https://dfilip.xyz/pcom

Dorinel Filip
dorinel.filip@upb.ro
Subiect: [PCom] ….

O legatura de date se caracterizeaza prin:

- Latimea de banda
- Un delay specific (cat dureaza sa ajunga datele din A la B)
- Un nivel de incredere (corupe date, pierde date, ….)
- Cam orice protocol de L2 are si un MTU (Maximum Transmission Unit) - cat de mare poate fi un cadru

Scenariul de lab 2:

- Nu corupe date si nu pierde date

Task:

- Sa trimitem un fisier de la send la recv
    - ./send fisier
    - ./recv
        - Recv sa salveze fisierul foo.txt ca foo.txt.bk
- Sa implementam un protocol de tipul START-STOP pentru controlul congestiei

Controlul congestiei apare la L2 cand avem protocoale cu bandwith garantat (NU la ethernet)

De ce nu START-STOP

MTU: 1500 octeti
Delay de 100ms

200ms pentru un frame => 5 frame/s = 7500 octeti / secunda

```
#!/bin/bash

SPEED=1 # 1Mbps
DELAY=1 # 1ms
LOSS=0  # nu se pot pierde pachete
CORRUPT=0 # nu se corup

{
    killall link
```

```
    killall recv
    killall send
} &> /dev/null

./link_emulator/link speed=$SPEED delay=$DELAY loss=$LOSS
corrupt=$CORRUPT &> /dev/null &
sleep 1
./recv &
sleep 1

./send fisier.exemplu
```

// Pasul 1
Senderul sa ii spuna recv cu ce nume sa salveze fisierul

// Pasul 2
sa trimitem continutul fiserului
Observatie: pot fi mai multe mesaje (1MB -> bucati de 1400 octeti)

// Pasul 3:
Recv sa afle cand a primit toate bucatile

Var 1:
---> Senderul (mesaj 2) ii trimite lungimea
---> Recv scade din lungime cat a primit -> 0 == break

Var 2:
----> Un mesaj de terminare
----> if r.len == 0, => inchide

Observatie: send trimite mesajul respectiv

```
struct msg {
        int len;
        char payload[MAX_LEN];
}
```

Mesaj = 3000

// Var 1 (3000 ca int, len = sizeof(int))

struct msg m;

*(int *)m.payload = 3000;

```
m.len = sizeof(int);

send_message(&m);

Problema: endianess / diferenta la sizeof(int) => uint32_t

// Var 2 (recomandata azi, trimitem val ca string)

// send
sprintf(m.payload, "%d", 3000);
m.len = strlen(m.payload) + 1;

send(...) + recv(&ack)

// recv
recv_message(&m); + send(ack)

sscanf(m.payload, "%d", &x); // atoi(m.payload)


// Receiver
recv(&nume_fisier);

// deschid fisier
fisier

while(1) {
        primesc bucati si le scriu in fisier
}

inchid fisierul

// Sender

trimite_nume

while(1) {
        citesc din fisier si trimite bucati
}



Verificare:
dd if=/dev/urandom of=fisier bs=1K count=40
```

diff original copie