fiind explicate de uratoarele reguli de generare si procesare a RST.

Generarea resetarii

Ca o regula generala, resetarea(RST) trebuie sa apara oricand ajunge un segment intr-o conexiune curenta in care n-ar trebui sa apara. Nu trebuie sa folosim resetarea daca nu suntem siguri ca aceasta este cauza.

Sunt trei grpe de stari:

1. Daca conexiunea nu exista (CLOSED), atunci e trimisa o resetare ca raspuns la orice alt segment cu exceptia unei alte resetari. Ca un caz particular, segmentele SYN care se adreseaza unei conexiuni ne-existente sunt respinse cu ajutorul resetarilor.

Daca segmentul care urmeaza sa soseasca are un camp ACK, resetarea isi ia numarul de secventa din campul ACK al segmentului, altfel, resetarea are numarul de ecventa zero si campul ACK este setat ca suma dintre numarul de secventa si lungimea segmentului ce soseste. Conexiunea ramnae in starea CLOSED.

2.Daca conexiunea este intr-una din starile nesincronizate (LISTEN, SYN-SENT, SYN-RECEIVED), si segmentul care soseste confirma ceva ce inca nu a fost trimis (segmentul poarta un ACK neacceptat), sau daca un segment ce soseste are un nivel de securitate sau un sector care nu se potriveste exact cu nivelul si sectorul cerut de conexiune, este trimisa o resetare.

Daca SYN-ul nu a fost confirmat si nivelul de precedenta a segmentului care soseste este mai mare decat nivelul de precedenta cerut, atunci fie se mareste nivelul de precedenta local(daca avem permisiunea de la utilizator sau de la sistem) sau se trimite o resetare; sau daca nivelul de precedenta a segmentului care soseste este mai mic decat nivelul de precedenta cerut, atunci se continua ca si cand nivelele de precedenta s-ar potrivi exact (daca TCP-ul indepartat nu poate ridica nivelul de precedenta pentru a se potrivi cu al nostru acest lucru va fi detectat in urmatorul segment pe care il trimite, si conexiunea va fi intrerupta chiar in acel moment). Daca SYN-ul nostru este confirmat(poate in segmentul care soseste), nivelul de precedenta al segmentului care soseste trebuie sa se potriveasca exact cu nivelul de precedenta local, daca nu se trimite o resetare.

Daca segmentul care soseste are un camp ACK, resetarea ia numarul de secventa din campul ACK al segmentului, altfel resetarea are numarul de secventa zero si campul ACK este setat cu sumadintre numarul de secventa si lungimea segmentului care soseste. Conexiunea ramane in aceeasi stare.

3. Daca conexiunea este intr-o stare sincronizata (ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT), orice segment neacceptat (in afara de numarul de secventa de fereastru sau de numarul de confirmare neacceptat) trebuie sa obtina doar un segment de confirmare gol continuand numarul de secventa de trimitere curent si o confirmare care sa indice urmatorul numar de secventa care urmeaza sa fie primit, iar conexiunea ramane in aceeasi stare.

Daca un segment care soseste are un nivel de securitate, sau un sector, sau o precedenta care nu se potriveste exact cu nivelul si cu sectorul si cu precedenta ceruta de conexiune, este trimisa o resetare si conexiunea trece in starea CLOSED. Resetarea isi ia numarul de secventa din campul ACK al segmentului ce soseste.

Procesarea resetarii

In toate stările cu excepția SYN-SENT, toate segmentele de resetare (RST) sunt validate verificându-le câmpurile SEQ. O resetare este validă dacă umarul sau de secvență este în fereastră. În starea SYN-SENT (un RST primit ca răspuns la un SYN inițial), RST este acceptat dacă SYN-ul este confirmat de câmpul ACK.

Destinatarul RST-ului mai întâi îl validează, apoi își schimbă starea. Dacă destinatarul se află în starea LISTEN, îl ignoră. Dacă destinatarul se află în starea SYN-RECEIVED și a fost înainte în starea LISTEN, atunci destinatarul revine în starea LISTEN, altfel destinatarul întrerupe conexiunea și trece în starea CLOSED. Dacă destinatarul se află în oricare altă stare, întrerupe conexiunea, avertizează utilizatorul și trece în starea CLOSED.

3.5. Închiderea unei conexiuni

CLOSE este o operație care semnifică: "Nu mai am date de trimis". Noțiunea de închidere a unei conexiuni full-duplex este subiectul unei interpretări ambigue, desigur, pentru că nu este prea clar cum trebuie tratată partea destinatarului a conexiunii. Am ales să tratăm CLOSE în manieră simplex. Utilizatorul care apelează CLOSE poate continua să PRIMEASCĂ (RECEIVE) până când este informat că și cealaltă parte s-a închis de asemenea. Deci, un program poate iniția câteva SEND-uri, urmate de un CLOSING, și apoi continua să RECEIVE (PRIMEASCĂ) până când e informat că un RECEIVE a eșuat din cauza că cealaltă parte s-a ÎNCHIS (CLOSED). Presupunem că un TCP va avertiza utilizatorul, chiar dacă RECEIVE-urile nu sunt deosebite, că cealaltă parte s-a închis, așa că utilizatorul își poate termina partea într-o manieră elegantă. Un TCP va livra în siguranță toate bufferele SEND înainte ca să se ÎNCHIDĂ (CLOSE) conexiunea, deci un utilizator care nu așteaptă să primească date înapoi are nevoie doar să primească confirmarea că s-a ÎNCHIS (CLOSED) conexiunea și că datele trimise de el au ajuns în întregime la TCP-ul destinatar. Utilizatorii trebuie să continue să citească conexiunile pe care închid pentru trimitere până când TCP-ul semnalează că nu a mai primit date.

Există trei cazuri esențiale:

1) Utilizatorul inițiază spunând TCP-ului să ÎNCHIDĂ conexiunea (să apeleze CLOSE)

2) TCP-ul îndepărtat inițiază trimițând un semnal de control FIN

Ambdoi utilizatorii fac CLOSE simultan.

Cazul 1:

În acest caz, un segment FIN poate fi construit și plasat în coada de plecare a segmentului. Nici un alt SEND de la utilizator nu va fi acceptat de TCP și intra în starea FIN-WAIT-1. RECEIVE-uri sunt acceptate în această stare. Toate segmentele care preced și includ FIN vor fi retransmise până vor fi confirmate. Când celălalt TCP a confirmat segmentul FIN și a trimis un FIN al său, primul TCP poate confirma acest FIN. De notat că un TCP care primește un FIN, va trimite un ACK dar nu va trimite propriul FIN până când utilizatorul sau a ÎNCHIS (CLOSED) de asemenea conexiunea.

Cazul 2: TCP-ul primește un FIN din rețea

Daca un FIN nesolicitat ajunge din retea, TCP-ul destinatari il poate confirma(ACK) si sa spuna utilizatorului ca se inchide conexiunea. Utilizatorul va raspunde cu un CLOSING, peste care TCP-ul poate trimite un FIN catre celalalt TCP dupa ce a terminat de trimis toate datele. Atunci TCP-ul asteapta pana cand propriul sau FIN este confirmat si imediat dupa aceea sterge conexiunea. Daca un ACK nu este la indemana dupa o pauza a utilizatorului, conexiunea e inchisa si utilizatorul este instiintat.

Cazul 3: amandoi utilizatorii inchid simultan

Un CLOSING simulatan a ambilor utilizatori de la cele 2 capete ale conexiunii determina schimbul de segmente FIN. Cand toate segmentele care preced segmentele FIN au fost procesate si confirmate, fiecare TCP poate confirma(ACK) FIN-ul pe care l-a primit. Amandoi vor inchide conexiunea dupa ce vor primi aceste ACK.

TCP A		TCP B
1. ESTABLISHED		ESTABLISHED
2. (Close)		
FIN-WAIT-1	--> <SEQ=100><ACK=300><CTL=FIN,ACK>	--> CLOSE-WAIT
3. FIN-WAIT-2	<-- <SEQ=300><ACK=101><CTL=ACK>	<-- CLOSE-WAIT
4. TIME-WAIT	<-- <SEQ=300><ACK=101><CTL=FIN,ACK>	(Close) <-- LAST-ACK
5. TIME-WAIT	--> <SEQ=101><ACK=301><CTL=ACK>	--> CLOSED
6. (2 MSL)		
CLOSED		

Secventa de inchidere normala

Figura 13.

TCP A		TCP B
1. ESTABLISHED		ESTABLISHED
2. (Close)		(Close)
FIN-WAIT-1	--> <SEQ=100><ACK=300><CTL=FIN,ACK>	... FIN-WAIT-1
	<-- <SEQ=300><ACK=100><CTL=FIN,ACK>	<--
	... <SEQ=100><ACK=300><CTL=FIN,ACK>	-->
3. CLOSING	--> <SEQ=101><ACK=301><CTL=ACK>	... CLOSING
	<-- <SEQ=301><ACK=101><CTL=ACK>	<--

... <SEQ=101><ACK=301><CTL=ACK> -->

4. TIME-WAIT	TIME-WAIT
(2 MSL)	(2 MSL)
CLOSED	CLOSED

Secventa de inchidere simultana

Figura 14.

3.6 Precedenta si securitate

Intentia este ca o conexiune sa fie permisa doar intre doua porturi cu exact aceeasi securitate si din acelasi sector si la nivelul superior de precedenta cerut de cele doua porturi.

Parametrii de precedenta si securitate folositi in TCP sunt exact cei definiti in Protocolul Internet (IP). In aceasta specificatie TCP termenul de "securitate/compartiment" este folosit pentru a indica parametrii de securitate utilizati in IP incluzand securitatea, compartimentul, grupul utilizatorului si restrictiile de manipulare.

O incercare de conexiune cu valori eronate de securitate/compartiment sau o valoare mica a precedentei trebuie respinsa utilizand o resetare. Respingerea unei conexiuni datorata valorii mici a precedentei apare dupa ce a fost primita o confirmare a SYN-ului.

De notat ca modulele TCP care opereaza doar la valori implicite ale precedentei tot vor trebui sa verifice precedenta segmentelor care vin si posibil sa-si ridice nivelul precedentei pe care il folosesc in conexiune.

Parametrii de securitate pot fi folositi chiar intr-un mediu nesigur (valorile ar indica date secrete), deci gazdele in aceste medii nesigure trebuie sa fie pregatite sa primeasca parametrii de securitate, desi nu trebuie sa-i trimita inapoi.

3.7. Comunicarea de date

Odata ce s-a stabilit conexiunea, datele sunt comunicate prin schimbul de segmente. Pentru ca segmentele se pot pierde datorita erorilor (suma de control eronata), sau datorita congestiei in retea, TCP-ul foloseste retransmiterea datelor (dupa o pauza) pentru a fi sigur de trimiterea la destinatie a fiecarui segment. Segmentele duplicate pot ajunge datorita retransmiterii din partea TCP sau din partea retelei. Asa cum am discutat in sectiunea cu numerele de secventa TCP-ul executa o serie de teste pentru numerele de secventa si numerele de confirmare in segmente pentru a verifica daca sunt acceptate.

Expeditorul de date tine minte urmatorul numar de secventa folosit in variabila SND.NXT. Destinatarul datelor tine minte urmatorul numar de secventa pe care in asteapta in variabila RCV.NXT. Destinatarul de date tine minte cel mai vechi numar de secventa neconfirmat in variabila SND.UNA. Daca fluxul de

date este momentan inutil si toate datele trimise au fost confirmate atunci cele trei variabile vor fi egale.

Cand expeditorul creeaza un segment si il transmite va incrementa SND.NXT. Cand destinatarul accepta un segment va incrementa RCV.NXT si va trimite o confirmare. Cand expeditorul de date primeste o confirmare va incrementa SND.UNA. Proportia in care valorile acestor variabile difera este o masura a intarzierii in comunicare. Valoare cu care vor fi incrementate este lungimea datelor din segment. Observati ca odata ce se ajunge in starea ESTABLISHED toate segmentele trebuie sa poarte informatii de confirmare curenta.

Apelul de inchidere al utilizatorului implica o functie push, asa cum implica acest lucru si flagul de control FIN intr-un segment ce soseste.

Expirarea timpului de retransmisie

Datorita varietatii retelelor care compun un sistem inter-retea si datorita utilizarilor masive a conexiunilor TCP expirarea retransmisiei trebuie determinata dinamic. Vom da aici o ilustratie a unei proceduri care determina expirarea timpului de retransmitere.

Un exemplu de procedura de expirare a timpului de retransmitere

Masuram timpul care se scurge intre trimiterea unui octet de date cu un anumit numar de secventa si primirea unei confirmari cu acest numar de secventa (segmentele trimise nu trebuie sa se potriveasca cu segmentele primite). Acest timp scurs se numeste Round Trip Time(RTT). Urmatoarea secventa calculeaza Smoothed Round Trip Time (SRTT) :

$$SRTT = (ALPHA * SRTT) + ((1-ALPHA) * RTT)$$

si pe baza acestui calcul se va calcula expirarea timpului de retransmitere :

$$RTO = \min[UNBOUND, \max[LBOUND, (BETA * SRTT)]]$$

unde UNBOUND este marginea superioara a timpului de expirare (de ex, 1 minut), LBOUND este marginea inferioara a timpului de expirare(de ex, 1 secunda), ALPHA este un factor de netezire (de ex, de la .8 la .9) si BETA este un factor de intarziere variabil(de ex, de la 1.3 la 2.0).

Comunicarea informatiilor urgente

Obiectivul mecanismului de urgenta a TCP este de a permite utilizatorului expeditor sa stimuleze utilizatorul destinatar sa accepte date urgente si sa permita TCP-ului destinatie sa indice utilizatorului destinatar cand toate datele urgente cunoscute in momentul de fata au fost primite la destinatar.

Acest mecanism permite crearea unui punct in fluxul de date numit punctul de final al informatiilor urgente. Ori de cate ori acest punct se afla inaintea numarului de secventa de primire (RCV.NXT) la TCP-ul de destinatie acest TCP

trebuie sa spuna utilizatorului sa intre in "modul de urgenta" cand numarul de secventa de primire ajunge din urma punctul de urgenta, TCP-ul trebuie sa spuna utilizatorului sa intre in "modul normal". Daca punctul de urgenta este updatat in timp ce utilizatorul este in "modul de urgenta", update-ul va fi invizibil utilizatorului.

Metoda utilizeaza un camp de urgenta care este transportat in toate segmentele trimise. Flagul de control URG indica faptul ca trebuie luat in considerare si campul de urgenta si ca acesta trebuie adaugat la numarul de secventa pentru a determina punctul de urgenta. Absenta acestui flag indica faptul ca nu exista date urgente de transmis.

Pentru a transmite o indicatie de urgenta utilizatorul trebuie de asemenea sa trimita cel putin un octet de date. Daca utilizatorul expeditor indica si un push atunci se grabeste trimiterea la destinatie a pachetului cu date urgente.

Manipularea Ferestrei

Fereastra trimisa in fiecare segment indica multimea numerelor de secventa pe care expeditorul ferestrei (destinatarul datelor) este pregatit sa o primeasca. Se face presupunerea ca fereastra e strans legata de spatiul bufferului care retine datele disponibile in acest moment pentru aceasta conexiune.

Indicarea unei ferestre mari incurajeaza transmiterile de date. Daca la destinatie ajung mai multe date decat pot fi acceptate ele vor fi aruncate. Acest lucru se intampla in retransmiteri excesive, prin adaugarea datelor care nu sunt necesare in retea si in TCP-uri. Indicarea unei ferestre mici poate restrictiona transmiterea de date pana in punctul de introducere a unei intarzieri round trip intre fiecare transmitere a unui nou segment.

Mecanismele oferite permit unui TCP sa previna aparitia unei ferestre mari si ulterior sa previna aparitia unei ferestre mult mai mici fara sa fi acceptat toate datele de dinainte. Aceasta asa zisa "micsorare ferestrei" este descurajata. Principiul robustetii dicteaza ca TCP-urile nu vor micsora ferestrele ele insele dar vor fi pregatite pentru asemenea comportament de alte TCP-uri.

TCP-ul expeditor trebuie sa fie gata sa accepte cereri de la utilizator si sa trimita cel putin un octet de date noi chiar daca fereastra de trimitere este zero. TCP-ul expeditor trebuie sa trimita regulat date TCP-ului destinatar chiar daca fereastra este zero. Se recomanda ca intervalul dupa care se vor retransmite datele atunci cand fereastra este zero sa fie de 2 minute. Aceasta retransmitere este esentiala pentru a garanta ca atunci cand oricare dintre dintre TCP-uri are fereastra zero redeschiderea ferestrei va fi raportata celuilalt TCP.

Cand TCP-ul destinatar are fereastra zero si ajunge un segment el tot trebuie sa trimita o confirmare pentru a indica urmatorul numar de secventa asteptat si fereastra curenta (zero).

TCP-ul expeditor impacheteaza datele care vor fi trimise in pachete care incap in fereastra curenta si poate reimpacheta segmente in coada de retransmitere. Aceasta reimpachetare nu este necesara, dar poate fi de ajutor.

Intr-o conexiune cu flux de date intr-o sinfura directie, informatiile despre ferestre vor fi transportate in segmente de confirmare care au toate acelasi

numar de secventa asa ca nu va mai fi nevoie sa le ordonam daca nu ajung in ordine. Aceasta nu e o problema serioasa, dar va permite informatiilor de fereastra sa se bazeze ocaasional pe rapoarte vechi de la destinatarul de date. O prelucrare pentru a evita aceasta problema este sa actionam asupra informatiilor de fereastra din segmentele care transporta cel mai mare numar de confirmare (adica segmente cu numere de secventa mai mari sau egale decat cel primit anterior).

Procedura de management a ferestrei are o influenta semnificativa in performanta comunicatiei. Urmatoarele comentarii sunt sugestii pentru implementatori.

Sugestii de management a ferestrei

Alocand o fereastra foarte mica vom determina ca datele sa fie transmise in multe segmente mici cand o performanta mai buna se atinge daca folosim mai putine segmente dar mai mari.

O sugestie pentru evitarea ferestrelor mici este ca destinatarul sa intarzie sa updatarea unei ferestre pana cand alocarea aditionala este de cel putin X la suta din alocarea maxima posibila pentru o conexiune (unde X poate fi intre 20 si 40).

O alta sugestie este ca expeditorul sa evite trimiterea de segmente mici pana cand fereastra este suficient de mare pentru trimiterea de date. Daca utilizatorul indica o functie push atunci datele trebuie trimise chiar daca sunt trimise in segmente mici.

Observam ca nu trebuie intarziate confirmarile pentru ca pot rezulta retransmiteri de date care nu sunt necesare. O strategie ar fi sa trimitem o confirmare cand ajunge un segment mic (fara sa facem update la informatiile de fereastra) si apoi sa trimitem o alta confirmare cu noile informatii de fereastra cand fereastra s-a marit.

Segmentul transmis pentru a verifica fereastra zero poate de asemenea sa determine trimiterea datelor in segmente din ce in ce mai mici. Daca un segment ce contine un singur octet de date transmis pentru a verifica daca o fereastra este zero este acceptat el consuma un octet din fereastra acum disponibila. Daca TCP-ul expeditor transmite atat cat poate ori de cate ori fereastra nu este zero, datele transmise vor fi impartite alternand in segmente mari si mici. Pe masura ce trece timpul, pauzele ocazionale la destinatar facand disponibila alocarea ferestrei, vor determina fragmentarea segmentelor mari in doua segmente mai mici. Si dupa un anumit timp transmiterea de date se va face in segmente mici.

Sugestia consta in faptul ca implementarile TCP trebuie incerce ca combine activ alocarile mici de ferestre cu alocari mari de ferestre, intrucat mecanismele pentru manipularea ferestrei tind sa duca la utilizarea de multe ferestre mici in implementari naive.

3.8. Interfete

Sunt desigur doua interfete care ne preocupa: interfata utilizator/TCP si interfata TCP/nivel-inferior. Avem un model elaborat al interfetei utilizator/TCP, dar interfata cu modulul protocolului de nivel inferior nu este specificata aici, deoarece va fi specificata in detaliu de specificatiile de protocol de nivel inferior. In cazul in care acest nivel inferior este IP notam cateva valori de parametri pe care TCP-ul le poate folosi.

Interfata utilizator/TCP

Urmatoarea descriere functionala a comenzilor utilizatorului catre TCP este, in cel mai fericit caz, fictiva, deoarece fiecare sistem de operare are facilitati diferite. Drept urmare, trebuie sa avertizam cititorii ca diferite implementari TCP pot avea interfete diferite cu utilizatorul. Totusi, toate TCP-rile trebuie sa ofere un anumit set minim de servicii pentru a garanta ca toate implementarile TCP-ului poate suporta aceeasi ierarhie de protocoale. Aceasta sectiune specifica interfetele functionale cerute de toate implementarile TCP.

Comenzi Utilizator TCP

Urmatoarele sectiuni caracterizeaza dpdv functional o interfata Utilizator/TCP. Notatia utilizata este similara celei folosite la toate procedurile sau apelurile de functie in limbajele de nivel inal, dar acest tip de folosire nu are ca scop sa elimine apelurile servicii de tip capcana. (de ex, SVCs, UUOs, EMTs).

Comenzile utilizator descrise mai jos specifica functiile de baza pe care TCP-ul trebuie sa le execute pentru a suporta comunicatiile intre procese. Implementarile individuale trebuie sa-si defineasca propriul lor format si pot oferi combinatii ale functiilor de baza intr-un singur apel. In particular, unele implementari pot dori sa deschida automat o conexiune la primul SEND sau la primul RECEIVE trimis de utilizator pentru o anumita conexiune.

In oferirea de facilitati in comunicatiile intre-procese, TCP-ul nu trebuie doar sa accepte comenzi , dar deasemenea sa returneze informatii procesului pe care il serveste. Acest lucru consta in:

(a) informatii generale despre o conexiune (de ex, intreruperi, inchidere la distanta, legarea de socketuri straine nespecificate).

(b) raspunsuri date comenzilor utilizator indicand succes sau diferite tipuri de eroare.

Open

Format: OPEN (port local, socket strain, activ/pasiv
[, interval de timp] [, precedenta] [, securitate/compartiment] [,
optiuni])
-> numele conexiunii locale

Presupunem ca TCP-ul local cunoaste identitatea proceselor pe care le serveste si va verifica autoritatea procesului care va utiliza conexiunea specificata. Depinzand de implementarea TCP, reseaua locala si identificatorii TCP pentru

adresa sursa vor fi specificati fie de TCP, fie de protocolul de nivel inferior (de ex IP). Aceste consideratii sunt rezultat al grijii pentru securitate, in masura in care nici un TCP sa nu fie capabil sa se dea drept altul si asa mai departe. In mod similar, nici un proces nu se poate confunda cu altul fara acceptul TCP-ului.

Daca flagul activ/pasiv este setat pe pasiv, atunci avem de-a face cu un apel de LISTENRE a unei anumite conexiuni. O deschidere pasiva poate avea fie un socket strain specificat care asteapta o anumita conexiune, fie un socket strain nespecificat care asteapta orice apel. Un apel pasiv complet specificat poate fi facut activ de de executia ulterioara a unei TRIMITERI.

Un bloc de control a transmisiei (TCB) este creat si partial completat cu date luate de la parametrii comenzii OPEN.

La o comanda OPEN activa, TCP-ul va incepe procedura cu sincronizarea (adica stabilirea) conexiunii imediat.

Pauza(intervalul de timp), daca este prezenta, permite apelantului sa stabileasca un interval de timp pentru toate datele transmise TCP-ului. Daca datele nu sunt transmise cu succes la destinatie in acest interval de timp, TCP-ul va abandona conexiunea. Intervalul de timp implicit global este de 5 minute.

TCP-ul sau anumite componente ale sistemului de operare va verifica autoritatea utilizatorului de a deschide o conexiune cu precedenta si securitatea/compartimentul specificate. Lipsa specificarii precedentei si securitatii/compartimentului in apelul functiei OPEN indica faptul ca trebuie folosite valorile implicite.

TCP-ul va accepta cereri de potrivire doar daca informatiile de spre securitate/compartiment sunt exact aceleasisi doar daca precedenta este mai mare sau egala decat precedenta ceruta in apelul OPEN.

Precedenta pentru conexiune este cea mai mare valoare dintre valorile cerute in apelul OPEN si primite de cerere; precedenta are o valorea fixata pentru intreaga durata de viata a conexiunii. Implementatorii vor dori sa dea utilizatorului controlul asupra negocierii precedentei. De exemplu, utilizatorului trebuie sa i se permita sa specifice faptul ca precedenta trebuie potrivita exact, sau ca orice incercare de marire a precedentei sa fie confirmata de utilizator.

Apelul functiei OPEN va intoarce catre utilizator numele conexiunii locale. Numele conexiunii locale poate apoi fi utilizat ca o prescurtare a conexiunii definita de perechea <socket local, socket strain>.

Send

Format: SEND (numele conexiunii locale, adresa bufferului, contor de octeti, flagul PUSH , flagul URGENT [,interval de timp])

Acest apel determina ca datele continute in bufferul utilizator indicat sa fie trimise catre conexiunea indicata. Daca conexiunea nu a fost deschisa ,

apelul SEND(TRIMITE) intoarce eroare. Unele implementari pot permite utilizatorilor sa TRMITA mai intai; in acest caz se va apela automat OPEN. Daca procesul apelant nu este autorizat sa foloseasca aceasta conexiune, se intoarce o eroare.

Daca este setat flagul PUSH, datele trebuie trimise imediat la destinatar si bitul PUSH va fi setat in ultimul segment TCP creat din buffer. Daca flagul PUSH nu este setat, datele pot fi combinate cu date TRIMISE (SEND) ulterior pentru o transmitere mai eficienta.

Daca este setat flagul URGENT, segmentele trimise la TCP-ul destinatie vor avea pointerul de urgenta setat. TCP-ul destinatar va semnala conditiile urgente procesului de destinatie daca pointerul de urgenta indica faptul ca datele care preced acest pointer nu au fost consumate de procesul destinatar. Scopul flagului URGENT este de a stimula destinatarul sa proceseze datele urgente si sa indice destinatarului cand au fost primite toate datele urgente cunoscute in momentul de fata.

Datele urgente vor informa utilizatorul ori de cate ori numarul de semnale TCP urgente trimise de expeditor nu va fi egal cu numarul celor primite la destinatie .

Daca in apelul OPEN nu se specifica nici un socket strain, dar conexiunea este stabilita(pentru ca o conexiune care LISTEN a devenit specificata datorita unui segment strain a ajuns la socketul local), atunci bufferul desemnat este trimis la socketul strain sugerat. Utilizatorii care fac apeluri OPEN fara a specifica socketul strain pot de asemenea folosi apeluri SEND fara a sti neaparat adresa socketului strain.

Totusi, daca se face un apel SEND inainte de a se cunoaste socketul strain, va fi returnata o eroare. Utilizatorii pot folosi apelul functiei STATUS pentru a determina starea conexiunii. In unele implementari TCP-ul poate semnala utilizatorului cand s-a facut legatura cu un socket necunoscut.

Daca se specifica un interval de timp, intervalul de timp a utilizatorului curent pentru aceasta conexiune va fi schimbat in cel nou.

In cea mai simpla implementare, SEND nu va returna controlul procesului expeditor pana cand ori transmisia este completa sau intervalul de timp a fost depasit. Totusi, aceasta simpla metoda este supusa blocajelor(de ex, ambele parti ale conexiunii ar putea incerca sa faca SEND inainte sa faca RECEIVE) si ofera o performanta foarte slaba, de aceea nu e recomandata. O implementare mai sofisticata ar returna imediat pentru a permite procesului sa mearga concurent cu dispozitivul de I/O al retelei si, mai mult de atat, sa permita mai multe SEND-uri simultane. SEND-urile multiple vor fi tratate pe rand , deci TCP-ul le va pune in coada de asteptare pe cele pe care nu le poate servi imediat.

Am presupus implicit ca avem de-a face cu o interfata cu utilizatorul asincrona in care unSEND determina aparitia unui fel de semnal sau o pseudo-intrerupere de la TCP. De exemplu, SEND-urile pot trimite o confirmare locala imediata, chiar daca segmentul trimis nu a fost confirmat de TCP-ul destinatar. Putem sa presupunem cu optimism un eventual succes. Daca ne inselam, conexiunea se va intrerupe oricum datorita depasirii intervalului de timp. In implementari de acest fel (sincrone), vor exista totusi cateva semnale asincrone, dar acestea se vor ocupa de conexiune si nu de segmente anume sau buffere.

Pentru ca procesele sa distinga intre indicatiile de succes sau eroare pentru diferite apeluri SEND, ar fi putea fi adecvat sa se returneze adresa bufferului alaturi de raspunsul codat la cererea de SEND. Semnalele de la TCP la utilizator sunt discutate mai jos, indicand informatiile care ar trebui returnate procesului apelant.

Receive

Format: RECEIVE (numele conexiunii locale, adresa bufferului, contor de octeti) -> contorul de octeti , flagul urgent, flagul push

Aceasta comanda alocă un buffer de primire asociat cu o conexiune specificată. Dacă nici o comandă OPEN nu precede această comandă sau procesul apelant nu este autorizat să folosească această conexiune, este returnată o eroare.

În cea mai simplă implementare , controlul nu va fi dat programului apelant până când bufferul a fost umplut sau a apărut o eroare, dar această schemă este predispusă la blocaje. O implementare mai sofisticată ar permite mai multor PRIMIRI să existe simultan. Acestea vor fi tratate pe măsura ce ajung segmentele. Această strategie permite un rezultat îmbunătățit cu același cost cu al unei scheme mai elaborate (posibil asincronă) și anunță programul apelant că a fost trimis un PUSH sau s-a umplut un buffer.

Dacă datele umplu bufferul înainte ca să apară un PUSH, flagul PUSH nu va fi setat ca răspuns la RECEIVE(PRIMIRE). Bufferul va fi umplut cu date atât cât permite capacitatea sa. Dacă apare un PUSH înainte ca să se umple bufferul, bufferul va fi returnat așa cum se afla(umplut parțial) și flagul PUSH va fi setat.

Dacă există date urgente, utilizatorul va fi informat de îndată ce datele vor fi ajuns prin intermediul unui semnal TCP-catre-utilizator. Utilizatorul destinatar trebuie să fie deci în "modul urgent". Dacă flagul URGENT este setat, datele urgente adiționale rămân. Dacă flagul URGENT nu este setat, apelul RECEIVE întoarce toate datele urgente și utilizatorul poate renunța la modul urgent. Observăm că datele care urmează după pointerul de urgentă (datele ne-urgente) nu pot fi livrate utilizatorului în același buffer cu datele urgente care le preced decât dacă limita e clar delimitată pentru utilizator.

Pentru a face diferență între apeluri RECEIVE diferite și pentru a avea grijă de cazul în care un buffer nu este complet umplut, codul de returnare este însoțit de un pointer la buffer și un bit de contorizare indicând lungimea actuală a datelor primite.

Alte implementări ale RECEIVE determină ca TCP-ul să aloce spațiu pentru buffer, sau TCP-ul poate să împartă un buffer înel cu utilizatorul.

Close

Format: CLOSE (numele conexiunii locale)

Această comandă determină închiderea conexiunii specificate. Dacă conexiunea nu este deschisă sau procesul apelant nu este autorizat să folosească această conexiune, se întoarce o eroare. Închiderea conexiunii se încearcă a fi o operație elegantă în sensul că SEND-urile neefectuate vor fi transmise (și retransmise) după cum permite controlul fluxului, până când sunt toate servite. Deci, ar trebui să fie acceptate câteva apeluri SEND, urmate de un CLOSE și se așteaptă ca toate datele trimise să ajungă la destinație. Ar trebui de asemenea să fie clar că utilizatorii ar trebui să mai primească date la închiderea conexiunii, pentru că ar putea să trimită ultimele sale date. Astfel, CLOSE înseamnă "nu mai am date de transmis", dar nu înseamnă "nu mai primesc date". S-ar putea întâmpla ca (dacă nivelul utilizator de protocol nu este bine gândit) partea care se închide să nu poată scăpa de toate datele înainte de expirarea timpului. În acest caz, CLOSE se transformă în ABORT și TCP-ul care se închide renunță.

Utilizatorul poate închide conexiunea oricând din proprie inițiativă sau ca răspuns la diferite mesaje de la TCP (adică, execuții de închidere de la distanță, expirare a timpului de transmitere, destinație inaccesibilă).

Deoarece închiderea unei conexiuni necesită comunicarea cu un TCP îndepărtat, conexiunile pot rămâne în starea închisă pentru o scurtă perioadă de timp. Încercări de a redeschide conexiunea înainte ca TCP-ul să răspundă la apelul CLOSE determină răspunsuri de eroare.

Close presupune de asemenea și funcția push.

Status

Format: STATUS (numele conexiunii locale) -> întoarce datele de stare

Aceasta este o comandă a cărei implementare depinde de utilizator și poate fi exclusă fără efecte adverse. Informațiile returnate vin din TCB-ul asociat conexiunii. Această comandă întoarce un bloc de date ce conține următoarele informații:

- socketul local,
- socketul strain,
- ferestra glisantă destinatar,
- ferestra expeditor,
- starea conexiunii,
- numărul de buffere care așteaptă confirmare,
- numărul de buffere în timpul primirii,
- starea de urgență,
- precedentă,
- securitate/compartiment,
- și expirarea transmisiunii.

Depinzând de starea conexiunii sau de implementarea propriei zise, o parte a informației poate să nu fie disponibilă sau să aibă sens. Dacă procesul apelant nu este autorizat să folosească această conexiune, se întoarce o eroare. Aceasta previne ca procesele neautorizate să obțină informații despre conexiune.

Abort

Format: ABORT (numele conexiunii locale)

Aceasta comanda determina ca toate trimiterile sau primirile de date sa fie intrerupte, TCB-ul sa fie inlaturat, si un mesaj special de RESET sa fie trimis TCP-ului de la celalalt capat al conexiunii. Depinzand de implementare, utilizatorii pot primi indicatii de intrerupere pentru fiecare SEND sau RECEIVE neefectuat, sau pur si simplu pot primi o confirmare a intreruperii.

Mesaje TCP-catre-Utilizator

Se presupune ca mediul sistemului de operare ofera un mijloc pentru TCP sa semnaleze asincron catre programul utilizator. Cand TCP-ul trimite mesaje catre programul utilizator, anumite informatii sunt primite de utilizator. Deseori in specificatie informatia va fi un mesaj de eroare. In alte cazuri va fi o informatie despre terminarea procesarii unui SEND sau unui RECEIVE sau altui apel utilizator.

Sunt asigurate urmatoarele informatii:

Numele conexiunii locale	Intodeauna
Sirul de raspuns	Intodeauna
Adresa bufferului	Send & Receive
Contorul de octeti (numara octetii primiti)	Receive
Flagul push	Receive
Flagul urgent	Receive

Interfata TCP/Nivel-inferior

TCP-ul apeleaza un modul de protocol de nivel inferior pentru a trimite si a primi efectiv informatii in retea. Un caz este cel al sistemului inter-retea ARPA unde modulul de nivel inferior este IP (Internet Protocol).

Daca protocolul de nivel inferior este IP, el ofera argumente pentru tipuri de servicii si pentru timpul de viata. TCP-l utilizeaza urmatoarele setari pentru acesti parametri:

Tipul de serviciu = Precedenta: rutine, Intarziere: normala, Output: normal, Incredere: normal, sau 00000000.

Timpul de viata = un minut, sau 00111100

De notat ca timpul de viata maxim pentru un segment este de 2 minute. Aici cerem in mod explicit ca un segment sa fie distrus daca nu poate fi trimis de sistemul internet intr-un minut.

Daca nivelul inferior este IP (sau alt protocol care ofera aceeasi infatisare) si se foloseste rutarea sursei, interfata trebuie sa permita ca informatia rutata sa fie comunicata. Acest lucru este indeosebi important pentru ca adresele sursa si destinatie folosite in suma de control a TCP sa fie adresa sursa de origine si adresa destinatie finala. Este de asemenea important sa pastram ruta de returnare pentru a raspunde cererilor de conexiune.

Orice alt protocol de nivel inferior va trebui sa ofere adresa sursa, adresa destinatie si campurile de protocol, si sa determine intr-un fel "lungimea TCP-ului", pentru a oferi serviciul functional al IP-ului si pentru a fi folosit in suma de control a TCP.

3.9 Procesarea evenimentelor

Procesarea infatisata in aceasta sectiune este un exemplu a unei posibile implementari. Alte implementari pot avea secvente de procesare putin diferite, dar ar trebui sa difere de cele din aceasta sectiune doar in detaliu, nu ca substanta.

Activitatea TCP-ului poate fi caracterizata ca raspunzand la evenimente. Evenimentele care apar pot fi impartite in trei categorii: apeluri utilizator, segmente care sosesc si expirarea timpului. Aceasta sectiune descrie procesarea pe care o indeplineste TCP-ul ca raspuns la fiecare eveniment. In multe cazuri procesarea ceruta depinde de starea conexiunii.

Evenimente care apar:

Apeluri utilizator

OPEN
SEND
RECEIVE
CLOSE
ABORT
STATUS

Segmente care sosesc

SEGMENT ARRIVES

Expirari al timpului

USER TIMEOUT
RETRANSMISSION TIMEOUT
TIME-WAIT TIMEOUT

Modelul interfatei TCP/utilizator este acela prin care comenzile utilizator primesc o returnare imediata si posibil un raspuns intarziat printr-un eveniment sau printr-o pseude-intrerupere. In descrierile urmatoare, termenul "semnal" inseamna cauza a raspunsului intarziat.

Raspunsurile de eroare sunt vazute ca siruri de caractere. De exemplu, comenzi utilizator care referentiaza conexiuni care nu exista primesc "eroare: conexiunea nu e deschisa".

Notati in urmatorul fragment ca toata aritmetica privind numerele de secventa, numerele de confirmare, ferestrele, etc este modulo 2^{32} marimea spatiului numarului de secventa. Notati de asemenea ca " \leq " inseamna mai mic sau egal (modulo 2^{32}).

O cale naturala de a gandi cum sa fie procesate segmentele care sosesc este sa ne imaginam ca ele sunt mai intai testate pentru a se asigura ca au numarul de secventa corect (adica, continuturile lor se afla in marginile ferestrei de

glisare din spatiul numarului de secventa) si apoi ca sunt in general puse in coada si procesate in ordinea numerelor de secventa.

cand un segment se suprapune cu alte segmente deja ajunse vom reconstrui segmentul pentru a contine doar datele noi si vom ajusta campurile antet pentru a fi consistente.

De notat ca daca nu se mentioneaza nici o schimbare a starii TCP-ul va ramane in aceeasi stare.

Apelul OPEN (DESCHIDE)

Starea CLOSED (CLOSED) -adica TCB-ul nu exista

Creati un nou bloc de control a transmisiunii (TCB) pentru a retine informatii despre starea conexiunii. Completati identificatorul socketului local, socketul strain, precedenta, securitate/compartiment si informatii despre expirarea timpului utilizator. De notat ca unele parti ale socketului strain pot fi nespecificate intr-un OPEN pasiv si vor fi completate de parametrii segmentului SYN care soseste. Verificati daca securitatea si precedenta sunt permise pentru acest utilizator, iar in caz negativ se intoarce "eroare: precedenta nu este permisa" sau "eroare: securitate/compartiment nu sunt permise". Daca conexiunea e pasiva, introduceti starea LISTEN (ASCULTA) si returnati. Daca conexiunea e activa si socketul strain nu este specificat intoarceti "eroare: socket strain nespecificat"; daca conexiunea e activa si socketul strain este specificat, emiteti un segment SYN. Un numar de secventa initial (ISS) va fi selectat. Un segment SYN de forma <SEQ=ISS><CTL=SYN> este trimis. Setati SND.UA cu valoarea ISS, SND.NXT pe valoarea ISS+1, introduceti starea SYN-SENT si returnati.

Daca apelantul nu are acces la socketul local specificat, returnati "eroare: conexiune ilegala pentru acest proces". Daca nu exista spatiu pentru a crea o noua conexiune , returnati "eroare: resurse insuficiente".

Starea LISTEN (ASCULTA)

Daca socketul strain si cel activ sunt specificate, schimbati apoi conexiunea din pasiva in activa, selectati un ISS. Trimiteti un segment SYN, Setati SND.UNA cu valoarea ISS, SND.NXT cu valoarea ISS+1. Introduceti starea SYN-SENT. Datele asociate cu SEND pot fi trimise cu un segment SYN sau puse in coada de transmisie dupa ce se seteaza starea ESTABLISHED. Bitul urgent, daca este cerut in comanda trebuie trimis cu segmentele de date trimise ca rezultat al acestei comenzi. Daca nu este loc pentru pune cererea in coada, raspundeti cu "eroare: resurse insuficiente". Daca socketul strain nu a fost specificat, atunci returnati "eroare: socket strain nespecificat".

STAREA SYN-SENT
STAREA SYN-RECEIVED
STAREA ESTABLISHED
STAREA ASTEAPTA-FIN-1
STAREA ASTEAPTA-FIN-2
STAREA CLOSE-WAIT

STAREA CLOSED
STAREA LAST-ACK
STAREA TIME-WAIT

Returnati: "eroare: conexiunea deja exista".

Apelul SEND (TRIMITE)

Starea CLOSED (INCHIS) (adica TCB-ul nu exista)

Daca utilizatorul nu are acces la conexiune, returnati "eroare: conexiune ilegala pentru acest proces". Altfel, returnati "eroare: conexiunea nu exista".

Starea LISTEN (ASCULTA)

Daca socketul strain este specificat, atunci schimbati conexiunea din una pasiva in un activa, selectati ISS. Trimiteti segmentul SYN, setati SND.UNA cu valoarea ISS, SND.NXT cu valoarea ISS+1. Introduceti starea SYN-SENT. Datele asociate cu SEND pot fi trimise cu un segment SYN sau puse in coada de transmitere dupa setarea starii ESTABLISHED. Bitul urgent, daca este cerut in comanda, trebuie trimis cu segmentele de date trimise ca rezultat al acestei comenzi. Daca nu este pentru a pune in coada cererea, raspundeti cu "eroare: resurse insuficiente". Daca socketul strain nu este specificat, atunci returnati "eroare: socket strain nespecificat".

STAREA SYN-SENT
STAREA SYN-RECEIVED

Puneti datele de transmis in coada dupa setarea starii ESTABLISHED. Daca nu exista spatiu pentru a pune datele in coada, raspundeti cu "eroare: resurse insuficiente".

STAREA ESTABLISHED
STAREA CLOSE-WAIT

Fragmentati bufferul si trimite-ti-l cu o confirmare (valoarea confirmarii = RCV.NXT). Daca nu este suficient spatiu pentru a tine minte bufferul, intoarceti "eroare: resurse insuficiente".

Daca flagul urgent, atunci SND.UP <- SND.NXT-1 si setati pointerul urgent in segmentele care pleaca.

STAREA FIN-WAIT-1
STAREA FIN-WAIT-2
STAREA CLOSED
STAREA LAST-ACK
STAREA TIME-WAIT

Returneaza "eroare: inchiderea conexiunii" si nu serveste cererea.

Apelul RECEIVE (PRIMESTE)

Starea CLOSED (INCHIS) adica TCB-ul nu exista

Daca utilizatorul nu are acces la conexiune, returnati "eroare: conexiune ilegala pentru acest proces". Altfel returnati "eroare: conexiunea nu exista".

STAREA WAIT
STAREA SYN-SENT
STAREA SYN-RECEIVED

Se foloseste o coada pentru procesare dupa setarea starii in ESTABLISHED. Daca nu este spatiu pentru a pune cererea in coada, returnati "eroare: resurse insuficiente".

STAREA ESTABLISHED
STAREA FIN-WAIT-1

STAREA FIN-WAIT-2

Daca segmentele care sosesc sunt insuficiente pentru a satisface cererea, vom pune si cererea in coada. Daca nu este spatiu pentru a tine minte RECEIVE, raspundeti cu "eroare: resurse insuficiente". Reasamblati segmentelor din coadain bufferul de primire si returnati catre utilizator. Marcati "am vazut push" (PUSH) daca este cazul.

Daca RCV.UP este inaintea datelor curente pasate utilizatorului, anuntati utlizatorul de prezenta datelor urgente. Cand TCP-ul isi asuma raspunderea de a livra datele utilizatorului, acest lucru trebuie comunicat expeditorului printr-o confirmare. Structura unei asemenea confirmari este descrisa mai jos in discutia despre procesarea unui segment care soseste.

Apelul RECEIVE (PRIMESTE)

Starea CLOSE-WAIT

Pentru ca partea indepartata a trimis deja FIN, PRIMIRILE trebuie satisfacute de un test la indemana, dar nu inca trimis utilizatorului. Daca nici un text nu asteapta sa fie livrat, apelul RECEIVE va returna o "eroare: se inchide conexiunea". Altfel, orice alt text ramas poate fi folosit sa satisfaca RECEIVE.

STAREA CLOSED
STAREA LAST-ACK
STAREA TIME-WAIT

Returneaza "eroare: conexiunea se inchide".

Apelul CLOSE (INCHIDE)

Starea CLOSED (INCHIS) -adica, TCB-ul nu exista

Daca utilizatorul nu are acces la conexiune , returnati "eroare: conexiune ilegala pentru acest proces". Altfel, returnati "eroare: conexiunea nu exista".

Starea LISTEN (ASCULTA)

Orice PRIMIRI neefectuate sunt returnate cu raspunsuri de tipul "eroare : inchidere". Stergeti TCB-ul, introduceti starea CLOSED si returnati.

Starea SYN-SENT

Stergeti TCB-ul si returnati raspunsul "eroare : inchidere" la orice TRIMITERI din coada, sau la orice PRIMIRI.

Starea SYN-RECEIVED

Daca nu au fost furnizate SEND-uri si nu au fost intre timp alte date de trimis, atunci se formeaza un segment FIN si se trimite si se intra in starea FIN-WAIT-1; altfel, datele sunt puse in coada pentru procesare dupa intrarea in starea ESTABLISHED.

Starea ESTABLISHED

Se pun in coada cererea pana cand toate SEND-urile au fost segmentate, apoi se formeaza si se trimite un segment FIN. In orice caz, se intra in starea FIN-WAIT-1.

Starea FIN-WAIT-1
Starea FIN-WAIT-2

Strict vorbind, aceasta e o eroare si ar trebui sa se primeasca "eroare: se inchide conexiunea". Se accepta si raspunsul "ok" atata timp cat un alt semnal FIN nu este emis (primul FIN ar putea fi retransmis totusi).

Starea CLOSE-WAIT

Se pune in coada aceasta cerere pana cand toate SEND-urile precedente au fost segmentate; atunci se trimite un segment FIN si se intra in starea CLOSED.

Starea CLOSED
Starea LAST-ACK
Starea TIME-WAIT

Se raspunde cu "eroare: se inchide conexiunea".

Apelul ABORT

Starea CLOSED (adica, TCB-ul nu exista)

Daca utilizatorul nu ar avea acces la conexiune, se returneaza "eroare: conexiunea ilegala pentru acest proces". Altfel se returneaza "eroare: conexiunea nu exista".

Starea LISTEN

Orice RECEIVE-uri inca neefectuate ar trebui sa fie intoarse cu raspunsul "eroare: conexiune resetata". Se sterge TCB-ul, se intra in starea CLOSED si se returneaza.

Starea SYN-SENT

Toate SEND-urile si RECEIVE-urile din coada ar trebui sa primeasca instiintari de genul "conexiune resetata", se sterge TCB-ul, se intra in starea CLOSED si se returneaza.

Starea SYN-RECEIVED

Starea ESTABLISHED

Starea FIN-WAIT-1

Starea FIN-WAIT-2

Starea CLOSE-WAIT

Se trimite un segment de resetare:

<SEQ=SND.NXT><CTL=RST>

Toate PRIMIRILE si TRIMITERILE ar trebui sa primeasca o notificare de "resetare a conexiunii"; toate segmentele puse in coada pentru a fi transmise (exceptie facand TCB-ul format deasupra) sau retransmise ar trebui sa fie inlaturate, TCB-ul sters, intrarea in starea CLOSED si sa se returneze.

Starea CLOSING

Starea LAST-ACK

Starea TIME-WAIT

Se raspunde cu "ok" si se sterge TCB-ul, se intra in starea CLOSED si se returneaza.

Apelul STATUS (STARE)

Starea CLOSED (adica, TCB-ul nu exista)

Daca utilizatorul nu ar avea acces la asemenea conexiune, se intoarce "eroare: conexiune ilegala pentru acest proces". Altfel, se intoarce "eroare: conexiunea nu exista".

Starea LISTEN

Se intoarce "starea = LISTEN" si pointerul la TCB.

Starea SYN-SENT

Se intoarce "starea = SYN-SENT" si pointerul la TCB.

Starea SYN-RECEIVED

Se intoarce "starea = SYN-RECEIVED" si pointerul la TCB.

Starea ESTABLISHED

Se intoarce "starea = ESTABLISHED" si pointerul la TCB.

Starea FIN-WAIT-1

Se intoarce "starea = FIN-WAIT-1" si pointerul la TCB.

Starea FIN-WAIT-2

Se intoarce "starea = FIN-WAIT-2" si pointerul la TCB.

Starea CLOSE-WAIT

Se intoarce "starea = CLOSE-WAIT" si pointerul la TCB.

Starea CLOSING

Se intoarce "starea = CLOSING" si pointerul la TCB.

Starea LAST-ACK

Se intoarce "starea = LAST-ACK" si pointerul la TCB.

Starea TIME-WAIT

Se intoarce "starea = TIME-WAIT" si pointerul la TCB.

SEGMENT ARRIVES

(SEGMENTE AJUNSE)

Daca starea este CLOSED (adica, TCB-ul nu exista), atunci se renunta la toate datele din segmentul care soseste. Un segment care soseste si contine un RST este de asemenea inlaturat. Un segment care soseste si nu contine un RST determina trimiterea unui RST ca raspuns. Confirmarea si valorile campurilor de secventa sunt selectate pentru a face secventa de resetare acceptabila pentru TCP-ul care a trimis segmentul.

Daca bitul ACK nu este setat, este folosit numarul de secventa zero,

<SEQ=0><ACK=SEG.SEQ+SEG.LEN><CTL=RST,ACK>

Daca bitul ACK este setat

<SEQ=SEG.ACK><CTL=RST>

Se returneaza.

Daca suntem in starea LISTEN, atunci prima verificare este pentru un RST. Un segment RST care soseste ar trebui sa fie ignorat. Se returneaza. Se verifica a doua oara un ACK. Orice confirmare este eronata daca ajunge la o conexiune aflata inca in starea LISTEN. Un segment de resetare ar trebui sa fie formata pentru fiecare segment care soseste si transporta un ACK. RST-ul ar trebui formatat astfel:

<SEQ=SEG.ACK><CTL=RST>

Se returneaza.

A treia verificare este pentru un SYN. Daca bitul SYN este setat, se verifica securitatea. Daca securitatea/compartimentul unui segment care soseste nu se potriveste exact cu securitatea.compartimentul din TCB atunci se trimite un reset si se se returneaza.

<SEQ=SEG.ACK><CTL=RST>

Daca SEG.PRC este mai mare decat TCB.PRC atunci, daca se permite din partea utilizatorului si a sistemului se seteaza TCB.PRC<-SEG.PRC, daca nu se permite se trimite o resetare si se returneaza.

<SEQ=SEG.ACK><CTL=RST>

Daca SEG.PRC este mai mic decat TCB.PRC atunci se continua.

Se seteaza RCV.NXT la SEG.SEQ+1, IRS se seteaza la SEG.SEQ si orice alt control sau text ar trebui puse in coada pentru a fi procesate mai tarziu. ISS ar trebui selectate si trimis un segment SYN de forma:

<SEQ=ISS><ACK=RCV.NXT><CTL=SYN,ACK>

SND.NXT este setat la ISS+1 si SND.UNA la ISS. Starea conexiunii ar trebui schimbata la SYN-PRIMIT. Observati ca orice alt control sau data care soseste (combinata cu SYN) va fi procesata in starea SYN-RECEIVED, dar procesarea lui SYN si ACK nu ar trebui sa se repete. Daca ascultarea nu a fost specificata in intregime (adica, socketul strain nu a fost specificat in intregime), atunci campurile nespecificate ar trebui sa fie completate acum.

SND.NXT is set to ISS+1 and SND.UNA to ISS. The connection state should be changed to SYN-RECEIVED. Note that any other incoming control or data (combined with SYN) will be processed in the SYN-RECEIVED state, but processing of SYN and ACK should not be repeated. If the listen was not fully specified (i.e., the foreign socket was not fully specified), then the unspecified fields should be filled in now.

A patra verificare a unui text sau a unui control: Orice alt control sau segment ce transporta un text (care nu contine un SYN) trebuie sa aiba un ACK si astfel vor fi inlaturate de procesarea ACK-ului. Un segment RST care soseste nu poate fi valid deoarece nu ar fi putut fi trimis ca raspuns la orice data trimisa de aceasta instanta a conexiunii. Deci este putin probabil sa ajungeti aici, dar daca ajungeti, renuntati la segment si returnati.

Daca starea este SYN-SENT atunci se verifica prima data bitul ACK. Daca bitul ACK este setat, daca SEG.ACK <= ISS sau SEG.ACK > SND.NXT se trimite o resetare (exceptand cazul in care bitul RST este setat, caz in care se renunta la segment si se returneaza)

<SEQ=SEG.ACK><CTL=RST>

si se renunta la segment. Se returneaza.

Daca SND.UNA <= SEG.ACK <= SND.NXT atunci se accepta ACK.

A doua oara se verifica bitul RST. Daca bitul RST este setat, daca se accepta ACK-ul atunci se semnaleaza utilizatorului "eroare: conexiune resetata", se renunta la segment, se intra in starea CLOSED, se sterge TCB-ul si se returneaza.

A treia oara se verifica securitatea si precedenta. Daca securitatea/compartimentul in segment nu coincide cu securitatea/compartimentul din TCB, se trimite o resetare.

Daca exista un ACK

<SEQ=SEG.ACK><CTL=RST>

Altfel

<SEQ=0><ACK=SEG.SEQ+SEG.LEN><CTL=RST,ACK>

Daca exista un ACK, precedenta in segment trebuie sa se potriveasca cu precedenta din TCB, altfel, se trimite o resetare

<SEQ=SEG.ACK><CTL=RST>

Daca nu exista nici un ACK, daca precedenta din segment este mai mare decat precedenta din TCB atunci daca se permite din partea utilizatorului si a sistemului se creste precedenta in TCB pana la aceea din segment, daca nu se permite cresterea precedentei se trimite o resetare.

<SEQ=0><ACK=SEG.SEQ+SEG.LEN><CTL=RST,ACK>

Daca precedenta in segment este mai mica decat precedenta din TCB se continua.

Daca se trimite o resetare, se renunta la segment si se returneaza.

A patra verificare este pentru bitul SYN. Ar trebui sa se ajunga la acest pas doar daca ACK-ul este ok, sau daca nu exista nici un ACK si segmentul nu contine nici un RST.

Daca bitul SYN este setat si securitatea/compartimentul si precedenta sunt acceptate, atunci RCV.NXT este setat la SEG.SEQ+1, IRS este setat la SEG.SEQ.SND.UNA ar trebui incrementat pentru a fi egal cu SEG.ACK (daca exista vreun ACK) si orice segment din coada de retransmitere care sunt astfel confirmate ar trebui inlaturate.

Daca SND.UNA > ISS (SYN-ul nostru a fost confirmat), se schimba starea conexiunii in starea ESTABLISHED, se formeaza un segment ACK

<SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>

si este trimis. Datele sau controalele care sunt puse in coada pentru transmitere pot fi incluse. Daca exista alte controale sau texte in segment atunci se continua procesarea la pasul al saselea pana la locul unde este verificat bitul URG, altfel se returneaza.

In alte cazuri, se intra in starea SYN-RECEIVED, se formeaza un segment SYN,ACK

<SEQ=ISS><ACK=RCV.NXT><CTL=SYN,ACK>

si se trimite. Daca exista alte controale sau texte in segment, ele vor fi puse in coada pentru a fi procesate dupa ce s-a intrat in starea ESTABLISHED, apoi se returneaza.

A cincea verificare se face daca nici bitul SYN si nici RST nu sunt setate; in acest caz se renunta la segment si se returneaza.

Altfel, se verifica prima data numarul de secventa

Starea SYN-RECEIVED
Starea ESTABLISHED
Starea FIN-WAIT-1
Starea FIN-WAIT-2
Starea CLOSE-WAIT
Starea CLOSING
Starea LAST-ACK
Starea TIME-WAIT

Segmentele sunt procesate in secventa. Testele initiale la sosire sunt folosite pentru a inlatura duplicatele vechi, dar procesarea ulterioara este realizata in ordinea SEG.SEQ. Daca continutul segmentului determina suprapunerea marginilor vechi cu cele noi, doar continuturile noi trebuie procesate.

Sunt patru cazuri pentru testul de acceptare pentru un segment care soseste:

Lungimea Segmentului	Fereastra de Primire	Test
-----	-----	-----
0	0	SEG.SEQ = RCV.NXT
0	>0	RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND
>0	0	nu e acceptat
>0	>0	RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND or RCV.NXT =< SEG.SEQ+SEG.LEN-1 < RCV.NXT+RCV.WND

Daca RCV.WND este zero, nu va fi acceptat nici un segment, dar ar trebui sa existe o acceptare speciala pentru a putea accepta ACK-uri valide, URG-uri si RST-uri.

Daca un segment care soseste nu este acceptat, ar trebui sa se trimita o confirmare ca raspuns (in afara de cazul cand bitul RST este setat, caz in care se renunta la segment si se returneaza):

<SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>

Dupa trimiterea confirmarii, se renunta la segmentul neacceptat si se returneaza.

In urmatorul segment, se presupune ca segmentul este segmentul ideal care incepe la RCV.NXT si nu depaseste fereastra. Segmentele pot fi croite sa se potriveasca presupunerii prin aranjarea portiunilor care sunt in afara ferestrei (se includ SYN si FIN), si vor fi procesate mai departe doar daca segmentul

incepe la RCV.NXT. Segmentele cu numere de secventa de inceput mai mari vor fi pastrate pentru o procesare ulterioara.

apoi se verifica bitul RST,

Starea SYN-SENT

Daca bitul RST este setat

Daca aceasta conexiune este initializata cu un OPEN pasiv (adica, vine din starea de LISTEN), atunci conexiunea revine in starea LISTEN. Utilizatorul nu trebuie sa fie informat. Daca aceasta conexiune a fost initializata cu un OPEN activ (adica, vine din starea SYN-SENT), atunci conexiunea a fost refuzata si se semnalizeaza utilizatorul cu "conexiune refuzata". In alte cazuri, toate segmentele din coada de retransmitere ar trebui sa fie inlaturate. In cazul unui OPEN activ, se intra in starea CLOSE (CLOSED) si se sterge TCB-ul, iar apoi se returneaza.

ESTABLISHED

FIN-WAIT-1

FIN-WAIT-2

CLOSE-WAIT

Daca bitul RST este apoi setat, orice PRIMIRI si TRIMITERI neefectuate ar trebui sa primeasca raspunsul "resetare". Toate cozile cu segmente trebuie golite. Utilizatorii ar trebui deasemenea sa primeasca un semnal general nesolicitat: "conexiune resetata". Se intra in starea CLOSED, se sterge TCB-ul si se returneaza.

Starea CLOSING

Starea LAST-ACK

TIME-WAIT

Daca bitul RST este apoi setat, se intra in starea CLOSED, se sterge TCB-ul si se returneaza.

in al treilea rand se verifica securitatea si precedenta

SYN-RECEIVED

Daca securitatea/compartimentul si precedenta din segment nu se potrivesc exact cu securitatea/compartimentul si precedenta din TCB, se trimite o resetare si apoi se returneaza.

Starea ESTABLISHED

Daca securitatea/compartimentul si precedenta din segment nu se potrivesc exact cu securitatea/compartimentul si precedenta din TCB, se trimite o resetare, oricr PRIMIRI si TRIMITERI neefectuate ar trebui sa primeasca raspunsul "resetare". Toate cozile de segmente ar trebui sa fie golite. Utilizatorii ar trebui de asemenea sa primeasca un semnal general nesolicitat: "conexiune resetata". Se intra in starea CLOSED, se sterge TCB-ul si se returneaza.

De observat ca aceasta verificare este plasata dupa verificarea secventei pentru a preveni ca un segment dintr-o conexiune mai veche intre

aceste porturi, cu o securitate si o precedenta diferite, sa cauzeze o anulare a conexiunii curente.

a patra verificare este a bitului SYN,

SYN-RECEIVED
STAREA ESTABLISHED
STAREA FIN-WAIT-1
STAREA FIN-WAIT-2
STAREA CLOSE-WAIT
STAREA CLOSING
STAREA LAST-ACK
STAREA TIME-WAIT

Daca SYN-ul se afla in fereastra atunci avem de-a face cu o eroare, se trimite o resetare, orice RECEIVE-uri si SEND-uri neefectuate ar trebui sa primeasca raspunsul "resetare", toate cozile de segmente ar trebui sa fie golite, utilizatorul ar trebui de asemenea sa primeasca un semnal general nesolicitat: "conexiune resetata", se intra in stare CLOSED, se sterge TCB-ul si se returneaza.

Daca SYN-ul nu se afla in fereastra, nu se va ajunge la acest pas si ar fi trebuit sa se trimita o confirmare la primul pas (verificare numarului de secventa).

a cincea verificare se face asupra campului ACK,
daca bitul ACK este nesetat se renunta la segment si se returneaza
daca bitul ACK este setat,

STAREA SYN-RECEIVED

Daca $SND.UNA \leq SEG.ACK \leq SND.NXT$ atunci se intra in starea ESTABLISHED si se continua procesarea. Daca confirmarea segmentului nu este acceptata, se formeaza un segment de resetare,
<SEQ=SEG.ACK><CTL=RST>

si se trimite.

STAREA ESTABLISHED

Daca $SND.UNA < SEG.ACK \leq SND.NXT$ atunci, se seteaza $SND.UNA \leftarrow SEG.ACK$. Orice segmente din coada de retransmitere care sunt total confirmate sunt inlaturate. Utilizatorii ar trebui sa primeasca o confirmare pozitiva pentru bufferele care au fost SEND si confirmate in intregime (adica, bufferul SEND ar trebui sa fie intors cu raspunsul "ok"). Daca ACK este duplicat ($SEG.ACK < SND.UNA$), poate fi ignorat. Daca ACK confirma ceva ce nu a fost inca trimis ($SEG.ACK > SND.NXT$) atunci se trimite un ACK, se renunta la segment si se returneaza.

Daca $SND.UNA < SEG.ACK \leq SND.NXT$, urmatoarea fereastra ar trebui sa fie actualizata. Daca ($SND.WL1 < SEG.SEQ$ sau ($SND.WL1 = SEG.SEQ$ si $SND.WL2 \leq SEG.ACK$)), se seteaza $SND.WND \leftarrow SEG.WND$, se seteaza $SND.WL1 \leftarrow SEG.SEQ$ si $SND.WL2 \leftarrow SEG.ACK$.

Se observa ca SND.WND este un offset al SND.UNA, ca SND.WL1 inregistreaza numarul de secventa al ultimului segment folosit pentru a actualiza SND.WND, si ca SND.WL2 inregistreaza numarul de confirmare al ultimului segment folosit sa actualizeze SND.WND. Verificarea este utila in acest caz pentru ca previne actualizarea ferestrei de catre segmente vechi.

STAREA FIN-WAIT-1

Ca o completare a procesarii starii ESTABLISHED, daca FIN este confirmat in acest moment, se intra in starea FIN-WAIT-2 si se continua procesarea in aceasta stare.

STAREA FIN-WAIT-2

Ca o completare a procesarii starii ESTABLISHED, daca coada de retransmitere este goala, apelul CLOSE (INCHIDE) al utilizatorului poate fi confirmat ("ok"), dar nu se sterge TCB-ul.

STAREA CLOSE-WAIT

Se face aceeasi procesare ca in cazul starii ESTABLISHED.

STAREA CLOSING

Ca o completare a procesarii pentru starea ESTABLISHED, daca ACK-ul confirma FIN, atunci se intra in starea TIME-WAIT, altfel se ignora segmentul.

STAREA LAST-ACK

Singurul lucru care poate sa ajunga in aceasta stare este o confirmare a flagului FIN. Daca FIN-ul nostru este confirmat, se sterge TCB-ul, se intra in starea CLOSED si se returneaza.

STAREA TIME-WAIT

Singurul lucru care poate ajunge in aceasta stare este retransmiterea unui FIN de la distanta. Se confirma acest FIN si se reporneste timpul (timeout) egal cu de doua ori durata maxima de viata a unui segment (MSL).

in al saselea rand se verifica bitul URG,

STAREA ESTABLISHED

STAREA FIN-WAIT-1

STAREA FIN-WAIT-2

Daca este setat bitul URG, $RCV.UP \leftarrow \max(RCV.UP, SEG.UP)$ si se semnalizeaza utilizatorului ca celalalt capat are date urgente daca pointerul urgent (RCV.UP) este inaintea datelor consumate. Daca utilizatorul a fost deja anuntat (sau se afla inca in "modul urgent") de trimiterea acestei secvente continue de date urgente, nu se mai anunta inca o data utilizatorul.

STAREA CLOSE-WAIT

STAREA CLOSING

STAREA LAST-ACK

TIME-WAIT

Aceasta n-ar trebui sa apara fiindca s-a primit un FIN de la celalalt capat. Se ignora URG.

a saptea verificare proceseaza segmentul care contine textul,

STAREA ESTABLISHED
STAREA FIN-WAIT-1
STAREA FIN-WAIT-2

Odata in starea ESTABLISHED, este posibil sa livram segmentul cu text spre bufferul de RECEIVE a utilizatorului. Textul din segmente poate fi mutat in buffer pana cand ori bufferul se umple, ori segmentul se goleste. Daca segmentul se goleste si poarta flagul PUSH, atunci utilizatorul este informat, cand bufferul este returnat, ca a fost primit un PUSH.

Cand TCP-ul isi ia responsabilitatea pentru a livra datele utilizatorului, trebuie de asemenea sa confirme primirea datelor. Odata ce TCP-ul isi ia raspunderea pentru date, el trimite RCV.NXT peste datele acceptate si ajusteaza RCV.WND cat mai potrivit cu disponibilitatea bufferului curent. Totalul lui RCV.NXT si RCV.WND nu ar trebui sa fie redus.

A se lua in seama sugestiile privind manipularea ferestrelor in sectiunea 3.7.

Se trimite o confirmare a formei:

<SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>

Aceasta confirmare ar trebui sa fie intoarsa unui segment in curs de transmitere fara a-si atrage o intarziere exagerata.

STAREA CLOSE-WAIT
STAREA CLOSING
STAREA LAST-ACK
STAREA TIME-WAIT

Aceasta stare n-ar trebui sa apara, fiindca un flag FIN a fost primit de la celalalt capat. Se ignora segmentul cu text.

in al optulea rand, se verifica bitul FIN,

Nu se proceseaza bitul FIN daca starea este CLOSED, ASTEAPTA sau SYN-SENT, pentru ca SEG.SEQ nu poate fi validat; se renunta la segment si se returneaza.

Daca bitul FIN este setat, se anunta utilizatorul: "conexiunea se inchide" si se returneaza orice TRIMITERI in asteptare cu acelasi mesaj, se trimite RCV.NXT peste FIN si se trimite o confirmare pentru FIN. De notat ca FIN implica PUSH pentru orice segment cu text care nu a fost inca trimis utilizatorului.

STAREA SYN-RECEIVED
STAREA ESTABLISHED

Se intra in stare CLOSE-WAIT.

STAREA FIN-WAIT-1

Daca FIN-ul nostru a fost confirmat (poate in acest segment), atunci se intra in starea TIME-WAIT, se porneste cronometrul de timp de asteptare, se opresc celelalte cronometre; altfel se intra in starea CLOSING.

STAREA FIN-WAIT-2

Se intra in starea TIME-WAIT. Se porneste cronometrul de timp de asteptare si se inchid celelalte cronometre.

STAREA CLOSE-WAIT

Ramane in starea CLOSE-WAIT.

STAREA CLOSING

Ramane in stare CLOSING.

STAREA LAST-ACK

Ramane in starea LAST-ACK.

STAREA TIME-WAIT

Ramane in starea TIME-WAIT. Reporneste intervalul de timp de asteptare de doua MSL.

si se returneaza.

TIMEOUT UTILIZATOR

In oricare dintre stari, daca timeout-ul utilizatorului expira, se golesc toate cozile, se semnalizeaza utilizatorului "error: connection aborted due to user timeout" pentru orice apel nesatisfacut, se sterge TCB-ul, se intra in starea CLOSED si se returneaza.

TIMEOUT DE RETRANSMITERE

Pentru orice stare, daca timeout-ul expira pentru un segment aflat in coada de retransmitere, se trimite segmentul din nou la inceputul cozii, se reinitializeaza timer-ul de retransmitere si se returneaza.

TIME-WAIT TIMEOUT

Daca time-wait timeout expira intr-o conexiune, se sterge TCB-ul, se intra in starea CLOSED si se returneaza.

GLOSAR

1822

Raportul BBN 1822, "The Specification of the Interconnection of a Host and an IMP". Specificatiile sau interfata dintre o gazda si ARPANET.

ACK

Un bit de control (de confirmare) care nu ocupa spatiu de secventa, care indica faptul ca acel camp de confirmare al segmentului ne da urmatorul numar de secventa pe care expeditorul segmentului se asteapta sa-l primeasca, primind de aici inainte confirmarea a tuturor numerelor de secventa anterioare.

Mesaj ARPANET

Unitatea de transmisie intre o gazda si un IMP in ARPANET. Marimea maxima este de aproximativ 1012 octeti (8096 biti).

Pachet ARPANET

O unitate de transmisie folosita intern intre ARPANET si IMP-uri. Marimea maxima este de aproximativ 126 de octeti (1008 biti).

Conexiune

o cale de comunicare logica identificata de o pereche de socketuri

datagrama

Un mesaj trimis intr-o retea de comunicari intre calculatoare bazata pe schimburi.

Adresa de destinatie

Adresa de destinatie, de obicei identificatorii retea sau identificatorii gazda.

FIN

Un bit de control (de terminare) care ocupa un numar de secventa, care indica faptul ca utilizatorul nu va mai trimite date sau controale care sa ocupe spatiu de secventa.

fragment

O portiune dintr-o unitate logica de date, in particular un fragment de internet este o portiune dintr-o datagrama internet.

FTP

Un protocol de transfer de fisiere.

header

O informatie de control la inceputul unui mesaj, segment, fragment, pachet sau bloc de date.

gazda

Un calculator. In particular, sursa sau destinatia din punctul de vedere a comunicarii in retea.

Identificare

Un camp de Protocol Internet. Aceasta valoare asignata de expeditor ajuta la asamblarea fragmentelor unei datagrame.

IMP

Interface Message Processor (Procesorul de Mesaje de Interfata), pachetul de schimb al ARPANET.

adresa de internet

O adresa sursa sau destinatie specifica nivelului gazda.

datagrama internet

Unitatea de date schimbata intre modulul internet si protocolul de nivel superior impreuna cu headerul de internet.

fragment internet

O portiune de date dintr-o datagrama internet impreuna cu headerul de internet

IP

Internet Protocol.

IRS

Initial Receive Sequence number (numarul de Secventa de Primire Initial). Primul numar de secventa utilizat de expeditor intr-o conexiune.

ISN

Initial Sequence Number (Numarul de Secventa Initial). Primul numar de secventa folosit intr-o conexiune, (ori ISS, ori IRS). Este selectat in urma unei proceduri bazata pe un timer.

ISS

Initial Send Sequence number (Numarul de secventa de trimitere initial). Primul numar de secventa utilizat de expeditor intr-o conexiune.

leader

Informatie de control la inceputul unui mesaj sau bloc de date. In particular, in ARPANET, informatia de control intr-un mesaj ARPANET la interfata gazdei IMP.

secventa stanga

Acesta este urmatorul numar de secventa care va fi confirmat de TCP-ul care primeste date (sau cel mai mic numar de secventa curent neconfirmat) si se refera uneori la marginea stanga a ferestrei send.

pachet local

Unitatea de transmisie intr-o retea locala.

modul

O implementare, de obicei software, a unui protocol sau unei alte proceduri.

MSL

Maximum Segment Lifetime, timpul cat poate exista un segment intr-un sistem inter-retea. Arbitrar, este stabilit la 2 minute.

octet

8 biti.

Optiuni

Un camp Optiune poate contine cateva optiuni si fiecare optiune poate avea lungimea de cativa octeti. Optiunile sunt folosite in mod special in situatiile de testare; de exemplu, pentru a transporta amprente de timp. IP si TCP ofera campuri de optiuni.

pachet

Un pachet de date cu un header care poate fi sau nu complet. Mai degraba o impachetare fizica decat una logica de date.

port

Portiunea dintr-un socket care specifica care specifica care canal de intrare sau de iesire a unui proces este asociat cu datele.

proces

Un program aflat in executie. Sursa sau destinatia de date din punctul de vedere a TCP-ului sau a altui protocol host-to-host.

PUSH

Un bit de control care nu ocupa spatiu de secventa, indicand faptul ca acest segment contine date care trebuie impinse spre utilizatorul destinatar.

RCV.NXT

primirea urmatorului numar de secventa

RCV.UP

primirea pointerului de urgenta

RCV.WND

fereastra de primire

primirea urmatorului numar de secventa

Este urmatorul numar de secventa pe care TCP-ul local asteapta sa-l primeasca.

fereastra de primire

Reprezinta numarul de secventa pe care TCP-ul local este gata sa-l primeasca. Astfel, TCP-ul local considera ca segmentele care depasesc limita RCV.NXT cu $RCV.NXT + RCV.WND - 1$ transporta date sau controale acceptate. Segmentele care contin numere de secventa aflate cu totul in afara acestei limite sunt considerate duplicate si sunt inlaturate.

RST

Un bit de control (reset), care nu ocupa spatiu de secventa, indicand faptul ca utilizatorul ar trebui sa taie conexiunea fara alte interactiuni. Destinatarul poate determina, bazandu-se pe numarul de secventa si campurile de confirmare a segmentului care soseste, daca ar trebui sa respecte comanda de reset sau ar trebui sa o ignore. In nici un caz, destinatarul unui segment continand RST nu va raspunde cu un RST.

RTP

Real Time Protocol (Protocol in Timp Real): Un protocol host-to-host pentru comunicarea informatiilor urgente.

SEG.ACK

confirmarea unui segment

SEG.LEN
lungimea unui segment

SEG.PRC
valoarea de precedenta a unui segment

SEG.SEQ
secventa unui segment

SEG.UP
campul pointerului de urgenta al unui segment

SEG.WND
campul fereastră al unui segment

segment
O unitate logica de date, in particular un segment TCP este unitatea de date transferata intre o pereche de module TCP.

confirmarea unui segment
Numarul de secventa din campul de confirmare a segmentului care soseste.

lungimea unui segment
Cantitatea de spatiu de numar de secventa ocupat de un segment, incluzand orice controale care ocupa spatiul de secventa.

secventa de segment
Numarul din campul de secventa a segmentului care soseste.

secventa send
Este urmatorul numar de secventa pe care TCP-ul expeditor local il va folosi la o conexiune. Este selectata initial dintr-o curba ISN si este incrementata pentru fiecare octet de date sau control transmis.

fereastră send
Aceasta reprezinta numerele de secventa pe care TCP-ul destinatari de la distanta este gata sa le primeasca. Este valoarea campului fereastră specificat in segmente de TCP-ul care primeste datele de la distanta. Limita numerelor de secventa noi care pot fi emise de un TCP este intre SND.NXT si SND.UNA + SND.WND - 1. (Sunt asteptate si retransmiteri ale numerelor de secventa intre SND.UNA si SND.NXT)

SND.NXT
secventa send

SND.UNA
secventa ramasa

SND.UP
pointerul de urgenta send

SND.WL1
numarul de secventa a segmentului la ultima actualizare a ferestrei

SND.WL2
numarul de confirmare a segmentului la ultima actualizare a ferestrei

SND.WND

fereastra send

socket

O adresa care include un identificator de port, adica concatenarea unei adrese internet cu un port TCP.

Adresa Sursa

Este de obicei formata din identificatorii de retea si identificatorii gazda.

SYN

Un bit de control in segmentul care soseste, care ocupa un numar de secventa, utilizat la initierea unei conexiuni pentru a indica de unde va incepe numerotarea numerelor de secventa.

TCB

Transmission control block (blocul de control a transmisiei), structura de date care inregistreaza starea unei conexiuni.

TCB.PRC

Precedenta unei conexiuni.

TCP

Transmission Control Protocol: Un protocol host-to-host pentru comunicarea de incredere in mediile inter-retea.

TOS

Type of Service, (Tip de Serviciu) , un camp al IP.

Tip de Serviciu

Un camp al Protocolului Internet care indica tipul de serviciu pentru acest segment internet.

URG

Un bit de control (urgent), care nu ocupa spatiu de secventa, folosit pentru a indica faptul ca utilizatorul ar trebui anuntat sa faca procesare rapida atata timp cat exista date de consumat cu numere de secventa mai mici decat valoarea indicata de pointerul de urgenta.

pointerul de urgenta

Un camp de control semnificativ doar daca bitul URG este setat. Acest camp comunica valoarea pointerului de urgenta care indica octetul de date asociat cu apelul urgent a utilizatorului expeditor.

REFERINTE

[1] Cerf, V., si R. Kahn, "A Protocol for Packet Network Intercommunication", IEEE Transactions on Communications, Vol. COM-22, No. 5, pp 637-648, Mai 1974.

[2] Postel, J. (ed.), "Internet Protocol - DARPA Internet Program

Protocol Specification", RFC 791, USC/Information Sciences Institute, September 1981.

- [3] Dalal, Y. and C. Sunshine, "Connection Management in Transport Protocols", Computer Networks, Vol. 2, No. 6, pp. 454-473, December 1978.
- [4] Postel, J., "Assigned Numbers", RFC 790, USC/Information Sciences Institute, September 1981.