PROTOCOALE DE COMUNICAȚIE : Tema 1

MINI-KERMIT

Responsabili: Cătălin LEORDEANU, Alecsandru PĂTRAȘCU, Radu CIOBANU, Bianca OPREA
Termen de predare: 2 APRILIE 2018

Cuprins

Descriere generala **1**

Structura pachetului MINI-KERMIT **2**

Tipuri de pachete **2**
Pachet "S" (Send-Init) 2
Pachet "F" (File Header) 3
Pachetul "D" (Date) 3
Pachetul "Z" (EOF) 3
Pachetul "B" (EOT) 3
Pachet "Y" (ACK) 3
Pachet "N" (NAK) 3

Tratarea timeout-ului **3**

Tratarea erorilor de comunicatie **4**

Tratarea pierderii pachetelor **4**

Exemplu **4**

Detalii implementare si rulare **4**

Predarea temei **6**

Notarea temei **6**

Se cere implementarea protocolului KERMIT, in format redus (MINI-KERMIT), pentru transfer de fisiere, folosind coduri ciclice CRC pentru detectia erorilor. Pentru simularea mediului de comunicatie se va folosi aplicatia *link_emulator*.

Descriere generala

Protocolul KERMIT este un protocol ce face parte din clasa protocoalelor ARQ (Automatic Repeat Request), in care un pachet eronat sau neconfirmat este automat retransmis. Datele utile sunt impachetate, fiind inconjurate cu unele campuri de control. In timpul transmiterii unui pachet nu se face controlul fluxului. Fiecare pachet trebuie confirmat.

Avantajul protocolului KERMIT fata de altele asemanatoare este simplitatea de implementare, precum si:

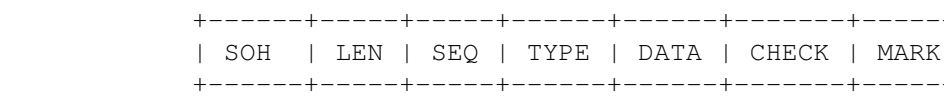
- Negocierea anumitor parametri de comunicatie intre emitator si receptor prin intermediul primelor pachete
- Posibilitatea de transfera mai multe fisiere in cadrul unei sesiuni
- Transmiterea numelor fisierelor
- Posibilitatea ca pachetele sa aiba tipuri si lungimi variabile

Transferul unui fisier decurge ca la orice protocol ARQ. Receptorul primeste pachetul, si dupa ce verifica numarul sau de secventa fata de precedentul pachet, calculeaza o suma de control locala pentru partea de date. Daca suma de control calculata coincide cu cea sosita se emite o confirmare pozitiva ACK (caracter sau pachet); in caz contrar se emite confirmare negativa NAK. In final se transmite un pachet de tipul EOF. Daca mai exista fisiere de transmis, se transmite header-ul urmatorului, iar in final un pachet de tipul EOT.

Structura pachetului KERMIT in forma redusa, numit MINI-KERMIT, precum si tipurile de pachete, vor fi detaliate in sectiunea urmatoare.

Structura pachetului MINI-KERMIT

Un pachet MINI-KERMIT are 7 campuri :



Semnificatia campurilor si dimensiunea acestora este:

- SOH (1 byte): Marcheaza inceput de pachet (Start-of-Header). Valoarea este 0x01.
- LEN (1 byte) : Numar de bytes care urmeaza acestui câmp (lungime_pachet - 2).
- SEQ (1 byte): Numar de secventa, modulo 64. Valoarea initiala este 0x00. Aceasta valoare se incrementeaza continuu de emitator si receptor
- TYPE (1 byte): Tipul pachetului, poate fi unul dintre caracterele:
  - 'S': Send-init (primul pachet care se transmite)
  - 'F': File Header
  - 'D': Data
  - 'Z': EOF (End Of File)
  - 'B': EOT(End Of Transaction) - intreruperea transmisiei
  - 'Y': ACK
  - 'N': NAK
  - 'E': eroare
- DATA ([0...MAXL] bytes): Datele care se transmit. Poate fi si vid. Campul MAXL este detaliat in sectiunea urmatoare.
- CHECK (2 bytes): Suma de control, calculata pe toate campurile in afara de campurile MARK si CHECK
- MARK (1 byte): Marcheaza sfarsitul de pachet (End of Block Marker). Valoarea sa este definita in sectiunea urmatoare, la tipul de pachet "S".

Tipuri de pachete

Pachet "S" (Send-Init)

Anunta receptorul asupra preferintelor si setarilor emitatorului. Are numarul de secventa 0. In cadrul acestui tip de pachet, campul DATA are urmatoarea structura:

MAXL	TIME	NPAD	PADC	EOL	QCTL	QBIN	CHKT	REPT	CAPA	R

Campuri obligatorii MINI-KERMIT(fiecare de cate 1 byte lungime):

- MAXL: dimensiunea maxima a campului DATA.
- TIME: durata de timeout pentru un pachet, in secunde.
- NPAD: numarul de caractere de padding emise inainte de fiecare pachet. Implicit este initializat cu valoarea 0x00.
- PADC: caracterul folosit pentru padding. Implicit este NUL (0x00) si este ignorat daca NPAD are valoarea 0x00.
- EOL: caracterul folosit in campul MARK. Implicit este caracterul 0x0D (CR).

Campurile QCTL, QBIN, CHKT, REPT, CAPA si R au cate 1 byte lungime si valoarea implicita 0x00. Desi nu sunt folosite efectiv in MINI-KERMIT, ele vor fi definite in cadrul pachetului.

Pachet "F" (File Header)

In cadrul acestui tip de pachet, campul DATA contine numele fisierului de transmis. Nu se specifica nume de cale sau drive. Singurele caractere permise in numele unui fisier transmis sunt cifre, litere si caracterul '.'.

Pachetul "D" (Date)

Contine datele utile transmise (portiuni din fisier).

Pachetul "Z" (EOF)

Este transmis dupa fiecare fisier transmis integral. Campul DATA e vid.

Pachetul "B" (EOT)

Este transmis la finalul tranzactiei (transmisiei), dupa ce au fost transferate toate fisierele. Campul DATA e vid.

Pachet "Y" (ACK)

Campul DATA e vid, cu exceptia confirmarii unui pachet "S", cand contine setarile receptorului. Structura campului DATA in acest caz este aceeasi ca la pachetul "S".

Pachet "N" (NAK)

Campul DATA e vid.

Tratarea timeout-ului

In cazul emitatorului, valoarea scrisa in campul TIME reprezinta numarul de secunde pe care il asteapta pentru primirea unui pachet de tipul ACK/NAK. In caz de timeout, se va retransmite ultimul pachet, pastrand valoarea campului SEQ.

In cazul receptorului, valoarea scrisa in campul TIME reprezinta numarul de secunde pe care il asteapta pentru primirea urmatorului pachet (SEQ+1). In caz de timeout, se va transmite ultimul pachet (ACK sau NAK). Pentru pachetul de tipul "S", in caz de timeout, nu se retransmite nimic, ci se asteapta maxim de inca 2 ori cate TIME secunde primirea acestuia.

Tratarea erorilor de comunicatie

In cazul in care apar erori de comunicatie si valoarea din campul CHECK nu este validata, receptorul va transmite un mesaj de tipul NAK catre emitator. Emitatorul va trebui sa retransmita ultimul pachet, actualizand valoarea din campul SEQ.

Daca valoarea din campul MARK este corupta, dar CHECK este validat, receptorul va transmite un mesaj de tipul ACK, ceea ce simbolizeaza faptul ca desi pachetul are ultimul byte corupt, informatia utila din el este intacta.

Tratarea pierderii pachetelor

In cazul in care un pachet este pierdut pe canalul de comunicatie dintre emitator si receptor, acesta se va retransmite, pastrand valoarea din campul SEQ.

Operatia de retransmitere a pachetelor se poate face de maxim 3 ori per pachet. Daca un pachet a fost retransmis de mai mult de 3 ori, transmisia se va intrerupe, iar emitatorul/receptorul isi va incheia executia.

Exemplu

Succesiunea de pachete, pentru transmiterea a 2 fisiere, ar putea fi precum in Tabelul 1. In parantezele drepte este scrisa valoarea campului SEQ.

Detalii implementare si rulare

In cadrul temei veti implementa, in C/C++, emitatorul si receptorul in fisiere sursa separate, pornind de la scheletul de cod atasat enuntului temei. Programele rezultate in urma compilarii se vor numi **ksender** si **kreceiver**.

Acestea se vor lansa cu urmatoarele linii de comanda:

```
./ksender nume_fis1 [nume_fis2 ... nume_fisn]
```

si

```
./kreceiver
```

unde *nume_fisX* reprezinta numele fisierului/fisierelor ce vor fi transferate.

Atat pentru emitator, cat si pentru receptor, campurile din pachetul de tip "S" vor avea urmatoarele valori:

- MAXL: 250 bytes
- TIME: 5 secunde

Tabelul 1: Exemplu de succesiune a pachetelor

Emitator	**Receptor**
[0] Send-init (cu parametrii emitatorului)	
	[1] ACK pentru [0] (cu parametrii receptorului)
[2] File Header 1	
	[3] ACK pentru [2]
[4] Data 1	
	[5] ACK pentru [4]
Reteaua este congestionata si pachetul [5] nu ajunge la Emitator in timp util. Receptorul nu primeste pachetul cu SEQ [6] in TIME secunde	
	[5] ACK pentru [4]
[6] Data 2	
Pachetul [6] ajunge la Receptor cu erori	
	[7] NAK pentru [6]
[8] Data 2	
	[9] ACK pentru [8]
[10] EOF	
	[11] ACK pentru [10]
[12] File Header 2	
	[13] ACK pentru [12]
[14] Data	
Pachetul [14] nu ajunge la Receptor si nu se primeste pachetul de ACK cu SEQ [15] in TIME secunde	
[14] Data	
	[15] ACK pentru [14]
[16] EOF	
	[17] ACK pentru [16]
[18] EOT	
	[19] ACK pentru [18]

Compilarea si rularea scheletului de cod:

```
cd link_emulator
make clean
make
cd ..
make
chmod +x run_experiment.sh
./run_experiment.sh
```

Alte detalii:

- Implementarea tratarii primirii pachetelor corupte sau a unui TIMEOUT se poate face in mai multe feluri si de aceea nu vom da detalii pe forum in acest sens, pentru a nu va influenta ideea de rezolvare in vreun fel. Algoritmul ales de voi va trebui detaliat in README.
- Nu trebuie sa fie modificata structura *msg* definita in *lib.h*. De asemenea, se vor utiliza doar campurile *len* si *payload*. Va trebui sa definiti una sau mai multe structuri pentru gestionarea tipurilor de mesaje, care sa fie scrisa/scrise in *payload*.
- Pentru a evita adaugarea de catre compilator a padding-ului intre campurile structurilor definite de voi, va trebui sa folositi atributul de GCC: *__attribute__((__packed__))*.
- Pentru calculul CRC se va folosi functia *crc16_ccitt()* definita in lib.h. Un exemplu de folosire gasiti in scheletul de cod.
- Pentru timeout, se va folosi functia *receive_message_timeout()*, definita in lib.h. Un exemplu de folosire

gasiti in scheletul de cod.
- *Link_emulator* va face pierderea pachetelor si coruperea bitilor numai in sensul de comunicatie de la emitator la receptor. Ambele evenimente vor trebui detectate si trimis NAK sau retransmis ultimul pachet. *Atentie! Se poate modifica orice byte de informatie din cadrul structurii definite de catre voi.*
- Fisierele transmise vor trebui sa interpretat la nivel binar.
- Drept fisiere puteti folosi orice fisiere locale pe care le aveti sau le puteti genera folosind comanda *dd* (variind parametrii *bs* si *count*). Exemplu de folosire *dd* pentru a genera un fisier de 512 bytes: *dd if=/dev/urandom of=fisier1.bin bs=128 count=4* (se genereaza 4 blocuri a cate 128 bytes fiecare).
- Fisierele din scheletul de cod sunt doar un posibil exemplu.
- La receptor, veti prefixa numele fisierului primit ca parametru cu *"recv_"*. De exemplu, pentru fisierul *"input1.txt"* primit ca parametru de la emitator, la receptor va trebui sa salvati datele in fisierul *"recv_input1.txt"*.
- Pentru lansarea in executie, folositi *run_experiment.sh*. Numele si numarul fisierelor primite ca parametru de ksender puteti sa le modificati. Nu este permisa modificarea valorilor campurilor SPEED, DELAY, LOSS sau CORRUPT.
- Tema va fi testata pe Linux si compilata cu GCC. Chiar daca folositi alte sisteme de operare si compilatoare pentru dezvoltare, asigurati-va ca tema compileaza si ruleaza pe un sistem Linux. O tema care nu compileaza va fi punctata cu 0p.
- Testarea temei se face automat de catre scriptul *run_experiment.sh* folosind utilitarul *diff* intre fisierul sursa si cel destinatie.

Predarea temei

Fișierele și directoarele care contribuie la rezolvarea temei trebuie OBLIGATORIU impachetate într-o arhivă de tip ZIP, cu numele 'Grupa_Nume_Prenume_TemaX.zip' (de exemplu, studenta Stan Sonia de la grupa 321CA va trimite o arhivă cu numele 321CA_Stan_Sonia_Tema1.zip).

Arhiva trebuie sa contina intreg codul sursa pentru emitator si receptor, un fisier *README* in care sa detaliati implementarea si fisierul Makefile pentru a compila emitatorul si receptorul. Puteti modifica fisierul *Makefile* din schelet oricum doriti, singura restrictie fiind existenta regulilor de **build** si **clean**.

ATENTIE! Fisierele din scheletul de cod vor trebui incluse de asemenea de voi in arhiva!

Tema nu va fi testata pe vmchecker.

Notarea temei

Se acordă 10 puncte pentru o tema care este rezolvata si functioneaza conform cerintelor ei. Criteriile luate in calcul la notarea temei sunt:

1. Functionarea conform cerintelor. Pentru a primi punctaj maxim, solutia trebuie sa treaca toate testele de la corectare.
2. Calitatea si eficienta solutiei temei. Atentie la respectarea cerintelor explicite referitoare la eficienta implementarii. Nu trebuie folosite mecanisme ineficiente unde se poate folosi ceva mai bun. De exemplu:
   - mesaje inutil de lungi sau trafic inutil de mesaje
   - algoritmi ineficienti.
3. Claritatea codului. Codul trebuie sa fie ușor de urmarit. Se vor lua in considerare: modularizarea, indentarea corecta, nume de variabile sugestive, cod aerisit, etc.
4. Claritatea explicatiilor. Pentru a primi maximum de punctaj, explicatiile trebuie sa fie clare si concise. Prin explicatii se intelege:
   - comentariile din codul sursa
   - explicatii din fisierul README.