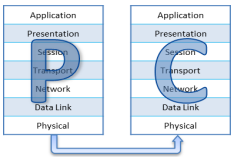
Universitatea Politehnica București
Facultatea de Automatică și Calculatoare



Protocoale de comunicație
Laboratorul 4

Responsabili laborator: Costin Raiciu(*costin.raiciu@cs.pub.ro*), Liviu Ioan(*liviu.ioan@cti.pub.ro*)

În cadrul laboratorului curent, legătura de date nu pierde și nu corupe informațiile.
Se cere să se implementeze un protocol prin care legătura de date să fie utilizată în mod eficient.

Bandwidth - delay product(BDP)
Legăturile de date asigură transmisia datelor între două mașini.
Proprietățile principale care definesc o legătură de date sunt:

- **viteza de transmisie** $V(bandwidth)$
  - reprezintă cantitatea de informație care poate fi transmisă într-o unitate de timp pe legătura de date
  - termeni sinonimi: rata de transmisie, capacitatea legăturii, lățimea de bandă
  - unitate de măsură: Mb/s - megabiți/secundă
- **timpul de propagare** $T_P(delay)$
  - reprezintă timpul în care un bit se propagă de la sursă la destinație de-a lungul legăturii de date
  - unitate de măsură: s - secunda

Legătura de date poate fi asemănată cu un cilindru în care datele sunt introduse de către transmițător și primite de către receptor. Aria secțiunii cilindrului reprezintă viteza de transmisie, iar înălțimea este timpul de propagare.
Volumul cilindrului determină cantitatea de informație aflată pe legătura de date, la un anumit moment de timp.
Deci, cantitatea de informație aflată *în zbor* la un anumit moment de timp este $V \times T_P$.

$$BDP = V \times T_P$$

Observație: BDP impune o limită asupra cantității de informație aflată *în zbor*.

Latența(L) reprezintă timpul necesar pentru a transmite un mesaj pe o legătură de date.
Pentru a defini latența, avem nevoie de:

- timp de serializare = T_S
  - este timpul necesar pentru a pune un mesaj(M biți) pe legătura de date
  - depinde de capacitatea conexiunii; conexiune rapidă → timp de serializare scurt
  - odată serializat, cadrul va fi propagat către destinație
- timp de propagare = T_P
- viteza de transmisie = V

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

Ferestre

Pentru a descrie timpul petrecut de un cadru în rețea, se folosește și noțiunea de *RTT*.
RTT(Round Trip Time) reprezintă timpul scurs din momentul în care un cadru din momentul în care este trimis până în momentul în care este primită confirmarea.
Observație: simulatorul din scheletul de cod consideră timpul de propagare pentru confirmare egal cu 0, deci $RTT = T_P$.
Dezavantaje *STOP AND WAIT*:

- un singur cadru în legătura de date pe parcursul unui RTT
- o mărire a vitezei de transmisie V nu aduce beneficii notabile

Fereastră W(*window*) = numărul maxim de cadre neconfirmate la orice moment de timp
Pentru protocolul *STOP AND WAIT*, fereastra este de 1 cadru.
Așadar, o utilizare din plin a legăturii presupune o fereastră egală cu *BDP*.

$$W = BDP$$

Ferestre glisante(Sliding Windows)

După ce am pus pe fir un număr de cadre egal cu dimensiunea ferestrei, așteptăm confirmările.
Fiecare confirmare ne va permite să introducem un nou cadru în rețea.
Un exemplu vizual($W = 3$):

- după trimiterea celor W cadre

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

- după primirea primului ACK(confirmare pentru $Cadru_1$), putem trimite următorul cadru($Cadru_4$)

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

Cadrele hașurate cu galben sunt trimise și neconfirmate.
La fiecare ACK pentru cel mai vechi cadru trimis, fereastra *glisează* spre dreapta.

Implementarea va porni de la scheletul de cod pentru laboratorul 4.

Pași rezolvare laborator

1. Modificați, din scriptul *run_experiment*, parametrii *SPEED* și/sau *DELAY*.
   Observați schimbările în timpul de rulare.
2. Implementați un protocol care să utilizeze eficient legătura de date.
3. *BONUS*: ce se întâmplă atunci când se trimite o cantitate de date care depășește valoarea *BDP*?
   Modificați programul pentru a evidenția acest caz. Discutați cu asistentul problemele care pot să apară.

Notă: nu trebuie să fie modificată structura *msg* definită în lib.h - se vor utiliza doar câmpurile *len* şi *payload*.

Software disponibil

- Simulator legătura de date - executabilul *link* - generat în urma *make*
- Schelet de cod
- *API* simulator:
  - *int send_message(msg* m)*
    - parametru: mesajul care va fi trimis
    - rezultat: numărul de octeţi transferaţi(în caz de succes) sau *-1* în caz de eroare
  - *int recv_message(msg* m)*
    - parametru: adresa la care se memorează datele primite
    - rezultat: numărul de octeţi recepţionaţi(în caz de succes) sau *-1* în caz de eroare