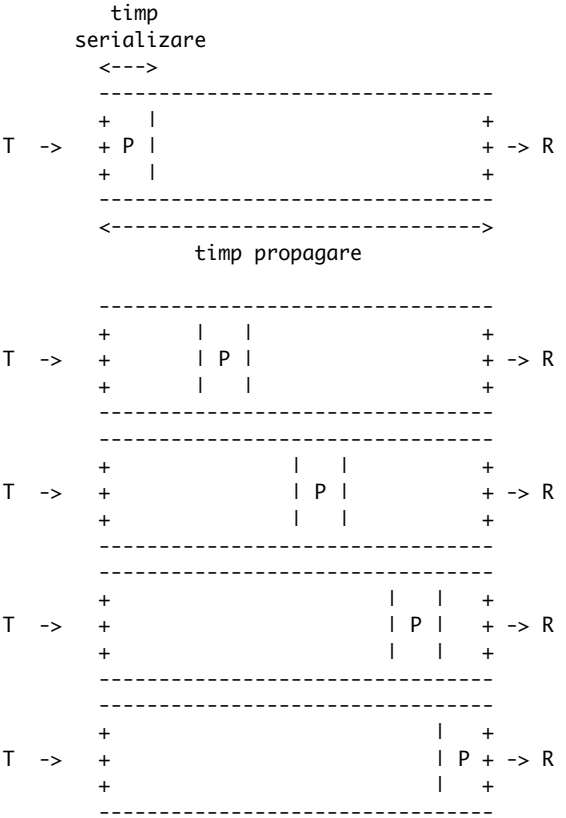
TEMA 1 PC

Legaturile de date asigura transmisia datelor intre doua calculatoare. Proprietatile principale care definesc o legature de date sunt:
- viteza de transmisie, masurata in megabiti/s (sau Mb/s), ne spune cat de multa informatie poate transmite intr-o secunda o legatura de date
- timpul de propagare a informatiei: ne spune cat dureaza pana un octet ajunge la destinatie, odata ce a plecat de la sursa.

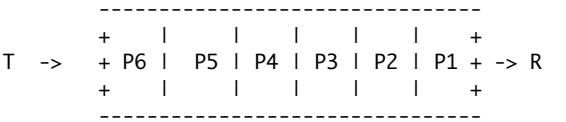
Este importanta distinctia intre viteza de transmisie si timpul de propagare (sau delay). De exemplu, legaturile 3G au viteze de transmisie in jurul a 2-3Mb/s si timpi de propagare destul de mari, de ordinul 100ms. Legaturile 802.11 (Wifi) au timpi de propagare mici (1-10ms) si viteze care variaza in functie de proprietatile canalului de comunicatie (1-54Mb/s).

Legatura de date poate fi asemanata cu un cilindru in care datele sunt introduse de catre transmitator (T) si primite de receptor (R). Pentru a transmite un cadru de date, legatura de date va serializa (transmite) cadrul cu ajutorul nivelului fizic. Timpul de serializare depinde de capacitatea conexiunii; cu cat e conexiunea mai rapida, cu atat timpul de serializare e mai scurt. Odata serializat, cadrul va fi propagat catre destinatie. Acesta va fi deserializat de catre receptor, dupa ce ultimul octet a fost primit.

Vizual, cadrul C este transmis astfel:



In exemplul de mai sus (protocolul start/stop), un singur cadru este transmis de legatura de date, apoi transmitatorul asteapta o confirmare de receptie, si trimite urmatorul cadru. Aceasta strategie simpla iroseste capacitatea conexiunii; pe scurt, conexiunea merge "in gol". Am dori sa putem transmite multe cadre concomitent, pentru a umple legatura de date, ca in exemplul de mai jos:



Pentru aceasta, este necesar sa trimitem mai multe cadre fara a astepta confirmare. In felul acesta legatura de date are de lucru in timp ce cadrele sunt livrate la destinatie (timpul de propagare). Cu cat e mai mare timpul de propagare, cu atat mai multe cadre trebuie sa transmitem fara a astepta confirmare. Pentru aceasta vom folosi un protocol cu fereastra glisanta (vedeti curs pentru mai multe detalii).

In aceasta tema vi se cere sa porniti de la o implementare a protocolului start-stop in C si sa implementati un protocol care utilizeaza eficient o legatura de date ce are urmatoarele caracteristici:
- o viteza de transmisie fixa
- un timp de propagare fix
- poate pierde a cadrele
- poate corupe cadrele, schimband unul sau mai multi octeti din cadruri aleatoare.

Imbunatatiti codul primit ca exemplu adaugand suport urmatoarele functionalitati:
a. Considerati ca fiind cunoscute viteza conexiunii (Mb/s) si timpul de propagare (acestea vor fi date ca parametri linia de comanda a send). Inlocuiti protocolul simplu start-stop din send.c cu un protocol cu fereastra glisanta care utilizeaza la maxim legatura de date.
(3p)

b. Implementati mecanisme pentru a retransmite cadrele lipsa.
(2p)

c. Implementati mecanisme pentru a detecta si retransmite cadrele transmise eronat (corupte)
(2p)

d. Implementati mecanisme pentru a permite unui receptor lent sa domoleasca viteza de transmisie a unui transmitator rapid.
(2p)

Se da un exemplu de cod care transmite si primeste mesaje folosind legatura de date (send.c, recv.c), si cod care simuleaza legatura de date (executabilul link). Se mai da ca exemplu un Makefile, si un script pentru a rula un experiment si a masura timpul de executie (run_experiment.sh).

Pentru rularea protocolului trebuie compilate send si recv, si executate link, recv si sender (in ordinea aceasta).Este important ca link sa nu ruleze deja atunci cand incercati il porniti; altfel, veti primi o eroare "Failed to bind". Rulati "killall link" pentru a opri link-ul.

Pentru a trimite si receptiona mesaje trebuie sa folositi metodele urmatoare definite in lib.h si implementate in lib.o:

*   send_message (msg* mesaj) - trimite mesaj catre destinatie; mesajul are dimensiune fixa de 1400B de continut util. Campul len permite transmitatorului sa specifice cati octeti sunt folositori. Campul type permite transmitatorului sa specifice tipul mesajului. In tema voastra puteti redefini semnificatia acestor campuri (de exemplu puteti crea noi tipuri pentru mesaj), o noua structura a payload, etc. Nu aveti voie insa sa editati lib.h; nu aveti voie sa modificati structura msg.

*   receive_message() - citeste un mesaj ; intoarce NULL daca a fost o eroare. Codul erorii se poate afisa cu perror(""). Apelul se va bloca pana cand un mesaj este primit.

*   receive_message_timeout(int ms) - la fel ca receive_message, insa asteapta maxim ms milisecunde un mesaj. Daca nu primeste in timpul specificat, va intoarce NULL.

Sender-ul ia ca parametru numele unui fisier F ce trebuie transmis, impreuna cu parametrii legaturii de date (speed=X, delay=Y, specificati la fel ca la link - vezi mai jos). Receiver-ul salveaza datele intr-un fisier numit recv_F. Receiver-ul primeste ca parametru fereastra folosita la primire (window=X); daca fereastra este 0 atunci receiver-ul nu va activa controlul fluxului (flow control, punctul d de mai sus).

Pentru a testa solutia, se va rula "link" cu parametri specificati din linia de comanda:
- speed=X seteaza viteza legaturii la X Mb/s. Tema va fi testata cu viteze de pana la 50Mb/s
- delay=Y seteaza timpul de propagare al legaturii de date, in ms. Valori de la 0 la 1000ms sunt permise.
- loss=Z specifica rata de pierdere de cadre; valori permise sunt de la 0 la 10%.
- corrupt=C specifica procentul de pachete care vor fi corupte (modificate); valori permise sunt de la 0 la 10%

Legatura de date dintre receiver si sender (folosita pentru confirmari) este ideala, avand timp de propagare 0, viteza infinita; nu pierde sau corupe pachete.

Legatura de date este bidirectionala, putand transporta pachete in ambele sensuri in acelasi timp (sender->receiver si receiver->sender). Nu exista nici un fel de interferente intre pachetele din sensuri diferite.

Solutia voastra trebuie sa functioneze pentru combinatii arbitrare de rate de pierdere de cadre, latente si viteze ale legaturii de date cu urmatoarele constrangeri:
- viteza legaturii de date este mai mica de 10Mb/s
- latenta este mai mica de 2s
- rata de pierdere de cadre este mai mica de 20%

Pentru predarea temei, fiecare student va incarca pe curs o arhiva tgz ce contine:
- codul sursa al temei (send.c si recv.c) SI
- un scurt document text (format txt, nu doc sau pdf) in care va explica cum a abordat fiecare dintre cerintele temei.

Pentru a crea o arhiva tgz rulati urmatoarea comanda din directorul in care ati implementat tema:
tar -czf nume_student_grupa.tgz send.c recv.c Makefile descriere.txt

Notarea se va face folosind:
- explicatiile din fisierul text
- compiland si ruland tema cu diferiti parametri ai legaturii de date pentru a transfera fisiere de diferite lungimi
a) fisierul sursa si destinatie for fi comparate automat folosind utilitarul diff
b) se va masura timpul de transmisie al fisierului

Pentru punctaj maxim, o tema trebuie sa treaca testul (a) si sa dea un timp aproape de minim (maxim cu 10% mai mare) la punctul (b). Nota se va obtine 25% din descrierea din fisierul text, si 75% din teste pentru fiecare din cerintele problemei.

Alte clarificari:
- descrierea din fisierul text trebuie sa corespunda cu implementarea; implementarea trebuie sa se compileze si sa functioneze conform descrierii. Descrieri ale solutiilor neimplementate in cod nu vor fi punctate.
- lib.o este compilat pe 32 biti; ca sa puteti rula tema trebuie sa compilati si voi pe 32 biti chiar daca aveti instalata o varianta Linux pe 64 biti. Pentru aceasta trebuie sa modificati Makefile pentru a adauga parametrul "-m32" la gcc. Daca primiti erori dupa acest pas, acestea se pot rezolva in ubuntu ruland: "sudo apt-get install gcc-multilib libc6-i386" iar in Fedora "yum install libc-devel"

Termen Predare Tema: 24/3/2012