PROTOCOALE DE COMUNICAȚIE : LABORATOR 4

Implementarea unui protocol cu fereastra glisanta

Responsabil: Alecsandru PĂTRAȘCU

Cuprins

Cerinte laborator

In cadrul laboratorului curent, legatura de date nu pierde si nu corupe informatia transmisa. Se cere sa se implementeze un protocol prin care legatura de date sa fie utilizata in mod eficient.

Bandwidth Delay Product (BDP)

Legaturile de date asigura transmisia datelor intre doua masini. Proprietatile principale care definesc o legatura de date sunt:

- **viteza de transmisie** (V) - *bandwidth*
  - reprezinta cantitatea de informatie care poate fi transmisa intr-o unitate de timp pe legatura de date
  - termeni sinonimi: rata de transmisie, capacitatea legaturii, latimea de banda
  - unitate de masura: *Mb/s* - megabiti/secunda
- **timpul de propagare** (T_P) - *delay*
  - reprezinta timpul in care un bit se propaga de la sursa la destinatie de-a lungul legaturii de date
  - unitate de masura: *s* - secunda

Legatura de date poate fi asemanata cu un cilindru in care datele sunt introduse de catre transmitator si primite de catre receptor. Aria sectiunii cilindrului reprezinta viteza de transmisie, iar inaltimea este timpul de propagare.

Volumul cilindrului determina cantitatea de informatie aflata pe legatura de date, la un anumit moment de timp. Deci, cantitatea de informatie aflata *in zbor* la un anumit moment de timp este: $V \times T_P$

$BDP = V \times T_P$

Latenta (L) reprezinta timpul necesar pentru a transmite un mesaj pe o legatura de date. Pentru a defini latenta, avem nevoie de:

- timp de serializare - T_S
  - este timpul necesar pentru a pune un mesaj (M biti) pe legatura de date
  - depinde de capacitatea conexiunii; conexiune rapida =¿ timp de serializare scurt
  - odata serializat, cadrul va fi propagat catre destinatie
- timp de propagare - T_P
- viteza de transmisie = V

Asadar: $L = T_S + T_P, T_S = \frac{M}{V}$

Ferestre

Pentru a descrie timpul petrecut de un cadru in retea, se foloseste si notiunea de RTT (Round Trip Time), ce reprezinta timpul scurs din momentul in care un cadru este trimis pana in momentul in care este primita confirmarea.

Observatie: simulatorul din scheletul de cod considera timpul de propagare pentru confirmare egal cu 0, deci $RTT = T_P$.

Dezavantaje *STOP AND WAIT*:

- un singur cadru in legatura de date pe parcursul unui RTT
- o marire a vitezei de transmisie V nu aduce benecii notabile

Fereastra W (window) = numarul maxim de cadre neconfirmate la orice moment de timp. Pentru protocolul *STOP AND WAIT*, fereastra este de 1 cadru. Asadar, o utilizare din plin a legaturii presupune o fereastra egala cu BDP =¿ $W = BDP$.

Ferestre glisante (Sliding Windows)

Dupa ce am pus pe fir un numar de cadre egal cu dimensiunea ferestrei, asteptam confirmarile. Fiecare confirmare ne va permite sa introducem un nou cadru in retea.

Un exemplu vizual pentru $W = 3$:

- dupa trimiterea celor W cadre:

$Cadru_1$	$Cadru_2$	$Cadru_3$	$Cadru_4$	$Cadru_5$	$Cadru_6$	$Cadru_7$

- dupa primirea primului ACK (confirmare pentru $Cadru_1$), putem trimite urmatorul cadrul ($Cadru_4$)

$Cadru_1$	$Cadru_2$	$Cadru_3$	$Cadru_4$	$Cadru_5$	$Cadru_6$	$Cadru_7$

Cadrele in chenar sunt trimise si neconfirmate. La fiecare ACK pentru cel mai vechi cadru trimis, fereastra gliseaza spre dreapta.

Pasi rezolvare laborator

Implementarea va porni de la scheletul de cod atasat laboratorului.

1. Modificati, din scriptul *run_experiment*, parametrii *SPEED* si/sau *DELAY*. Observati schimbarile in timpul de rulare.

2. Implementati un protocol care sa utilizeze eficient legatura de date.

3. BONUS: ce se intampla atunci cand se trimite o cantitate de date care depaseste valoarea BDP? Modicati programul pentru a evidentia acest caz. Discutati cu asistentul problemele care pot sa apara.

Nota: nu trebuie sa fie modicata structura *msg* definita in lib.h - se vor utiliza doar campurile *len* si *payload*.

Software disponibil

1. Simulator legatura de date - executabilul *link* - generat in urma comenzii *make*
2. Schelet de cod pentru transmitator si receptor
3. API simulator:
   - *int send message(msg* m)*
     - parametru: mesajul care va trimis
     - rezultat: numarul de octeti transferati(in caz de succes) sau -1 in caz de eroare
   - *int recv message(msg* m)*
     - parametru: adresa la care se memoreaza datele primite
     - numarul de octeti receptionati(in caz de succes) sau -1 in caz de eroare
4. Compilare schelet de cod: *make*