PROTOCOALE DE COMUNICAȚII: TEMA 1

PROTOCOL CU FEREASTRĂ GLISANTĂ

Responsabili: Costin Raiciu (costin.raiciu@cs.pub.ro),
Vlad Ilie (ilievlad73@gmail.com), Mihalache Ionuț (ipopescu46@gmail.com)

Termen de predare: 6 Aprilie 2019, ora 23:55

Cuprins

1 Descriere generală

Se cere să se implementeze un protocol cu fereastră glisantă folosit pentru a transmite un fișier. Fișierul va fi citit de către transmițător(send.c) și trimis către receptor(recv.c). Protocolul implementat trebuie să folosească eficient legătura de date: legătura de date trebuie menținută plină. Implementarea va porni de la scheletul de cod pentru tema curentă. Transmițătorul primește următoarele argumente:

- numele fișierului care va fi transmis (fileX)
- viteza de transmisie (Mbps)
- timpul de propagare (ms)

Receptorul va salva datele primite în fișierul recv_fileX (numele fișierului trebuie transmis către receiver de sender).

2 Simulator legătură de date

- Viteza și timpul de propagare sunt constante (acești parametri sunt fixați și nu se modifică în timpul execuției programului).
- Viteza de transmisie are valori între 5Mbps și 20 Mbps.
- Timpul de propagare are valori între 10ms și 1000ms.
- Legătura de date poate pierde maxim 10% din cadrele trimise. Confirmările (trimise de către receptor) nu se pierd, nu se corup și sunt transmise instantaneu.
- Legătura de date poate să reordoneze pachetele.
- Legătura de date poate să corupă maxim 10% din cadrele trimise.
- Un cadru corupt conține un singur octet invalid în câmpul payload al structurii msg din lib.h.
- Parametrii pentru legătura de date se setează din scriptul run_experiment.sh. Valorile valide, așa cum sunt precizate mai sus, sunt:
  - SPEED = 5 ... 20
  - DELAY = 10 ... 1000
  - LOSS = 0 ... 10
  - CORRUPT = 0 ... 10
  - REORDER = 0 ... 10

3 API simulator

- int send_message(msg* m)
  - parametru m: mesajul care va fi trimis
  - rezultat: numărul de octeți transferați în caz de succes sau -1 în caz de eroare
- int recv_message(msg* m)
  - parametru m: adresa la care se memorează datele primite
  - rezultat: numărul de octeți recepționați în caz de succes sau -1 în caz de eroare
  - apelul este blocant: se așteaptă până la primirea unui mesaj
- int recv_message_timeout(msg *m, int timeout)
  - parametri:

    - timeout: timpul scurs până la ieșirea din apel (măsurat în ms)
    - m: adresa la care se memorează datele primite
  - rezultat: numărul de octeți recepționați în caz de succes sau -1 în caz de eroare sau timeout
  - apelul este blocant: se așteaptă până la primirea unui mesaj sau până la expirarea timeout-ului

4 Cerințe și notare

Să se implementeze un protocol cu fereastră glisantă care utilizează eficient legătura de date pentru a transmite un fișier, retransmite în mod eficient cadrele pierdute, detectează cadrele corupte și le retransmite și salvează cadrele care ajung "out-of-order" până când pot fi stocate pe disk.

Testarea soluției se va face prin rularea cu diferiți parametri pentru legătura de date pentru a verifica urmatoarele functionalități:

- **20p:** Transmiterea eficientă a fișierului atunci când legătura de date nu pierde și nu corupe cadre.
- **20p:** Transmiterea corectă și eficientă a fișierului atunci când legătura de date nu corupe datele, însă poate pierde cadre. Se pierd maxim 10% din cadre.
- **20p:** Transmiterea corectă și eficientă a fișierului atunci când legătura de date poate pierde și corupe cadrele. Se pierd maxim 10% din cadre. Se corup maxim 10% din cadre.
- **25p:** Transmiterea corectă și eficientă a fișierului atunci când legătura de date reordonează cadrele.

În afara testelor, se va acorda următorul punctaj:

- **10p:** README care va explica modul de abordare a cerințelor și structura codului.
- **10p:** Absența warning-urilor (tema compilează cu succes folosind flag-ul -Werror).
- **5p:** Indentare corectă și coding style.

5 Detalii suplimentare

- O transmisie corectă a fișierului prespune ca fișierul recepționat să fie identic cu cel transmis.
- O transmisie eficientă a fișierului presupune ca timpul de transmisie să fie maxim 1.25 * dimensiune fișier / bandwidth + delay.

- Pentru transmiterea mesajelor, se va folosi structura msg din lib.h. Nu este permisă modificarea acestei structuri.
- Nu aveți voie să utilizați alte mecanisme pentru timeout în afara celui pus la dispoziție (recv_message_timeout).
- Pentru lucrul cu fișiere, veți utiliza funcțiile open, read, write, și close. În acest sens, puteți consulta rezolvarea laboratorului 2 sau paginile de manual din Linux(man open). Dimensiunea fișierului transmis este între 100KB și 10MB.
- Se recomandă folosirea utilitarului dd pentru a genera fișiere de test.
  ex: dd if=/dev/urandom of=test file bs=512k count=4
- După ce vă creați propria structură, pe care o veți transmite în cadrul câmpului payload al structurii msg, aveți grijă ca sizeof(structura voastră) să coincidă cu dimensiunea payload-ului, adică sizeof(structura voastră) să fie 1400. Dacă nu coincide, posibil din motive de aliniere, atunci scădeți dimensiunea payload-ului din structura voastră pentru a atinge dimensiunea de 1400.
- Testele vor cuprinde valori pentru SPEED și DELAY astfel încât dimensiunea fereastrei să nu depășească 1000 de cadre.
- Implementarea temei se poate face în C sau C++.

6 Trimiterea temei

- Denumire arhivă: NumePrenume_Grupă_Tema1.zip
- Conținut arhivă:
  - send.c
  - recv.c
  - alte surse adăugate de voi
  - Makefile
  - README (format txt)
- Nu este necesar să includeți folderul link_emulator. Acesta va fi suprascris la corectarea temei, inclusiv fișierul lib.h.
- Pentru trimiterea temelor se va folosi interfața vmchecker.
- Depunctările pentru întârzieri se vor aplica conform regulamentului.