**Tema de casa**

**Retele de calculatoare**

**Mazilu Razvan si Popescu Mihai**

**Problemele designului nivelului legatura de date**

**(Data Link Layer Design Issues)**

Cuprins :

 Flow Control – Mazilu Razvan

            Flow Control protocols

 Stop and wait protocol

 Sliding window protocol

 Go back n protocol

            Hardware flow control

           Software flow control

 Reprezentare

 Mecanism

Error Control- Popescu Mihai

 Error types

Error recovery up to higher layers

Automatic Repeat Request

**Flow control(controlul fluxului)**

Un aspect important de proiectare care apare la nivelul legătură de date este cum trebuie procedat cu un emiţător care transmite date în mod systematic cadre mai repede decât poate să accepte receptorul. Această situaţie poate să apară uşor atunci când emiţătorul rulează pe un calculator rapid (sau mai puţin încărcat) şi receptorul rulează pe o maşină lentă (sau foarte încărcată). Emiţătorul continuă să transmită cadre la o rată înaltă până când receptorul este complet inundat. Chiar dacă transmisia este fără erori, la un anumit punct receptorul nu va mai fi capabil să trateze cadrele care sosesc şi va începe să piardă unele dintre ele. Trebuie făcut ceva pentru a evita această situaţie. Există două abordări des utilizate. In cazul celei dintâi, **controlul fluxului bazat pe reacţie (feedback-based flow control)**, receptorul acordă emiţătorului permisiunea de a mai transmite date, saucel puţin comunică emiţătorului informaţii despre starea sa. În cea de-a doua, **controlul fluxului bazat pe rată (rate-based flow control)**, protocolul dispune de un mecanism integrat care limitează rata la care emiţătorul poate transmite, fără a folosi informaţii de la receptor. La nivelul legatura de date, se utilizeaza controlul fluxului bazat pe reactive. Sunt cunoscute diferite scheme de control al fluxului, dar cele mai multe dintre ele utilizează acelaşi principiu de bază. Protocolul conţine reguli bine definite despre momentul când emiţătorul poatetrimite următorul cadru. Deseori aceste reguli interzic trimiterea cadrelor înainte ca receptorul săo permită, implicit sau explicit. De exemplu, când se stabileşte o conexiune, receptorul trebuie săspună: "Acum poţi să-mi trimiţi *n* cadre, dar după ce au fost trimise, să nu trimiţi altele până când nuîţi spun să continui". [1]

**Protocolul Stop and Wait(Stop and Wait protocol)**

Se presupune că nu se produc erori pe canalul de comunicaţie şi că traficul de date este simplex.

Principala problemă care trebuie evitata este cum să evităm ca emiţătorul să inunde receptorul cu date care sosesc mai rapid decât poate acesta să prelucreze. În esenţă, dacă receptorul are nevoie de un anumit timp 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 acel interval de timp. Mai mult, dacă se presupune 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.

În anumite situaţii speciale (de exemplu, transmisie sincronă şi un nivel legătură de date receptor complet dedicat prelucrării unei singure linii de intrare) ar putea fi posibil ca emiţătorul să introducă pur şi simplu o întârziere în protocolul 1, pentru a-l încetini suficient, astfel încât să se evite inundarea receptorului. Totuşi, de obicei, fiecare nivel legătură de date va avea mai multe linii de luat în considerare şi intervalul de timp între sosirea unui cadru şi începutul prelucrării sale 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. Problema cu această abordare este aceea că este prea conservatoare. Ea conduce la o utilizare a lărgimii de bandă care este cu mult sub optim, cu excepţia situaţiei în care cazurile cel mai favorabil şi cel mai defavorabil sunt aproape la fel (adică, variaţia timpului de reacţie al nivelului legătură de date este mică).

O soluţie mult mai generală a acestei dileme este ca receptorul să furnizeze o reacţie către emiţător. După trimiterea unui pachet către nivelul său reţea, receptorul trimite un mic cadru către emiţător care, de fapt, îi dă emiţătorului permisiunea să transmită următorul cadru. După ce a transmis un cadru, emiţătorul este obligat de protocol să intre în aşteptare un timp, până când soseşte acel cadru de confirmare. Utilizarea reacţiei de la receptor pentru a anunţa emiţătorul că poate trimite date este un exemplu de control al fluxului menţionat anterior.

Protocoalele în care emiţătorul trimite un cadru şi apoi, înainte de a continua, aşteaptă o confirmare, se numesc **stop-and-wait** (pas-cu-pas). 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. Totuşi, acest protocol impune o alternanţă strictă a fluxului: mai întâi emiţătorul trimite un cadru, apoi receptorul trimite un alt cadru, apoi emiţătorul trimite alt cadru ş.a.m.d. În acest caz este suficient un canal fizic semiduplex.

Emiţătorul începe prin preluarea unui pachet de la nivelul reţea, utilizarea lui pentru construirea unui cadru şi trimiterea acestuia. Emiţătorul trebuie să aştepte până când soseşte un cadru de confirmare, înainte de a relua ciclul şi a prelua următorul pachet de la nivelul reţea. 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. 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.[1]



*http://www.javiervalcarce.eu/upload/0/0c/StopAndWait.png*

Mai sus este prezentata o diagrama in timp a implementarii unui protocol stop and wait. Acesta prezinta o imbunatatire datorita faptului ca asteapta confirmarea un anumit moment de timp T0. In cazul in care timpul a expirat si confirmarea nu a sosit, emitatorul va retrimite ultimul pachet.

**Protocolul cu fereastra glisanta(Sliding window protocol)**

**Go back n protocol**

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. Uneori această presupunere este în mod cert falsă. În aceste situaţii timpul mare de transfer poate avea implicaţii importante pentru eficienţa utilizării lărgimii de bandă. Evident, combinaţia dintre un timp de tranziţie lung, lărgime de bandă mare şi un cadru de lungime mică este dezastruoasă din punct de vedere al eficienţei.

Practic, soluţia constă în a permite emiţătorului să transmită până la *w* cadre, în loc de unul singur. Cu o alegere potrivită a lui *w* emiţătorul va putea să transmită continuu cadre pentru un timp egal cu timpul de tranzit, fără a umple fereastra. În exemplul anterior *w* va fi minim 26. Emiţătorul începe emiterea cadrului 0 ca mai înainte. În momentul în care se termină trimiterea a 26 de cadre, la *t* = 520, va sosi şi confirmarea pentru cadrul 0. 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. Cu alte cuvinte dimensiunea maximă a ferestrei emiţătorului este de 26.

Nevoia pentru o fereastră mare la emiţător apare atunci când produsul lărgime de bandă x timpul de propagare dus-întors este mare. Dacă lărgimea de bandă este mare, chiar şi pentru întârzieri moderate, emiţătorul îşi va termina repede fereastra. Daca întârzierea este mare (de exemplu, canal de satelit), emiţătorul îşi va termina fereastra chiar şi pentru lărgimi de bandă moderate. Produsul acestor doi factori spune de fapt care este capacitatea canalului, iar pentru a opera la eficienţă maximă, emiţătorul trebuie sa fie capabil sa o umple fără să se oprească.

Această tehnică este cunoscută ca **bandă de asamblare** (eng.: **pipelining**). Considerând capacitatea canalului de *b* biţi pe secundă, dimensiunea cadrului de *l* biţi şi timpul de propagare dus-întors *R* secunde, timpul necesar pentru a transmite un singur cadru este *l/b* secunde. După ce a fost transmis ultimul bit al unui cadru de date, apare o întârziere de *R/2* înainte ca biţii să ajungă la receptor şi o altă întârziere de cel puţin *R/2* pentru sosirea confirmării, rezultând o întârziere totală de *R*. În cazul protocoalelor pas-cu-pas, linia este ocupată pentru un timp egal cu *l/b* şi în aşteptare pentru un timp egal cu *R*, rezultând: utilizarea liniei = *l/(l+bR*).

Dacă *l* < *bR,* eficienţa va fi mai mică de 50%. Deoarece până la întoarcerea confirmării există întotdeauna o întârziere nenulă, în principiu poate fi folosită banda de asamblare, pentru a ţine linia ocupată tot acest interval, dar dacă intervalul este mic, complexitatea suplimentară face efortul inutil. Utilizarea benzii de asamblare în cazul unui canal de comunicaţie nesigur ridică probleme serioase. Mai întâi să vedem ce se întâmplă dacă un cadru din mijlocul unui şir lung este modificat sau pierdut. Multe cadre succesive vor ajunge la receptor înainte ca emiţătorul să observe că ceva este greşit. Atunci când un cadru modificat ajunge la receptor este evident că el trebuie eliminat, dar ce trebuie să facă receptorul cu toate cadrele corecte care urmează? Să reamintim că nivelul legătură de date receptor este obligat să livreze pachete către nivelul reţea în secvenţă. În figura de mai sus, se prezintă efectele utilizării benzii de asamblare asupra revenirii în caz de eroare.



Folosirea benzii de asamblare şi revenirea din eroare. Efectul unei erori când (a) dimensiunea

ferestrei receptoare este 1 şi (b) dimensiunea ferestrei receptorului este mare.

[*http://computing.dcu.ie/~humphrys/Notes/Networks/data.sliding.html*](http://computing.dcu.ie/~humphrys/Notes/Networks/data.sliding.html)

Există două moduri de bază de tratare a erorilor în prezenţa benzii de asamblare. Un mod, numit **revenire cu n paşi** (eng.: **go back n**), este ca receptorul să elimine pur şi simplu cadrele care urmează, netrimiţând confirmări pentru cadrele eliminate. 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. 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.

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.

Cealaltă strategie generală de tratare a erorilor atunci când este folosită banda de asamblare se numeşte **repetare selectivă** (eng.: **selective repeat**). Când aceasta este utilizată, 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. Repetarea selective este deseori combinată cu utilizarea confirmărilor negative (NAK), care sunt trimise atunci când se detectează o eroare, de exemplu când se primeşte un cadru cu suma de control incorectă sau cu număr de secvenţă necorespunzător. Confirmările negative simulează retransmisia înainte de expirarea contorului de timp corespunzător, îmbunătăţind astfel performanţa.

Strategia de repetare selectivă corespunde unei ferestre a receptorului mai mare ca 1. Orice cadru din interiorul ferestrei poate fi acceptat şi memorat până când toate cele precedente vor fi trimise nivelului reţea. Dacă fereastra este mare, această abordare poate necesita un spaţiu mare de memorie pentru nivelul legătură de date.

Aceste două alternative reprezintă compromisuri între lărgimea de bandă şi spaţiul ocupat de tampoane la nivelul legătură de date. În funcţie de care resursă este mai deficitară, poate fi utilizată una sau cealaltă. Fig. 3-17 prezintă un protocol de tip bandă de asamblare în care nivelul legătură de date receptor acceptă cadre ordonate; cadrele ce urmează după o eroare sunt eliminate. În acest protocol, pentru prima dată, am renunţat la presupunerea că nivelul reţea are o rezervă infinită de pachete care trebuie trimise. Atunci când nivelul reţea are un pachet pe care doreşte să-l trimită, poate produce un eveniment *network\_layer\_ready*. Totuşi, pentru a impune regula de control al fluxului, conform căreia nu pot exista decât cel mult *MAX\_SEQ* cadre neconfirmate, nivelul legătură de date trebuie să poată să interzică nivelului reţea să îl perturbe cu mai multe. Această funcţie este realizată de funcţiile de bibliotecă *enable\_network\_layer* şi *disable\_network\_layer*.

Observaţi că în orice moment pot exista cel mult *MAX\_SEQ* cadre şi nu *MAX\_SEQ*+1 cadre neconfirmate,

chiar dacă există *MAX\_SEQ*+1 numere de secvenţă: 0, 1, 2, ...*MAX\_SEQ*. Pentru a vedea

de ce este necesară această restricţie, să considerăm următorul scenariu cu *MAX\_SEQ* = 7.

1. Emiţătorul trimite cadrele de la 0 la 7.

2. O confirmare ataşată pentru cadrul 7 ajunge la emiţător.

3. Emiţătorul trimite alte opt cadre, din nou cu numerele de secvenţă de la 0 la 7.

4. Acum ajunge o altă confirmare ataşată pentru cadrul

**Hardware flow control**

In standardul RS 232 sunt perechi de linii de control care se refera usual la controlul fluxului de date hardware.

RTS(request to send) si CTS(clear to send) sunt folosite in mecanismul RTS de control al fluxului

DTR(data terminal ready) si DSR(data set ready sunt folosite in mecanismul DTR

 Mecanismul hardware de control al fluxului este in general manipulate de DTE.[6]

**Software flow control**

Controlul fluxului software este o metoda de control a fluxului utilizata la nivelul legatura de date, in special pe comunicatia serial RS-232. Foloseste coduri special, transmise in banda, peste canalul primar de comunicatii. Aceste coduri sunt denumite in special **“XOFF/XON”.** De aceea controlul fluxului software este denumit cateodata si “XON/XOFF flow control”. Acesta este in contrast cu controlul fluxului cu semnale in afara benzii, denumit “hardware flow control”.[4]

**Reprezentare**

Pentru sistemele care folosesc codul ASCII, XOFF este in general reprezentat de caracterul cu valoarea 19, iar XON cu valoarea 17.

Standardul ASCII nu isi rezerva caractere de control specifice pentru a folosi XON/XOFF. Totusi, ASCII asigura patru caractere generice de control ale dispozitivelor, caracterele DC1 pana la DC4. Modelul Teletype 33 ASR adopta doua din aceste caractere DC3 si DC1 pentru a le utiliza ca XOFF si XON. Aceasta folosire a fost imprumutata si de altii, devenind intr-o oarecare masura un standard. Echivalentele acestor caractere(XON/XOFF) la tastatura sunt CTRL+S pentru XOFF si CTRL+Q pentru XON.[4][5]

**Mecanism**

Cand un terminal data link nu este capabil sa mai accepte date sau se apropie de acest punct, el trimit un semnal XOFF catre celalat terminal. Cand celalalt terminal a primit semnalul XOFF, acesta suspenda transmisia. Odata ce primul terminal este gata sa accepte date din nou, trimite un semnal XON si atunci se reia transmisia.

De exemplu, sa ne imaginam un computer conectat la o imprimanta cu o viteza mai mica. Computerul incepe sa transmita date imprimantei. Datorita diferentei de viteza(el fiind mai rapid ca imprimanta), imprimanta incepe sa fie coplesita de date. Raspunsul imprimantei e sa trimita un semnal XOFF. Sistemul de operare ce ruleaza pe computer vede acest semnal si temporar se opreste din a transmite date. Odata ce imprimanta a prelucrat datele, va transmite un semnal XON, computerul vede acest semnal si incepe iar sa transmita date.[4]

In concluzie diferenta semnificativa dintre cele 2 mecanisme de control al fluxului(hardware si software) este aceea ca, cel hardware foloseste linii special de comanda a fluxului de date, pe cand cel software trimite aceste mesaje de control al fluxului pe aceleasi linii pe care trimite si datele.

Bibliografie

*Reţele de calculatoare, Editia a patra Andrew S. Tanenbaum[1]*

[*http://www.javiervalcarce.eu*](http://www.javiervalcarce.eu) *[2]*

[*http://www.cs.virginia.edu/~zaher/classes/CS457/lectures/flow-control.pdf*](http://www.cs.virginia.edu/~zaher/classes/CS457/lectures/flow-control.pdf) *[3]*

[*http://en.wikipedia.org/wiki/Software\_flow\_control*](http://en.wikipedia.org/wiki/Software_flow_control) *[4]*

[*http://computing.dcu.ie/~humphrys/Notes/Networks/data.sliding.html*](http://computing.dcu.ie/~humphrys/Notes/Networks/data.sliding.html) *[5]*

[*http://en.wikipedia.org/wiki/Flow\_control\_(data)#Hardware\_flow\_control*](http://en.wikipedia.org/wiki/Flow_control_%28data%29#Hardware_flow_control) *[6]*