Nivelul Transport

Furnizează proceselor de aplicaţii servicii de transfer date cap la cap. Sunt definite două protocoale la acest nivel: TCP (Transmission Control Protocol) şi UDP (User Datagram Protocol).

PORTURI ŞI SOCKET-URI

Conceptul utilizării de porturi şi socket-uri oferă posibilitatea identificării în mod uniform şi unic a conexiunilor, programelor şi sistemelor implicate în comunicaţie, independent de identificatorii proceselor respective.

Fiecare proces care necesită comunicarea cu un alt proces se identifică în cadrul unei stive TCP/IP printr-un port sau mai multe.

Un *ID port* este un număr reprezentat pe 16 biţi utilizat de către un protocol capăt la capăt pentru a identifica cărui protocol de nivel superior sau program (proces) de aplicaţie trebuie să-i livreze mesajele recepţionate.

Există două tipuri de porturi:

- Porturi consacrate: aceste porturi aparţin unor servere standard, cum ar fi serverul Telnet care foloseşte portul 23. Numerele asociate porturilor consacrate aparţin intervalului de la 1 la 1023. Cele mai multe servere necesită numai un singur port. Excepție și excepții, precum serverul FTP care foloseşte două porturi (20 şi 21). Porturile consacrate sunt controlate şi alocate de către Autoritatea de Asociere a Numerelor în Internet, IANA (Internet Assigned Number Authority) şi în cele mai multe sisteme nu pot fi alocate decât proceselor sistem sau programelor executate de utilizatorii cu drepturi speciale. Porturile consacrate permit clienţilor să găsească serverele fără a necesita informaţii de configurare suplimentare. –
- Porturi temporare: unii clienţi nu necesită numere de porturi consacrate deoarece aceştia iniţiază comunicaţiile cu serverele, iar numărul portului utilizat de fiecare dintre clienţi este conţinut în mesajul UDP/TCP transmis către server. Fiecărui proces client i se asociază un număr de port, de către sistemul pe care acesta rulează, pentru un interval de timp nedefinit, cât necesită procesul client să transfere datele. Numerele porturilor temporare au valori mai mari decât 1023, uzual fiind în intervalul de la 1024 la 65535.

Interfaţa de tip *socket* reprezintă una dintre interfeţele de programare a aplicaţiei, API (Application Programming Interfaces) cu protocoalele de comunicaţie. Deşi nu sunt standardizate, socketurile API au devenit un standard implicit pentru implementarea socketurilor reţelelor TCP/IP.

- Un socket reprezintă un tip special de identificator, care este utilizat de un proces pentru a solicita serviciile de reţea de la un sistem de operare;
- Adresa unui socket este reprezentată de tripletul: <protocol, adresă locală, port local>. Spre exemplu, în versiunea 4 a stivei TCP/IP se utilizează următoarea adresă a socketului: <tcp, 192.168.14.234, 8080>;

PROTOCOLUL UDP

UDP este doar o interfaţă de aplicaţie pentru IP şi nu serveşte decât pentru multiplexarea / demultiplexarea fluxurilor de aplicaţie, transmiterea şi recepţionarea datagramelor, utilizarea

porturilor pentru redirectarea datagramelor.

Mesajul UDP se compune dintr-un antet relativ redus (8 octeţi) şi câmpul de date. Antetul precizează porturile de protocol ale sursei şi destinaţiei între care se face transferul mesajului, lungimea totală a mesajului UDP (inclusiv antetul) măsurată în octeţi.

Specificarea portului sursă este opţională. Acest port va fi utilizat ca port destinaţie pentru datagramele de răspuns. Dacă nu se specifică acest port câmpul respectiv din antet se completează cu biţi 0. Este opţională, de asemenea, şi utilizarea secvenţei de verificare care, în caz că se foloseşte, ia în considerare nu numai antetul (ca la IP) ci şi câmpul de date.

Lungimea mesajului UDP este calculată în număr de octeţi şi include antetul.

PROTOCOLUL TCP

Asigură un serviciu cu conexiune fiabil, cu livrarea mesajelor la recepţie în ordinea în care au fost emise. Cele mai multe protocoale de aplicaţie utilizează TCP, cum ar fi Telnet sau FTP.

Mesajele transferate la nivelul transport între două sisteme prin protocolul TCP, se numesc segmente.

Fiecare segment se compune dintr-un antet urmat de date. Antetul conţine informaţie de identificare şi informaţie de control. Primele două câmpuri, port sursă şi port destinaţie, conţin numerele porturilor TCP ce identifică cele două programe de aplicaţie de la capetele conexiunii.

Numărul de secvenţă reprezintă numărul de ordine, în secvenţă, al primului octet din câmpul de date. Dacă bitul de control SYN este fixat la valoarea 1, atunci numărul de secvenţă reprezintă numărul iniţial de secvenţă (n), iar primul octet de date este n+1.

Numărul confirmat reprezintă numărul de secvenţă al octetului de date ce se aşteaptă a fi recepţionat. Dacă bitul de control ACK este fixat la valoarea 1, acest câmp conţine valoarea numărului de secvenţă următor care se aşteaptă a fi recepţionat. Trebuie remarcat că numărul de secvenţă se referă la fluxul datelor ce se transmit în acelaşi sens cu segmentul respectiv, iar numărul confirmat se referă la fluxul datelor transmise în sens invers.

Lungimea antetului se exprimă printr-un întreg care reprezintă numărul cuvintelor de 32 biţi din care este constituit antetul. Este necesar să se indice lungimea antetului deoarece lungimea câmpului rezervat pentru specificarea opţiunilor este variabilă.

Există mai multe tipuri de segmente, funcţie de scopul în care sunt folosite: transfer date, stabilire conexiune, eliberare conexiune, numai pentru confirmări (fără transfer date), transfer date în regim de urgenţă etc.

Specificarea tipului de segment se face prin biţii notaţi U, A, P, R, S şi F. Aceşti biţi au următoarele semnificaţii:

- U (URG) – specifică semnificaţia importantă a câmpului de pointer urgent în cadrul acestui segment;
- A (ACK) – specifică semnificaţia importantă a câmpului de confirmare în cadrul acestui segment;
- P (PSH) – Funcţia de forţare (push) a segmentelor. În unele situaţii, o aplicaţie trebuie să fie informată că toate datele transferate la nivelul TCP au fost transmise către destinaţie. Din acest motiv s-a introdus funcţia de forţare a transmisiei segmentelor TCP rămase încă în unităţile de memorie, către sistemul destinaţie. Închiderea normală a conexiunii va determina deasemenea forţarea datelor către destinaţie;
- R (RST) – Resetarea conexiunii;
- S (SYN) – Sincronizează numerele de secvenţă;
- F (FIN) – Oprirea transferului de date de la emiţător.

Câmpul “pointer urgent” indică poziţia datelor care se transmit în regim de urgenţă în fluxul general al datelor.

În câmpul “fereastră” se specifică numărul de octeţi, începând cu cel menţionat în câmpul de confirmare, pe care transmiţătorul îi poate accepta (pe sensul invers de transmisiune).

Protocolul TCP foloseşte câmpul “opţiuni” pentru negocieri între cele două capete ale conexiunii.

Există şapte opţiuni posibile pentru segmentele TCP:

- Tip optiune = 0: Sfârşitul listei de opţiuni
- Tip optiune = 1: Fără operaţii
- Tip optiune = 2: Dimensiunea maximă a segmentului
- Tip optiune = 3: Ajustarea ferestrei
- Tip optiune = 4: Permiterea SACK.
- Tip optiune = 5: SACK
- Tip optiune = 8: Etichete temporale

Opţiunea de specificare a dimensiunii maxime a segmentului. Această opţiune este utilizată numai pe durata stabilirii conexiunii (bitul de control SYN fiind setat) şi cererea este transmisă de către capătul care urmează să recepţioneze datele, pentru a indica lungimea maximă a segmentului pe care acest sistem o poate suporta. Dacă această opţiune nu este utilizată atunci este permisă orice dimensiune a segmentului.

În ceea ce priveşte dimensiunea segmentelor, teoretic valoarea optimă a acesteia este determinată de dimensiunea maximă admisă de reţea pentru pachetele IP, pe calea de la sursă la destinaţie, aşa încât să nu fie necesară fragmentarea segmentelor. Fragmentarea segmentelor conduce la formarea de pachete ce corespund aceluiaşi segment dar care sunt în mod independent tratate de reţea. Toate fragmentele trebuie să ajungă la destinaţie, altfel trebuie să se retransmită întreg segmentul. Deoarece probabilitatea de pierdere (şi rejectare) a unui fragment (pachet IP) nu este zero, creşterea dimensiunii segmentului peste pragul de fragmentare conduce la multiplicarea situaţiilor în care este necesară retransmiterea segmentelor. Desigur, segmentele retransmise pot avea dimensiuni diferite în raport cu cele iniţiale, acesta fiind un motiv suplimentar pentru care în TCP confirmarea se face nu la nivel de segment ci la nivel de octet.

Opţiunea de ajustare a ferestrei. Ambele capete ale conexiunii trebuie să transmită opţiunea de scalare a ferestrei în cadrul segmentelor SYN pentru a permite ajustarea dimensiunii ferestrei în sensul de transmisie. Transmiterea acestei opţiuni extinde dimensiunea ferestrei TCP la o reprezentare pe 32 de biţi. Dimensiunea ferestrei reprezentată pe 32 de biţi este definită prin intermediul unui factor de scalare (transmis în segmentul SYN) faţă de fereastra standard reprezentată pe 16 biţi. Receptorul acestui mesaj modifică dimensiunea ferestrei de la valoarea standard pe 16 biţi prin adăugarea factorului de scalare. Această opţiune poate fi utilizată numai în faza de iniţializare a conexiunii. Astfel, dimensiunea ferestrei nu se mai poate schimba pe toată durata menţinerii conexiunii.

Opţiunea de permitere a SACK. Această opţiune este introdusă numai atunci când pe conexiunea TCP respectivă se utilizează confirmări selective (SACK – Selective ACKnowledgement). Pentru această opţiune câmpul de date opţiune nu este utilizat (se transmit numai tipul opţiunii şi lungimea).

Opţiunea SACK. Confirmarea selectivă SACK permite receptorului să informeze transmiţătorul asupra tuturor segmentelor recepţionate corect. Astfel, transmiţătorul va retransmite numai segmentele pierdute. În cazul în care numărul de segmente pierdute, înregistrat de la ultimul SACK transmis, este foarte mare atunci şi dimensiunea opţiunilor SACK va fi foarte mare. Din acest motiv numărul de blocuri de segmente care pot fi raportate de către o opţiune SACK este limitat la patru.

Opţiunea etichetelor temporale (TS). Această opţiune transmite o valoare a etichetei temporale TS (Time Stamp) care specifică valoarea curentă a timpului indicat de ceasul local de la transmiţător. Valoarea “TS ecou” este utilizată dacă bitul ACK este setat cu 1 logic în antetul TCP.

Stabilirea conexiunilor TCP

Înainte de a începe transferul datelor între cele două procese de utilizator (programe de aplicaţie) se stabileşte, prin intermediul reţelei, o conexiune logică numită circuit virtual. Cele două procese ajung, prin modulele destinate protocolului de comunicaţie, să îşi transmită mesaje prin reţea pentru a verifica dacă transferul este autorizat (acceptat) şi dacă ambele părţi sunt gata pentru acest transfer.

Conexiunea fiind stabilită, cele două programe de aplicaţie sunt informate că transferul datelor poate să înceapă. Programul de aplicaţie transmiţător transferă datele spre nivelul transport sub forma unui flux de biţi, împărţit în octeţi. La nivelul transport fluxul octeţilor primiţi de la programe de aplicaţie este împărţit în segmente, care sunt apoi transmise în reţea spre destinaţie. Fiecare segment, format dintr-un număr oarecare de octeţi, este transportat în reţea de un pachet IP. Conexiunile asigurate de TCP/IP la acest nivel sunt conexiuni duplex, ceea ce înseamnă că cele două procese îşi pot transmite date simultan unul către celălalt. Un avantaj al acestei conexiuni duplex constă în faptul că se poate transmite informaţia de control (al erorii, al fluxului) înapoi către sursă în pachete ce transferă datele în sens opus, reducându-se astfel traficul în reţea (metoda piggy-back).

Pentru controlul fluxului şi al erorii se utilizează temporizatoare la emisie şi confirmări pozitive la recepţie, ACK (Positive acknowledgement).

Atunci când se transmite un segment cu un anumit număr de secvenţă se porneşte la emisie un timer. Dacă până expiră timerul nu soseşte o confirmare ACK, atunci segmentul este retransmis automat. Deoarece protocolul TCP oferă un serviciu cu conexiune, conexiunea ce se stabileşte pentru transferul datelor este identificată printr-o pereche de puncte de capăt (porturi de protocol). În felul acesta, acelaşi port de protocol poate fi simultan capăt al mai multor conexiuni.

Spre exemplu, un sistem poate permite accesul concurenţial la serviciul său de poştă electronică, acceptând pe un acelaşi port de protocol mesajele transmise simultan de la mai multe calculatoare, cu fiecare dintre acestea având stabilită o altă conexiune, identificată distinct de celelalte. Protocolul TCP realizează multiplexarea conexiunilor pe baza numerelor porturilor, aşa cum se realizează şi la UDP.

Pentru stabilirea conexiunii, un proces (serverul, de obicei) transmite o cerere de deschidere pasivă a conexiunii (passive OPEN), iar celălalt proces transmite o cerere activă (active OPEN). Cererea pasivă nu este luată în considerare până când celălalt proces nu încearcă să se conecteze la primul printr-o cerere activă

Această procedură de stabilire a conexiunii este cunoscută sub numele de "*three-way handshake*". Segmentele TCP schimbate la stabilirea conexiunii includ şi valorile iniţiale ale numerelor de secvenţă de la ambele capete, care vor fi utilizate pentru următoarele transmisii de date. Astfel, sincronizarea numerelor de secvenţă se realizează odată cu stabilirea conexiunii.

Închiderea unei conexiuni se realizează prin transmiterea unui segment TCP care are bitul FIN (nu mai urmează alte date) setat cu 1. Deoarece conexiunea este bidirecţională simultan (full-duplex) atunci segmentul FIN încheie transmisiunea datelor numai pentru un sens al conexiunii. Celălalt proces va transmite datele rămase şi va încheia transmisiunea la rândul său cu un segment TCP care are bitul FIN setat cu 1. O conexiune se consideră a fi închisă numai atunci se încheie transmisiunea în ambele sensuri.

Diferitele stări ale unei conexiuni TCP sunt enumerate în continuare:

- LISTEN: Aşteptarea unei cereri de conexiune de la un alt nivel TCP;
- SYN-SENT: S-a transmis un segment de sincronizare SYN, iar TCP aşteaptă un segment SYN de răspuns;
- SYN-RECEIVED: S-a recepţionat un segment SYN, s-a transmis un segment SYN, iar TCP aşteaptă un segment de confirmare ACK;
- ESTABLISHED: Schimbul celor trei mesaje de stabilire a conexiunii s-a încheiat;
- FIN-WAIT-1: Aplicaţia locală a emis o cerere de închidere CLOSE. TCP a transmis FIN şi aşteaptă un ACK sau un FIN;
- FIN-WAIT-2: S-a transmis un FIN şi s-a recepţionat un ACK. TCP aşteaptă un FIN de la nivelul TCP corespondent;
- CLOSE-WAIT: TCP a recepţionat un FIN şi a transmis un ACK. Nivelul TCP aşteaptă o cerere ce închidere de la aplicaţia locală înainte de a transmite FIN;
- CLOSING: S-a transmis un FIN, s-a recepţionat un FIN şi s-a transmis un ACK. TCP aşteaptă un ACK pentru segmentul FIN pe care l-a transmis;

- LAST-ACK: S-a recepţionat un FIN şi un ACK, iar un segment FIN a fost transmis. TCP aşteaptă un ACK;
- TIME-WAIT: FIN a fost recepţionat şi confirmat cu ACK;
- CLOSED: Indică faptul că o conexiune a fost eliminată din tabela de conexiuni.

Controlul fluxului şi al erorii cu fereastra glisantă

Livrarea la destinaţie a fluxului de octeţi în ordinea în care aceştia au fost emişi, fără pierderi şi fără duplicate, se asigură prin folosirea unei tehnici de confirmare pozitivă cu retransmitere, combinată cu fereastră glisantă. Mecanismul ferestrei glisante utilizat de TCP, operând la nivelul octeţilor şi nu al segmentelor, permite atât o transmisiune eficientă, cât şi un control al fluxului.

Octeţii din fluxul datelor primite de la programul aplicaţie sunt numerotaţi în ordine. Transmiţătorul, să-l notăm A, emite continuu segmente, formate cu aceşti octeţi, către celălalt capăt al conexiunii, să-l notăm B, urmărind în acelaşi timp confirmările venite în sens invers în chiar segmentele de date transmise de B către A. O confirmare venită de la capătul B specifică numărul de ordine (de secvenţă) al primului octet din segmentul pe care acesta (capătul B) îl aşteaptă, ceea ce înseamnă, implicit, o confirmare pentru recepţia tuturor octeţilor anteriori transmişi de A. Numerele de ordine ale octeţilor pe care îi poate emite transmiţătorul fără a avea confirmarea de recepţie a lor sunt specificate de fereastra glisantă.

Pentru o conexiune transmiţătorul alocă trei valori care vor preciza în orice moment poziţia ferestrei glisante:

- Valoarea 1 marchează începutul ferestrei glisante (limita inferioară), separând octeţii emişi şi pentru care s-au primit confirmările de recepţie de cei pentru care nu s-au primit confirmări.
- Valoarea 2 indică sfârşitul ferestrei (limita superioară), adică numărul de ordine al ultimului octet ce poate fi inclus într-un segment ce urmează a fi transmis fără să mai vină vreo confirmare de recepţie.
- Valoarea 3 marchează în interiorul ferestrei limita ce separă octeţii deja transmişi de cei încă netransmişi.

Deasemenea, receptorul va folosi o fereastră asemănătoare pentru a reconstitui fluxul octeţilor emişi. În acelaşi timp, având în vedere că protocolul TCP stabileşte conexiuni duplex şi că pe fiecare sens de transmisiune se utilizează câte o fereastră de emisie şi una de recepţie, rezultă că în fiecare capăt al conexiunii vor exista două ferestre, una glisând pe fluxul datelor ce se emit iar cealaltă pe fluxul datelor ce se recepţionează.

Protocolul TCP permite modificarea dimensiunii ferestrei glisante pentru a reliza un control al fluxului. Fiecare confirmare, pe lângă specificarea numărului de octeţi recepţionaţi (în ordine), conţine şi o indicaţie privind numărul de octeţi pe care receptorul îi poate accepta, dependent de mărimea spaţiului liber din memoria tampon a acestuia. Transmiţătorul îşi va modifica dimensiunea ferestrei de emisie corespunzător acestei indicaţii a receptorului, ceea ce va conduce şi la creşterea eficienţei transmisiei prin reducerea simţitoare a numărului pachetelor pierdute, rejectate de receptor şi, în consecinţă, reducerea şi a numărului retransmiterilor.

Algoritmi de control al congestiei în TCP

O diferenţă esenţială dintre UDP şi TCP este că TCP utilizează un algoritm de control al congestiei. Algoritmul de control al congestiei în TCP previne situaţia în care transmiţătorul ar depăşi capacitatea de prelucrare a reţelei. Astfel, TCP poate adapta debitul transmiţătorului în funcţie de capacitatea reţelei, astfel încât să se evite posibilele situaţii de congestie şi deci, pierderea segmentelor. Dealungul timpului s-au adăugat şi s-au propus mai multe îmbunătăţiri ale TCP cu scopul de a controla congestiile. Această căutare de algoritmi de evitare a congestiilor este încă o direcţie de cercetare deschisă, dar implementările moderne ale TCP conţin patru algoritmi de bază pentru Internet (de obicei utilizându-se variante combinate ale acestora):

- iniţializare lentă;
- evitarea congestiei;
- retransmisie rapidă;
- recuperare rapidă.

Iniţializarea lentă (Slow start). Implementările TCP mai vechi deschid o conexiune prin introducerea de la transmiţător a mai multor segmente în reţea, până când se atinge dimensiunea ferestrei anunţate de receptor. Deşi totul este în regulă atunci când sistemele se află în acelaşi LAN, atunci când există routere între cele două sisteme şi se utilizează legături lente atunci pot apărea probleme. Unele routere intermediare nu pot face faţă situaţiei şi atunci pachetele se pierd, ceea ce determină apariţia retransmisiilor şi deci, reducerea performanţelor. Algoritmul utilizat pentru evitarea acestei situaţii poartă numele de iniţializare lentă. Acesta operează prin observarea faptului că rata cu care se introduc noi pachete în reţea ar trebui să fie egală cu rata sosirii confirmărilor de pe celălalt sens. Iniţializarea lentă mai adaugă o fereastră la nivelul TCP transmiţător: *fereastra de congestie*. Atunci când se stabileşte o nouă conexiune cu un sistem dintr-o altă reţea se iniţializează fereastra de congestie cu dimensiunea de un segment (spre exemplu, dimensiunea segmentului anunţată de celălalt capăt sau implicit cu o valoare tipică de 536 sau 512). De fiecare dată când se recepţionează un segment ACK dimensiunea ferestrei de congestie creşte cu un segment. Transmiţatorul poate transmite cea mai mică dimensiune a ferestrei de congestie sau pe cea anunţată. Fereastra de congestie este impusă de controlul de flux de la transmiţător, în timp ce fereastra anunţată este impusă de controlul de flux de la receptor. Prima dintre acestea două se bazează pe evaluarea făcută de către transmiţător a congestiei percepute în reţea, iar a doua este legată de numărul de locaţii libere ale cozilor de aşteptare de la recepţie, pentru această conexiune.

Transmiţătorul începe prin a transmite un segment şi apoi aşteaptă confirmarea ACK pentru acesta. Atunci când soseşte confirmarea dimensiunea ferestrei de congestie este incrementată de la unu la doi, iar apoi se pot transmite două segmente. Atunci când ambele segmente sunt confirmate, dimensiunea ferestrei de congestie se va incrementa la patru. Acest mecanism determină o creştere exponenţială a traficului, deşi această creştere nu este tocmai exponenţială deoarece receptorul trebuie să întârzie confirmările sale, în mod uzual, prin transmiterea unei ACK pentru fiecare două segmente pe care le recepţionează. La un anumit moment dat, capacitatea reţelei IP este atinsă, iar ruterul intermediar va începe să arunce pachete. Acest lucru va specifica transmiţătorului că fereastra sa de congestie a devenit prea mare.

Evitarea congestiei. Presupunerea care se face în cazul acestui algoritm este că pierderea pachetelor cauzată de reţea este foarte mică (mult mai mică de 1%). Astfel, pierderea unui pachet va determina semnalizarea congestiei undeva în reţea, între sursă şi destinaţie. Există două indicatoare ale pierderii unui pachet:

- expirarea unui temporizator;
- recepţionarea unor confirmări ACK duplicat.

Evitarea congestiei şi deschiderea lentă sunt algoritmi independenţi, care au obiective diferite. Totuşi, atunci când se produce o congestie, TCP va trebui să scadă rata de transmisiune în reţea a segmentelor şi să utilizeze iniţializarea lentă pentru a reporni transmisiunea. În practică aceşti doi algoritmi sunt implementaţi împreună.

Evitarea congestiei şi iniţializarea lentă necesită menţinerea a două variabile pentru fiecare conexiune:

- o fereastră de congestie, notată cu *cwnd*;
- un prag de iniţializare lentă, notat cu *ssthresh*.

Algoritmul rezultat prin combinarea celor doi algoritmi de mai sus funcţionează astfel:

1. Iniţializarea unei anumite conexiuni presupune setarea cwnd cu valoarea de un segment şi ssthresh cu valoarea 65535 octeţi;
2. Rutina TCP de ieşire nu transmite niciodată mai mult decât valoarea cea mai mică dintre cwnd şi fereastra anunţată de receptor;

3. Atunci când se produce o congestie (expirare timp sau ACK duplicat), jumătate din dimensiunea ferestrei curente este salvată în ssthresh. În plus, dacă aceeaşi congestie e indicată de expirarea timpului, cwnd este setat la un segment;
4. 4. Atunci când datele noi sunt confirmate de către celălalt capăt se creşte cwnd, dar valoarea cu care creşte aceasta depinde dacă TCP realizează o iniţializare lentă sau evitarea congestiei. Dacă cwnd este mai mic sau egal cu ssthresh, atunci TCP lucrează cu iniţializarea lentă; în caz contrar, TCP rulează cu evitarea congestiei.

Se utilizează iniţializarea lentă până când se ajunge la jumătate din valoarea la care ajunsese când s-a produs congestia (deoarece a memorat jumătate din dimensiunea ferestrei care a cauzat problema, în pasul 3), iar apoi se comută pe modul de evitare a congestiei. Iniţializarea lentă începe cu cwnd având valoarea de un segment, iar apoi se incrementează cu un segment la recepţionarea fiecărei ACK. Aşa cum s-a menţionat anterior, fereastra creşte exponenţial: se trimite un segment, apoi două, apoi patru, etc. Evitarea congestiei stabileşte ca cwnd să fie incrementată cu dim_segm* dim_segm /cwnd de fiecare dată când se recepţionează o ACK, unde dim_segm reprezintă dimensiunea segmentului, iar cwnd se măsoară în octeţi. Aceasta este o creştere liniară a cwnd, prin comparaţie cu creşterea exponenţială în cazul iniţializării lente. Creşterea cwnd ar trebui să fie cel mult cu un segment pentru fiecare interval de propagare dus-întors (indiferent câte ACK sunt recepţionate în acest interval), în timp ce iniţializarea lentă incrementează cwnd cu numărul de ACK recepţionare în acest interval de propagare dus-întors.

Retransmisie rapidă. Algoritmul de retransmisie rapidă evită ca TCP să aştepte expirarea unui temporizator pentru a retransmite segmentele pierdute. În 1990 s-au propus nişte modificări pentru algoritmul de evitare a congestiei. Înainte de a descrie modificările făcute trebuie făcută observaţia că TCP poate genera imediat o confirmare ACK duplicat atunci când se recepţionează un segment care nu respectă ordinea în secvenţă. Această ACK duplicat nu ar trebui întârziată. Scopul acestei ACK duplicat este de a informa celălalt capăt asupra factului că un segment a fost recepţionat în afara ordinii din secvenţă şi să specifice care este numărul de secvenţă aşteptat.

Deoarece TCP nu poate identifica dacă o ACK duplicat este cauzată de pierderea unui segment sau doar de o reordonare a segmentelor, acesta se va aştepta să recepţioneze un număr redus de confirmări ACK duplicate. Se presupune că dacă este doar cazul unei reordonări a segmentelor, atunci se vor produce una sau două ACK duplicate până când va fi procesat segmentul reordonat, care va genera şi el o nouă ACK. Dacă se recepţionează trei sau mai multe ACK duplicate în ordine, aceasta înseamnă că un segent a fost pierdut. Atunci, TCP va retransmite segmentele ce par să lipsească, fără a mai aştepta expirarea temporizatorului de retransmisie.

Recuperare rapidă. După ce algoritmul de retransmisie rapidă decide transmiterea segmentului care aparent lipseşte, se utilizează algoritmul de evitare a congestiei; în nici un caz, nu se va utiliza iniţializarea lentă. Acesta este algoritmul cu numele de recuperare rapidă. Acesta determină o îmbunătăţire a performanţelor deoarece permite un debit mare în condiţii de congestii moderate, în special pentru ferestre mari.

Motivul pentru care nu se utilizează algoritmul de iniţializare lentă în acest caz este acela că recepţia unor ACK duplicate informează TCP mai mult decât pierderea unui segment. Deoarece receptorul poate genera un ACK duplicat numai dacă un alt segment este recepţionat, acel segment a fost scos din reţea şi se află în registrul receptorului. Astfel, în acest moment mai există date se ce propagă între cele două capete şi este de dorit ca TCP să nu reducă viteza brusc prin utilizarea iniţializării rapide. Algoritmii de retransmisie rapidă şi recuperare rapidă sunt de obicei implementaţi împreună, după cum urmează:

1. Atunci când un al treilea ACK duplicat este recepţionat la rând se va seta ssthresh la jumătate din fereastra curentă de congestie, cwnd, dar nu mai puţin de două segmente. În continuare se va retransmite segmentul lipsă. Se setează cwnd cu valoarea din ssthresh plus de trei ori dimensiunea segmentului. Această procedură conduce la creşterea ferestrei de congestie cu numărul de segmente care au fost scoase din reţea, fiind memorate la celălalt capăt (în etapa 3).

2. De fiecare dată când soseşte un ACK duplicat se incrementează cwnd cu dimensiunea segmentului. Această procedură conduce la creşterea ferestrei de congestie cu dimensiunea segmentului care a fost scos din reţea. Se transmite un segment dacă acest lucru este permis de noua valoare din cwnd.
3. Atunci când soseşte noua ACK care confirmă noile date, se setează cwnd cu valoarea din ssthresh (valoarea setată în etapa 1). Această ACK este confirmarea pentru retransmisia din etapa 1, care soseşte după un timp de propagare dus-întors de la acea retransmisie. În plus, această ACK confirmă toate segmentele transmise între timp, între momentul pierderii segmentului şi recepţia primei ACK duplicat. În această etapă se utilizează evitarea congestiei, deoarece TCP a scăzut viteza la jumătate faţă de valoarea pe care o avea în momentul pierderii segmentului.