CS4514 Project 2
Help Session
(B06)

Mingzhe Li
Nov 9, 2006
The goal is to implement a Positive Acknowledgement with Retransmission (PAR) protocol on top of an emulated physical layer.

- The receiver acknowledges only the correctly received segments and the sender uses timeout to detect and send the lost segment.
- Physical layer is emulated by a TCP connection plus an error module.
- Your programs should compile and work on “ccc[1-10].wpi.edu”
Framework

Do NOT attempt to put everything in one big main()
Network Layer

Client

photo[1-5].jpg

read (block)

Network Layer

nwl_recv(ack)

pkt

Network layer ack

pkt

Server

photonew[1-5].jpg

write (block)

Network Layer

nwl_recv(pkt)

pkt

Network layer ack

Packet size: 200 bytes
Data Link Layer

Client

Network Layer

Network layer ack

dll_send(pkt)

Data Link Layer

dll_recv(frm)

ack/frm
(Network Layer ack)

frm

Server

Network Layer

Network layer ack

pkt

dll_send(pkt)

Data Link Layer

dll_recv(frm)

frm

ack/frm
(Network Layer ack)

Client link layer does not need to ACK “Network Layer ACK” frame!
Physical Layer

Client

Datalink Layer

ack

phl_send()

Physical Layer

phl_recv()

Server

Datalink Layer

frm

phl_send()

Physical Layer

phl_recv()

TCP Connection
Client: dll_send(pkt, ...)

1. Read a block and split into payloads

2. For each payload

2.1 Create Frame (frm)

2.2 Start a Timer

2.3 phl_send(frm, ...)

2.4 phl_recv(ack/frm,...)

ack/frm?

frm

ack ok?

no

ack

yes

2.2.1 Timeout Handler: phl_send(frm, ...)

2.4.1 nwl_ack received

3. Waiting for Network Layer ack:
   if (!nwl_ack) phl_recv(frm,...)

client.log

phl_send(frm, ...):
Force bit error every 5-th Frame
Create Frame

1. Compute Seq Number, Frame Type and End-Of-Packet (EOP) bytes

2. Error-Detection (ED) bytes (XOR on Seq + FT + EOP + Data)

EOP: End of Packet
FT: Frame Type
ED: Error Detection
Seq: Sequence Num
Server: dll_recv(frm, ...)

1. Compute ED byte
2. Create ACK Frame (ack)
3. phl_send(ack, ...)
4. Reassemble the packet
5. nwl_recv(pkt, ...)

ED ok?

no

Dup?

yes

Drop frm

Return

phl_send(ack, ...): Force bit error every 7-th Frame

EOP?

no

yes

server.log
Create ACK Frame

1. Compute Seq Number and Frame Type

2. Error-Detection (ED) bytes (ED = Seq)

EOP: End of Packet
FT: Frame Type
ED: Error Detection
Seq: Sequence Num
Timers

- The client uses a timer to detect a frame loss.
  - The client sets a timer when it transmits a frame.
  - When the timer expires, the client retransmits the frame.

- Two kinds of timer
  - Select: easier to use
  - Signal and Timer: nicer implementation
# include <sys/select.h>
# include <sys/time.h>

int select (int maxfdp1, fd_set *readset, fd_set *writeset,
           fd_set *exceptset, const struct timeval *timeout);

struct timeval {
    long tv_sec; /* seconds */
    long tv_usec; /* microseconds */
}
Example: Select

```c
fd_set bvfdRead;
int readyNo;
struct timeval timeout;
int sockfd;

while (1) {
    timeout.tv_sec = 0;
    timeout.tv_usec = 500;
    FD_ZERO(&bvfdRead);
    FD_SET(sockfd, &bvfdRead);
    readyNo = select(sockfd+1, &bvfdRead, 0, 0, &timeout);
    if(readyNo < 0)
        error_handler();
    else if(readyNo == 0)
        timeout_handler();
    else {
        FD_ZERO(&bvfdRead);
        receive_handler();
    }
}
```
Signal and Timer: Soft Interrupt

- **Head files**
  
  ```c
  #include <signal.h>
  #include <time.h>
  ```

- **Register a function to TIMEOUT signal**
  
  ```c
  signal(SIGALRM, timeout);
  ```

- **Create a timer and begin to run**
  
  ```c
  timer_create();
  timer_settime();
  ```

- **Compile with option “-lrt” (link runtime library)**
Example: Signal and Timer

timer_t timer_id;

void timeout(int signal_number){
    printf("\n SIGNUM: %d\n", signal_number);
    exit(0);
}

void start_timer(){
    struct itimerspec time_val;
    signal (SIGALRM, timeout);
    timer_create(
        CLOCK_REALTIME,
        NULL, &timer_id);

    /* set timeout to 1 second */
    time_val.it_value.tv_sec = 1;
    time_val.it_value.tv_nsec = 0;
    time_val.it_interval.tv_sec = 0;
    time_val.it_interval.tv_nsec = 0;
    timer_settime(timer_id, 0,
        &time_val, NULL);
}

main(){
    start_timer();
    while(1);
}
Open a File

- Open a file for read:
  ```c
  int rfile;
  if ((rfile = open("filename1", O_RDONLY)) < 0) {
    perror("Input File Open Error");
    exit(1);
  }
  ```

- Open a file for write (create if not exist):
  ```c
  int ofile;
  if ((ofile = open("filename2", O_WRONLY|O_CREAT|O_TRUNC, S_IRUSR|S_IWUSR|S_IRGRP|S_IWGRP)) < 0) {
    perror("Output File Open Error");
    exit(1);
  }
  ```
Read from file

while ((rd_size = read(rfile, buf, 256)) > 0)
{
    do something with “buf” here
}

if (rd_size < 0)
{
    perror("File Read Error");
    exit(1);
}
else
{
    printf ("Reach the end of the file\n");
}
File Write/Close

- **Write to File**
  
  ```c
  if ((wr_size = write(ofile, buf, rd_size)) < 0)
  {
    perror("Write Error:");
    exit(1);
  }
  ```

- **Close files**
  
  ```c
  close(rfile);
  close(ofile);
  ```
Display Image in Linux

- Make sure you have “X forwarding” with your ssh client
- And you need have an Xserver (X-Win32 or etc.) running on you windows computer.
- The image display is not required for the Project.
- These code tested on ccc[1-10].wpi.edu

```c
if (fork() == 0) {
    execl("/usr/local/bin/xv", "xv", "image.jpg", NULL);
} else {
    wait(NULL);
    printf("Done display! \n");
}
```
Thanks!
and
Questions?