Dirijarea cu vectori distanta

In reţelele de calculatoare se prefer folosirea algoritmilor dinamici de dirijare în locul celor statici, deoarece algoritmii statici nu ţin seama de încărcarea curentă a reţelei. Un exemplu e algoritmul de dirijare cu vectori distanţă.

Algoritmul de dirijare cu vectori distanţă (distance vector routing) presupune că fiecare ruter menţine o tabelă (de exemplu un vector) care păstrează cea mai bună distanţă cunoscută spre fiecare destinaţie şi linia care trebuie urmată pentru a ajunge acolo. Aceste tabele sunt actualizate prin schimbul de informaţii între nodurile vecine.

Cunoscuti si sub alte nume: algoritmul distribuit de dirijare BellmanFord şi algoritmul Ford-Fulkerson.

Metrici

În dirijarea pe baza vectorilor distanţă, fiecare ruter păstrează o tabelă de dirijare conţinând câte o intrare pentru fiecare ruter din subreţea.

Această intrare are două părţi: linia de ieşire preferată care se foloseşte pentru destinaţia respectivă şi o estimare a timpului sau distanţei până la acea destinaţie.

Metrica folosită poate fi numărul de salturi, întârzierea în milisecunde, numărul total de pachete care aşteaptă în cozi de-a lungul căii, sau ceva asemănător.

Se presupune că ruterul cunoaşte „distanţa” spre fiecare dintre vecinii săi. Dacă se foloseşte metrica salturilor, distanţa este doar de un salt. Dacă metrica folosită este cea a lungimii cozilor de aşteptare, ruterul examinează pur şi simplu lungimile acestor cozi. Dacă metrica este cea a întârzierilor, ruterul o poate măsura direct prin pachete speciale ECHO, în care receptorul va marca doar timpul curent (ştampila de timp) şi le va trimite înapoi cât mai repede posibil

Exemplu

Presupunem ca se foloseste metrica intazierilor, astfel routerul va cunoaste intarzierea spre fiecare dintre vecinii sai. La un interval de timp deltaT fiecare router trimite spre fiecare vecin o lista a estimarilor timpilor spre fiecare destinatie. De asemenea, routerul current va trimite cate un pachet catre fiecare vecin pentru a vedea timpul actual pe care l-ar face transmiterea unui pachet. Pentru a vedea ce cale allege, se va uita in tabelele primite si face suma dintre timpul nou rezultat din trimiterea pachetului spre un vecin si timpul pe care vecinul il face pana la destinatie. Routerul isi va actualize tabela, va allege minimul dintre aceste valori pentru pachetul current si il va trimite catre vecinul indicat.

Făcând aceste calcule pentru fiecare vecin, un ruter poate stabili care estimare pare a fi cea mai bună, pentru a folosi această estimare, împreună cu linia corespunzătoare în noua tabelă de dirijare. Este de remarcat faptul că vechea tabelă de dirijare nu intervine în calcule.

Inundarea

Cand un pachet ajunge intr-un router, se fac atatea copii cati vecini are si copiile sunt trimise catre vecini.

Inundarea este un alg static cunoscut si sub numele de flooding. Acesta presupune ca fiecare pachet recepţionat este trimis mai departe pe fiecare linie de ieşire, cu excepţia celei pe care a sosit. O masura este tehnica autodistrugerii: existenta unui contor in antet, care la fiecare nou salt este decrementat. Ideal ar fi ca acest contor să fie iniţializat cu lungimea căii de la sursă la destinaţie. Dacă emiţătorul nu cunoaşte lungimea căii, poate iniţializa contorul la valoarea cea mai defavorabilă, adică diametrul subreţelei. Cand ajunge la valoarea 0 pachetul este distrus si vom sti ca cel putin un pachet a ajuns la destinatie.

Este folosit in aplicatiile militare, pentru ca in orice moment poate fi scos un numar mare de routere, iar robustetea inundarii e necesara.

Dirijarea centralizata

Putem sa ne uitam de sus la reteaua noatra sis a ne uitam de sus care sunt valorile fixe, teoretice la reteaua noastra si putem aplica pe acea retea un alg de tip Dijkstra care nu ar insemna decat ca avem un alg centralizat. Determina de la fiecare pct catre toate elem dintr o retea care este drumul minim. Ideea principala ppe care se bazeaza alg de tip Dijkstra e ca daca intre I si J avem un element k si care se afla pe drumul optim de la I la j atunci k j si I k sunt drumurile optime pe care trebuie sa le parcurga un pachet. Ii trimite fiecarui nod intern din retea o tabela de valori care ii spune “Daca iti vine un pachet care are o anumita destinatie, el ttrebuie trimis prin vecinul X”(determinat din tabelul respectiv) impreuna cu distant ape care o are de parcurs.

Ce probleme are o dirijare centralizata☹care sunt deficientele principale:

1. Ea depinde de centrul ded control. Acel supercomputer calculeaza tot, el pastreaza toata informatia. Daca el se defecteaza el nu mai poate calcula nici distantele si nici sa trimita routerului care e drumul pe care trebuie sa o ia pe un pachet anume. Poti sa distrugi o retea distrugand centrul de comanda – principal vulnerabilitate. (sol: dublarea centrului)
2. Daca la fiecare moment eu trimit catre toate nodurile tabele ca sa vada pe unde trebuie sa mearga, se va supraincarca reteaua

Dirijarea distribuita

Este bazat pe algoritmul lui Dijkstra, d ar o varianta inversata: vom calcula care sunt drumurile minime de la toate nodurile catre o anumita destinatie. In acest caz, routerul trebuie sa aiba o memorie foarte mare. Este convergent, intrucat asigura gasirea drumului minim intr-un nr finit de pasi. Stim cum toate nodurile din retea pot ajunge catre nodul sursa??. Pachetul nostru primeste destinatia si calculeaza pt toate nodurile dinr eteaua noastra la un moment dat care sunt val minime catre dest sip e und ear trebui sa mearga. Stim pentru fiecare nod prin care vecin trebuie sa treaca pt a ajunfge la destinatie si disttanta minima. Trebuie la fiecare moment sa trimita la fiecare router ca sa stim toate calculele pe care trb sa le facem.

Problema:

Ce facem cand un nod e adaugat sau sters din retea: trebuie recalculate toate dimensiunile si valorile distantelor prin care un pachet trb sa treaca.

Poate fi utilizat doar pentru datagrame, deoarece: 1. sint posibile modificari ale cailor, pe durata transmiterii pachetelor, astfel incit pachetele trimise pe o cale pot ajunge pe o alta cale; 2. - pe durata intervalului de convergenta, algoritmul nu evita producerea buclelor (trecerea pachetelor de mai multe ori prin acelasi nod).

Pentru a putea realiza calculele, fiecare nod al retelei pastreaza trei tablouri:

C - tabloul distantelor; C[d][v] este lungimea (sau costul) drumului de la nodul curent la nodul destinatar,prin nodul vecin v;

D - tabloul distantelor minime; D[d] este lungimea drumului minim de la nodul curent la nodul destinatar d;

V - tabloul de dirijare; V[d] este nodul vecin prin care se transmit datele,pe drumul minim,spre destinatarul d.

La producerea unei modificari,fiecare nod care o sesizeaza actualizeaza tabloul distantelor minime si de dirijare.

Toate modificarile tabelei de dirijare sint comunicate vecinilor, in forma unor mesaje de forma (s,d,Dsd), unde Dsd este distanta minima de la s la d. In cazul adaugarii unei noi legaturi la retea, noului vecin i se transmite toata tabela de dirijare. Mesajele mentionate pot fi transmise izolat sau grupate in aceeasi unitate. Primirea unui mesaj de forma (s,d,Dsd) declanseaza in receptor procesul de calcul si actualizare a tabelei de dirijare, modificarile fiind transmise, la rindul lor, tuturor vecinilor. Procesul este convergent, la un moment dat, schimbul de mesaje si actualizarea tabelelor incheindu-se

Dirijarea ierarhica

Acum retelele sunt foarte mari. Daca vrem sa Trimitem un pachet din Bucuresti in Brasilia, o retea de aceasta forma trebuie sa fie foarte mare, poate contine toate nodurile din lume. Ca sa incercam sa aplicam oricare din algoritmi, ar insemna sa avem un ssitem centralizat care sa tina cont de toata reteaua mondiala iar fiecare router sa aiba atatea intrari cate routere sunt in toata lumea. Daca apar erori nu vor fi usor de reparat.

Sol: nu facem calcule pentru retele mari la nivel global, vom face calculi la nivel local. Calculele se fac la nivelul unei retele.

Pe măsură ce reţelele cresc în dimensiune, tabelele de dirijare cresc proporţional. Pe lângă faptul că memoria ruterului este consumată de fiecare nouă creştere a tabelelor, pentru parcurgerea lor este necesar tot mai mult timp de calcul şi se foloseşte tot mai mult din lăţimea de bandă pentru a trimite rapoartele de stare despre ele. La un moment dat, reţeaua poate creşte până la un punct în care nu mai este posibil ca fiecare ruter să deţină o intrare pentru fiecare alt ruter, astfel încât dirijarea trebuie făcută ierarhic

Ruterele sunt împărţite în ceea ce vom numi regiuni (regions), fiecare ruter ştiind toate detaliile necesare pentru a dirija pachete spre destinaţie în cadrul regiunii sale, dar neştiind nimic despre organizarea internă a celorlalte regiuni. Când reţele diferite sunt interconectate, este natural să privim fiecare reţea ca pe o regiune, pentru a elibera ruterele dintr-o reţea de sarcina de a cunoaşte structura topologică a celorlalte.

Stop-and-Wait

Tampon infinit:

Ceea ce adduce nou acest protocol fata de protocolul 1 este ca nu mai consideram ca receptorul are un tampon infinit in care sunt memorate cadrele sosite cat timp isi asteapta randul.

Totuşi, se presupune în continuare că nu se produc erori pe canalul de comunicaţie şi că traficul de date este încă simplex. Emiţătorul lucrează la nivelul legătură de date al maşinii sursă, iar receptorul la nivelul legătură de date al maşinii de destinaţie. Cele două niveluri reţea, de transmisie şi de recepţie, sunt considerate tot timpul pregătite.

Problema rezolvata:

Principala problema care trebuie rezolvata este să evităm ca emiţătorul să inunde receptorul cu date care sosesc mai rapid decât poate acesta să prelucreze. Daca receptorul are nevoie de un timp ∆t ca să execute from\_physical\_layer şi to\_network\_layer, atunci emiţătorul trebuie să transmită la o viteză medie mai mică de un cadru la fiecare interval de timp de ∆t. Mai mult, dacă presupunem că echipamentul receptor nu realizează automat memorarea în zona tampon şi gestiunea cozii de aşteptare, atunci emiţătorul nu trebuie să transmită niciodată un nou cadru până când cel vechi nu a fost preluat de rutina from\_physical\_layer, ca nu cumva cel nou să se scrie peste cel vechi.

Sol 1: O solutie ar fi introducerea unei intarzieri, astfel incat emitatorul sa fie incetinit incat sa fie evitata inundarea receptorului. Insa aceasta intarziere intre sosirea cadrului si inceputul prelucrarii lui poate varia considerabil. Dacă cei ce proiectează reţele pot calcula comportamentul receptorului în cazul cel mai

defavorabil, atunci pot programa emiţătorul să transmită atât de încet, încât, chiar dacă fiecare cadru

va suferi întârzierea maximă, nu vor exista depăşiri. Ea conduce la o utilizare a lărgimii de bandă care este cu mult sub optim, cuexcepţia situaţiei în care cazurile cel mai favorabil şi cel mai defavorabil sunt aproape la fel.

Sol 2: O soluţie mult buna a acestei probleme este ca receptorul să furnizeze o reacţie către emiţător. După trimiterea unui pachet către nivelul său reţea, receptorul trimite un mic cadru fictiv către emiţător prin care anunta emitatorul să transmită următorul cadru. După ce a transmis un cadru, emiţătorul este obligat de protocol să intre în aşteptare un timp, până când soseşte micul cadru fictiv (deci confirmarea). Cum am presupus ca avem un canal fara zgomote, nici cadrul si nici confirmarea nu se pot pierde.

Chiar dacă traficul de date este simplex, mergând numai de la emiţător la receptor, cadrele se deplasează în ambele direcţii. În consecinţă, canalul de comunicaţie dintre cele două niveluri legătură de date trebuie să permită transferul de informaţie bidirecţional. Totusi fluxul alterneaza: mai întâi emiţătorul trimite un cadru, apoi receptorul trimite un alt cadru, apoi emiţătorul trimite alt cadru ş.a.m.d.

La inceput, receptorul preia un pachet de la nivelul retea, construieste un cadru si il trimite spre nivelul fizic. Deoarece am presupus ca nivelul retea e mereu pregatit, mereu va exista un pachet de preluat; ulterior se va renunta la aceasta presupunere. Spre deosebire de protocolul 1, acum emitatorul va trebui sa astepte un eveniment, acesta fiind sosirea unui cadru de confirmare de la receptor. Nivelul legătură de date care transmite nu are nevoie să inspecteze cadrul care soseşte: nu există decât o singură posibilitate. Cadrul primit este întotdeauna o confirmare. Dup ace s-a produs evenimentul, emitatorul isi reia ciclul si trimite un nou cadru nivelului fizic.

Singura diferenta la receptor fata de potocolul 1 este ca dup ace primeste pachetul si il trimite spre nivelul retea, receptorul trimite un cadru de confirmare inapoi la emitator, spre nivelul fizic, înainte de a intra din nou în bucla de aşteptare. Deoarece numai sosirea cadrului de întoarcere la emiţător este importantă, nu şi

conţinutul lui, receptorul nu trebuie să pună nici o informaţie particulară în el.

Exemplu: A ii trimite cadru lui B, B ik primeste si trimite confirmare

Protocol cu confirmare si retransmitere

Acest protocol ia in calcul situaţia normală a unui canal de comunicaţie unde pot apărea erori. Cadrele pot fi modificate sau pierdute in tranzit. In cazul in care un cadru a fost modificat, echipamentul receptor va detecta acest lucru atunci când calculează suma de control.

O prima solutie la care ne puteam gandi ar fi adaugarea unui contor de timp. Emitatorul trimite un cadru, iar receptorul trimite un cadru de confirmare numai dacă informaţia a fost recepţionată corect. Dacă la receptor ajunge un cadru modificat, este eliminat. După un timp, emiţătorul va ieşi din aşteptare şi va retrimite cadrul. Acest proces va fi repetat până când cadrul va ajunge în final intact.

Insa aceasta metoda produce o eroare. Nivelul reţea de pe B nu are nici o posibilitate să ştie că un pachet a fost pierdut sau duplicat, aşa că nivelul legătură de date trebuie să garanteze că nici o combinaţie de erori de transmisie, indiferent cât de puţin probabile, nu poate produce un pachet duplicat care să fie transmis nivelului reţea.

Să considerăm următorul scenariu:

1. Nivelul reţea de pe A trimite pachetul 1 către nivelul său legătură de date. Pachetul este corect recepţionat de B şi este trimis nivelului reţea de pe B. B trimite un cadru de confirmare înapoi lui A.

2. Cadrul de confirmare s-a pierdut complet. El nu va mai ajunge deloc. Viaţa ar fi cu mult mai simplă în cazul în care canalul ar altera sau pierde doar cadre de date, nu şi cadre de control, dar, din nefericire, canalul nu face discriminări.

3. Nivelul de legătură de date de pe A aşteaptă expirarea timpului limită. Nerecepţionând o confirmare, el presupune (incorect) că acel cadru de date a fost modificat sau pierdut şi trimite încă o dată cadrul conţinând pachetul 1.

4. Cadrul duplicat ajunge şi el cu bine la nivelul legătură de date B şi este trimis nivelului reţea de acolo. Dacă A trimite un fişier lui B, o porţiune de fişier va fi duplicată (adică, copia fişierului făcută de B va fi incorectă şi eroarea nu va fi detectată). Cu alte cuvinte, protocolul va eşua.

Numar de secventa:

În mod clar, este necesară o soluţie ca receptorul să poată distinge un cadru pe care îl vede pentru

prima dată de o retransmisie. Soluţia evidentă este aceea ca emiţătorul să pună un număr de secvenţăîn antetul fiecărui cadru pe care îl trimite. Apoi receptorul poate verifica numărul de secvenţă al fiecărui cadru sosit pentru a vedea dacă este un cadru nou sau un duplicat ce trebuie eliminat. Este deci suficient un număr de secvenţă de 1 bit (0 sau 1). La fiecare moment de timp, receptorul aşteaptă un anumit număr de secvenţă. Orice cadru sosit, care conţine un număr de secvenţă greşit este rejectat ca duplicat. Atunci când soseşte un cadru cu număr de secvenţă corect, acesta este acceptat şi transmis nivelului reţea. Apoi numărul de secvenţă aşteptat este incrementat modulo 2.

Protocolul 3 se deosebeşte de predecesorii săi prin aceea că şi emiţătorul şi receptorul au o variabilă a cărei valoare este păstrată cât timp nivelul legătură de date este în starea de aşteptare. Emiţătorul păstrează numărul de secvenţă al următorului cadru de transmis în next\_frame\_to\_send; receptorul păstrează numărul de secvenţă al următorului cadru aşteptat în frame\_expected.

Explicatie cod:

După transmiterea unui cadru, emiţătorul declanşează contorul de timp. Dacă acesta era deja

pornit, atunci va fi resetat pentru un nou interval complet. Intervalul de timp trebuie să fie ales astfel, încât să permită sosirea cadrului la receptor, prelucrarea sa de către receptor, chiar şi în cazul cel mai defavorabil, şi propagarea cadrului de confirmare înapoi la emiţător. Numai atunci când a expirat acest interval de timp se poate spune cu siguranţă că s-a pierdut fie cadrul transmis, fie confirmarea sa şi se poate trimite un duplicat. Dacă intervalul de timp este prea mic, emiţătorul va transmite cadre care nu sunt necesare. Aceste cadre nu influenţează corectitudinea protocolului, dar îi afectează performanţele.

După transmiterea unui cadru şi pornirea contorului de timp, emiţătorul aşteaptă să se întâmple

ceva interesant. Există doar trei posibilităţi: un cadru de confirmare ajunge intact, soseşte un cadru

de confirmare eronat sau expiră timpul. Dacă soseşte o confirmare validă, atunci emiţătorul preia

următorul pachet de la nivelul reţea şi îl pune în tampon, scriind peste pachetul anterior. De asemenea avansează numărul de secvenţă. Dacă soseşte un cadru modificat sau nu soseşte nici un cadru, nu se modifică nici tamponul şi nici numărul de secvenţă, aşa că poate fi transmis un duplicat.

Atunci când la receptor soseşte un cadru corect, este verificat numărul de secvenţă, pentru a vedea dacă nu cumva este un duplicat. Dacă nu, este acceptat, transmis nivelului reţea şi este generată

o confirmare. Cadrele duplicate şi cele modificate nu sunt trimise către nivelul reţea.

Protocoalele cu fereastra glisanta

În protocoalele anterioare, cadrele cu date erau transmise într-o singură direcţie. În cele maimulte situaţii practice, este necesar să se transmită date în ambele direcţii.

Ideea folosita e de a utiliza acelasi circuit in ambele directii. Cadrele cu date de la A la B sunt amestecate cu cadrele de confirmare de la A la B. Uitându-se la câmpul kind din antetul cadrului ce a sosit, receptorul poate spune dacă este vorba de un cadru de date sau de confirmare.

O alta optimizare este legata de cadrul de confirmare. Atunci când soseşte un cadru cu date, în locul emiterii imediate a unui cadru de control separat, receptorul stă şi aşteaptă până când nivelul reţea îi dă următorul pachet. Confirmarea este ataşată cadrului cu date de ieşire (utilizând câmpul ack din antetul cadrului). De fapt, confirmarea este transportată pe gratis de către următorul cadru cu date de ieşire. Astfel se va utiliza mult mai bine lungimea de banda disponibila. Campul ack din antetul cadrului ocupă doar câţiva biţi, în timp ce un cadru separat va necesita un antet, confirmarea şi o sumă de control. În plus, mai puţine cadre transmise înseamnă mai puţine întreruperi datorate "sosirii cadrelor" şi probabil mai puţine zone tampon în receptor.

Insa o problema intalnita in aceasta situatie este cat va astepta nivelul legatura de date pachetul la care sa ataseze confirmmarea. Dacă nivelul legătură de date aşteaptă mai mult timp decât perioada de timeout a emiţătorului, cadrul va fi retransmis, anulând complet rolul confirmărilor.

Dacă un nou pachet soseşte repede, confirmarea este adăugată în el; altfel, dacă dup ace trece un numar fixat de ms nu a sosit un nou pachet, nivelul legătură de date trimite un cadru de confirmare separat.

EXEMPLU:

Până acum am făcut presupunerea tacită că timpul de transmisie necesar pentru ca un cadru să ajungă la receptor plus timpul de transmisie a confirmării este neglijabil. Dar de multe ori aceasta presupunere e falsa.

Daca consideram un canal de satelit de 50Kbps si timpul de intarziere datorita propagarii dus intors de 500 ms si vrem sa transmitem cadre de 1000 de biti prin satelit.

La t = 0 se transmite primul cadru. Acesta va face, in cel mai pozitiv caz in care nu sta la coada, pana la receptor 250 ms (intarzierea pe un sens) + 20 ms timpul de transport = 270 ms pentru a ajunge la receptor, iar confirmarea nu poate ajunge inapoi inainte de 520 ms. Deci emitatorul a fost blocat pentru 500/520 din timp, adica 96% din timp, adica a fost utilizata doar 4% din largimea de banda.

Problema descrisă anterior poate fi privită ca o consecinţă a regulii care cere ca un emiţător să aştepte o confirmare înaintea trimiterii unui alt cadru. O eficienta mult mai ridicata obtinem daca permitem emiţătorului să transmită până la m cadre, în loc de unul singur. Cu o alegere potrivită a lui m emiţătorul va putea să transmită continuu cadre pentru un timp egal cu timpul de tranzit, fără a umple fereastra.

In exemplul daca consideram dimensiunea maxima a ferestrei 26 vom avea urmatorul caz: la t = 0 este trimis primul cadru, iar la t = 520 vor fi trimise 26 de cadre si va ajunge si confirmarea primului cadru. Apoi, confirmările vor sosi la fiecare 20 milisecunde, aşa încât emiţătorul primeşte întotdeauna permisiunea să continue exact atunci când doreşte. În permanenţă există 25 sau 26 cadre neconfirmate.

Atunci cand va primi confirmarea pentru primul cadru, celelalte cadre din fereastra vor fi transmise.

Ideea protocoalelor cu fereastră glisantă este ca emitatorul are, la orice moment de timp, o mulţime de numere de secvenţă care corespund cadrelor pe care are permisiunea să le trimita. Ele apartin ferestrei de transmisie. De asemenea receptorul menţine o fereastră de recepţie (eng.: receiving window), ce corespunde mulţimii de cadre care pot fi acceptate. Fereastra emiţătorului şi fereastra receptorului nu trebuie să aibă aceleaşi limite minime şi maxime şi nici măcar aceeaşi dimensiune.

In continuare protocolul trebuie să livreze pachetele la nivelul reţea destinaţie în aceeaşi ordine în care acestea sunt trimise către nivelul legătură de date de pe maşina emiţătoare.

Numerele de secvenţă din cadrul ferestrei emiţătorului reprezintă cadre transmise sau cadre ce pot fi transmise, dar încă neconfirmate. De fiecare dată când de la nivelul reţea soseşte un nou pachet, acestuia îi este atribuit următorul număr de secvenţă, iar marginea superioară a ferestrei este avansată cu unu. Atunci când soseşte o confirmare, creşte cu unu limita inferioară a ferestrei. Deoarece cadrele din fereastra curentă a emiţătorului pot fi pierdute sau modificate pe traseu, emiţătorul trebuie să păstreze toate aceste cadre în memoria sa pentru o posibilă retransmisie. Astfel, dacă dimensiunea maximă a ferestrei este n, emiţătorul are nevoie de n tampoane pentru a păstra cadrele neconfirmate. Dacă fereastra creşte la dimensiunea maximă, nivelul legătură de date al emiţătorului trebuie să forţeze închiderea nivelului reţea până când se eliberează un tampon. De aceea putem da enable sau disable la network layer

Problema principala e ce se intampla cand un cadru din mijlocul unui şir lung este modificat sau pierdut. Multe cadre succesive vor ajunge la receptor înainte ca emiţătorul să observe că cevaeste greşit.

1. Varianta 1, go back n:

Inseamna ca receptorul să elimine pur şi simplu cadrele care urmează, netrimiţând confirmări pentru cadrele eliminate. Cu alte cuvinte, nivelul legătură de date refuză să accepte orice cadru exceptându-l pe următorul care trebuie livrat către nivelul reţea. Dacă fereastra emiţătorului se umple înaintea expirării contorului de timp, banda de asamblare va începe să se golească. În cele din urmă, timpul emiţătorului va expira şi se vor retransmite toate cadrele neconfirmate, în ordine, începând cu cadrul pierdut sau modificat. Dacă rata erorilor este mare, această abordare poate risipi o mare parte din lărgimea de bandă.

Fig pg 195

Această strategie corespunde unei ferestre de recepţie de dimensiune 1. Cu alte cuvinte, nivelul legătură de date refuză să accepte orice cadru exceptându-l pe următorul care trebuie livrat către nivelul reţea. În fig. 3-16 (a) este prezentat protocolul de revenire cu n paşi pentru cazul în care fereastra receptorului are dimensiune unu. Cadrele 0 şi 1 sunt primite şi confirmate corect. Cadrul 2, totuşi, este alterat sau pierdut. Emiţătorul, care nu ştie de această problemă, continuă să trimită cadre până când timpul pentru cadrul 2 expiră. Apoi se întoarce la cadrul 2 şi o ia de la început cu el, trimiţând din nou cadrele 2, 3, 4 etc

1. Varianta 2: reppetare selective: ă, un cadru incorect este respins, dar toate cadrele corecte care îl urmează sunt memorate. Când contorul de timp al emiţătorului expiră, cel mai vechi cadru neconfirmat este retransmis. Dacă acest cadru ajunge corect, receptorul poate transmite către nivelul reţea, în ordine, cadrele pe care le-a memorat.

Toate pachetele la stânga ferestrei sunt sigur deja recepționate. Toate pachetele la dreapta ferestrei nu au fost încă transmise nici măcar o dată, deci nu pot fi confirmate.

Protocoale cu revenire cu N pași = Go back N

Un mod de a implementa o fereastră glisantă este prin revenire cu N pași. Emițătorul transmite, în ordine, până la N pachete. Pe măsură ce primește ACK-uri, avansează fereastra. Dacă nu primește în timp util ACK pentru primul pachet din fereastră, le retransmite pe toate (dacă primește un ACK de la un pachet de după primul, consideră că toate până la acela au fost recepționate).

Nivelul legatura de date:

Pentru a îndeplini aceste scopuri, nivelul legătură de date primeşte pachete de la nivelul reţea, pe care le încapsulează în cadre în vederea transmiterii. Fiecare cadru conţine un antet, un câmp de informaţie utilă pentru pachet şi încheiere, după cum se vede în fig. 3-1. Gestionarea cadrelor reprezintă esenţa a ceea ce face nivelul legătură de date.

Atentie: deseneaza figura carte pg 166

Principalul serviciu este transferul datelor de la nivelul reţea al maşinii sursă la nivelul reţea al maşinii destinaţie.

Acesta se ocupa cu transferal datelor la nivelul a 2 masini adiacente pe cand la niv retea nu.

La legatura de date unitatea de baza e cadrul, iar la niv retea pacheetul

-ce tin fiecare in antet

Nivelul legătură de date poate fi proiectat să ofere diferite servicii.

Serviciile efective oferite pot varia de la sistem la sistem. Trei posibilităţi de bază, oferite în mod curent, sunt:

1. Serviciu neconfirmat fără conexiune.

2. Serviciu confirmat fără conexiune.

3. Serviciu confirmat orientat-conexiune.

Serviciul neconfirmat fără conexiune constă din aceea că maşina sursă trimite cadre independente către maşina destinaţie, fără ca maşina destinaţie să trebuiască să confirme primirea lor. În acest caz, nu sunt necesare stabilirea şi desfiinţarea unei conexiuni logice. Dacă un cadru este pierdut datorită zgomotului de pe linie, la nivelul legătură de date nu se face nici o încercare pentru recuperarea lui. Utilizat pentru traficul de timp real, ca cel de voce.

Serviciu confirmat fara conexiune: Atunci când este oferit acest serviciu, încă nu se utilizează conexiuni, dar fiecare cadru trimis este confirmat individual. În acest mod, emiţătorul ştie dacă un cadru a ajuns sau nu cu bine. Dacă nu a ajuns într-un interval de timp specificat, poate fi trimis din nou. Acest serviciu este folositor pentru canale nesigure, cum ar fi sistemele fără fir.

Atunci când este utilizat serviciul orientat conexiune, transferurile au trei faze distincte. În prima fază este stabilită conexiunea, ambele părţi iniţializând variabile şi contoare, utilizate pentru a ţine evidenţa cadrelor care au fost recepţionate şi a celor care nu au fost. În a doua fază, sunt transmise unul sau mai multe cadre. În a treia şi ultima fază, conexiunea este desfiinţată, eliberând variabilele, tampoanele şi alte resurse utilizate la menţinerea conexiunii.

Functia principala e sa transmita datele fara erori catre nivelul retea.

PROTOCOLUL IP:

Identificare: Trecem intr –o retea si pachetul se dovedeste a afi mai mare decat capacitatea noii retele de a transmite informatii; bucatele mai mici ale aceluiasi pachet – fragmentele apartin aceluiasi pachet – DIRIJARE IERARHICA

DF: putem forta transmitera pachetului ai sa nu treaca prin retele care sa sparga pachetul in bucati mai mici. Daca e spart in fragmente sanseke cresc. Daca ajung intr-o poarta sau ajunge intr-un circuit virtual care ar trebui sa il sparga, cauta alta cale.

MF: dirijare ierarhica

Deplasament: essential la datagrama

OPTIUNI:

niciun alg de dirijare despre care am vb nu luau in calcul aceste optiuni. Cand optiunile sunt setate, alg de dirijare trb sa tina cont de ele

De securitate: indifferent de prioritate noi nu trebuie s ail interceptam, s ail deschidem. Dar cand e setata aceasta optiune, cei care sunt interesati sa intercepteze, devin mai curiosi cand vad optiunea de securitate.

Impunerea unor drumuri:

* Desi noi folosim o datagrama, in anumite situatii aplicam un algoritm in anumite situatii pentru a afla drumul cel mai scurt, noi tot ii spunem calea pe unde trebuuie sa mearga din motive de securitate(in cazul in care ar putea sa treaca printr o zona prin care nu as dori)
* Indic anumite routere prin care vreau sa treaca, care nu trebuie sarite

Evitarea blocarii:

Un algoritm de dirijare bun poate ajuta la evitarea congestiei prin răspândirea traficului de-a lungul tuturor liniilor, în timp ce un algoritm neperformant ar putea trimite toate pachetele pe aceeaşi linie, care deja este congestionată.

Daca se allege drumul cel mai scurt, se va congestiona o ruta.

Imi dau seama daca un pachet nu ajunge sau intarzie prea mult.

1. Transmisia cu cv

* Toate pachetele (tot mesajjl) vor avea acelasi drum, vor alege acelasi circuit virtual, poate conduce la congestionare
* In momentul in care eu deschid un cv, pentru fiecare nod intermediar voi rezerva zone tampon; cozile de asteptare vor avea lungimea de cat poate sa fie prelucrat fara a ajunge la blocare; cand zonele nu sunt disponibile, pachetul care gaseste drumul il respinge si alege alt circuit. Nu te lasa sa ajungi in situatia de a fi blocat, pentru ca pachetul martir va decide drumul

1. SI cv si datagrama

* Ce fac daca am trimis pachete, primele pachete trimise sunt in zona in care putem sa acceptam, mai trimitem inca unul si zona tampon s-a umplut; ajungem intr un pct care este plin; pachetul pe care l am trimis este ignorat si este refuzat sa fie trimis la destinatie(nu e procesat) . cum nu primeste confirmare, e retrimis

Blocare definitiva:

M- diametrul + 1 zone de tampon

-blocaj intre A si B; a trans lui b si b lui A

LAN:

Pg 15

Modelul osi pg 34