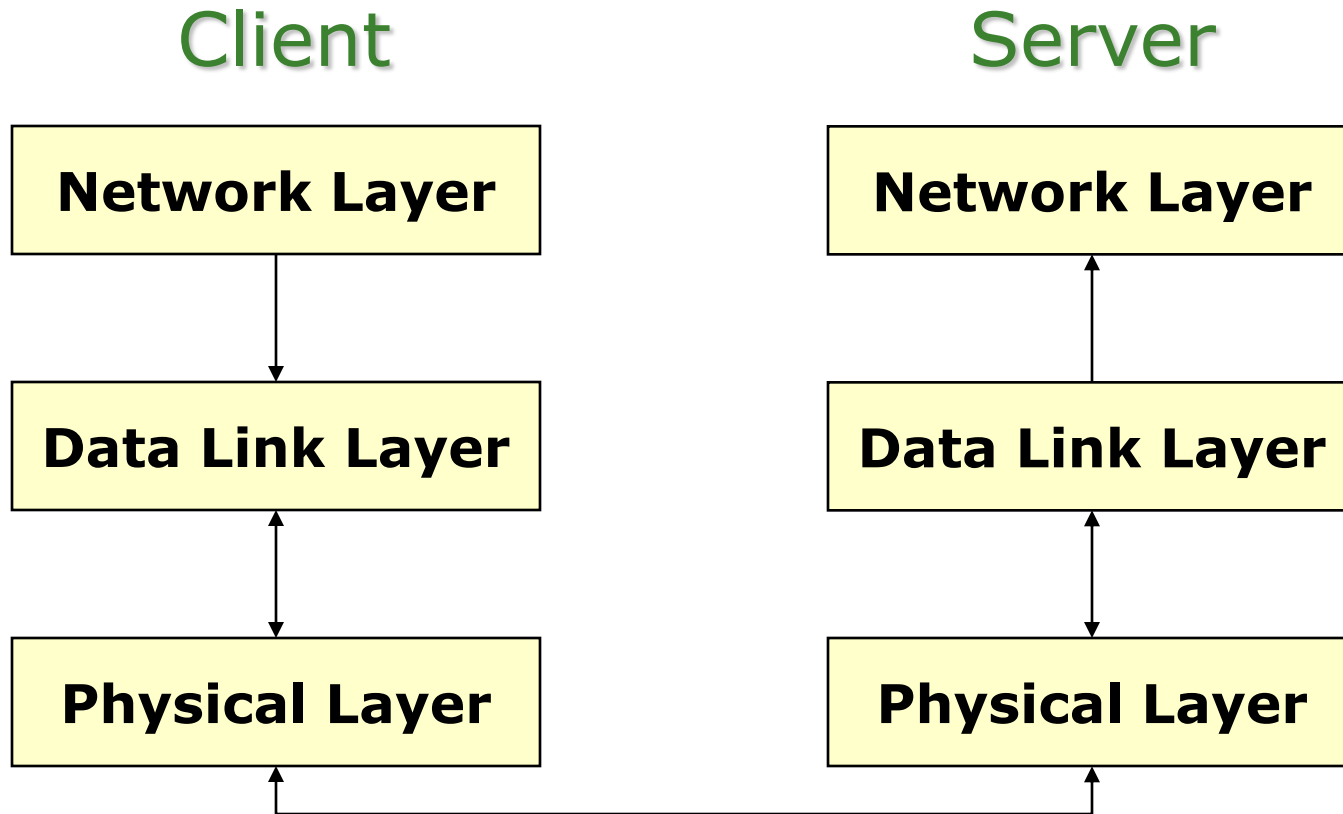

CS4516 Program 2 Help Session (C10)

Choong-Soo Lee and Rabin Karki
Jan. 26, 2010

Description

- The goal is to implement a concurrent server with 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 "cccwork[1-2].wpi.edu"

Framework



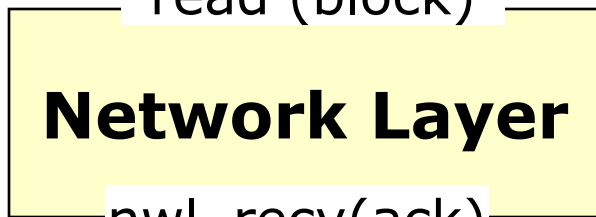
Do NOT attempt to put everything in one big main()

Network Layer

Client

photo[1-5].jpg

read (block)



Network Layer

nwl_rcv(ack)

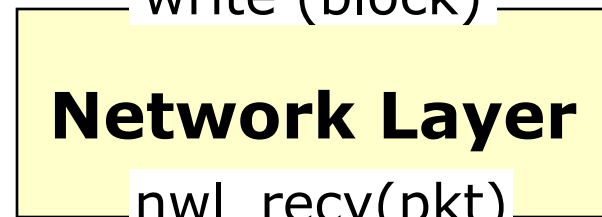
Network
layer ack

pkt

Server

photonew[id][1-5].jpg

write (block)



Network Layer

nwl_rcv(pkt)

Network
layer ack

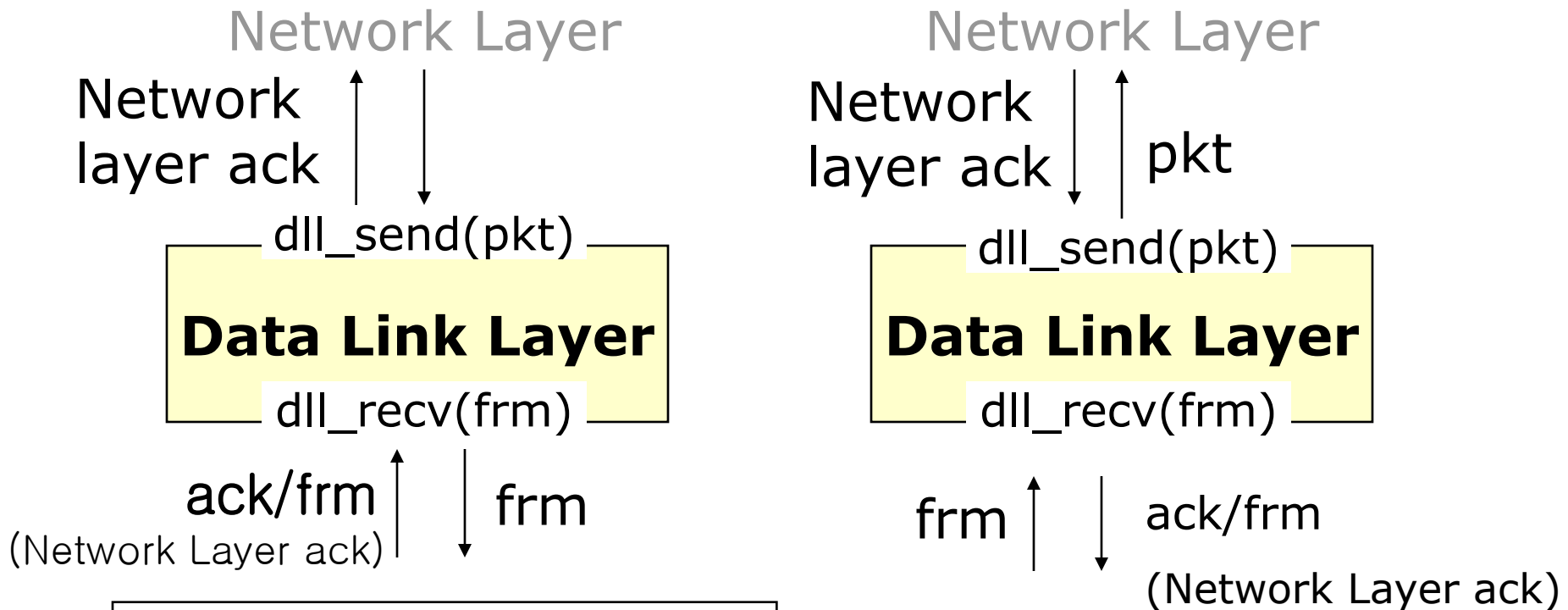
pkt

Packet size: 128 bytes

Data Link Layer

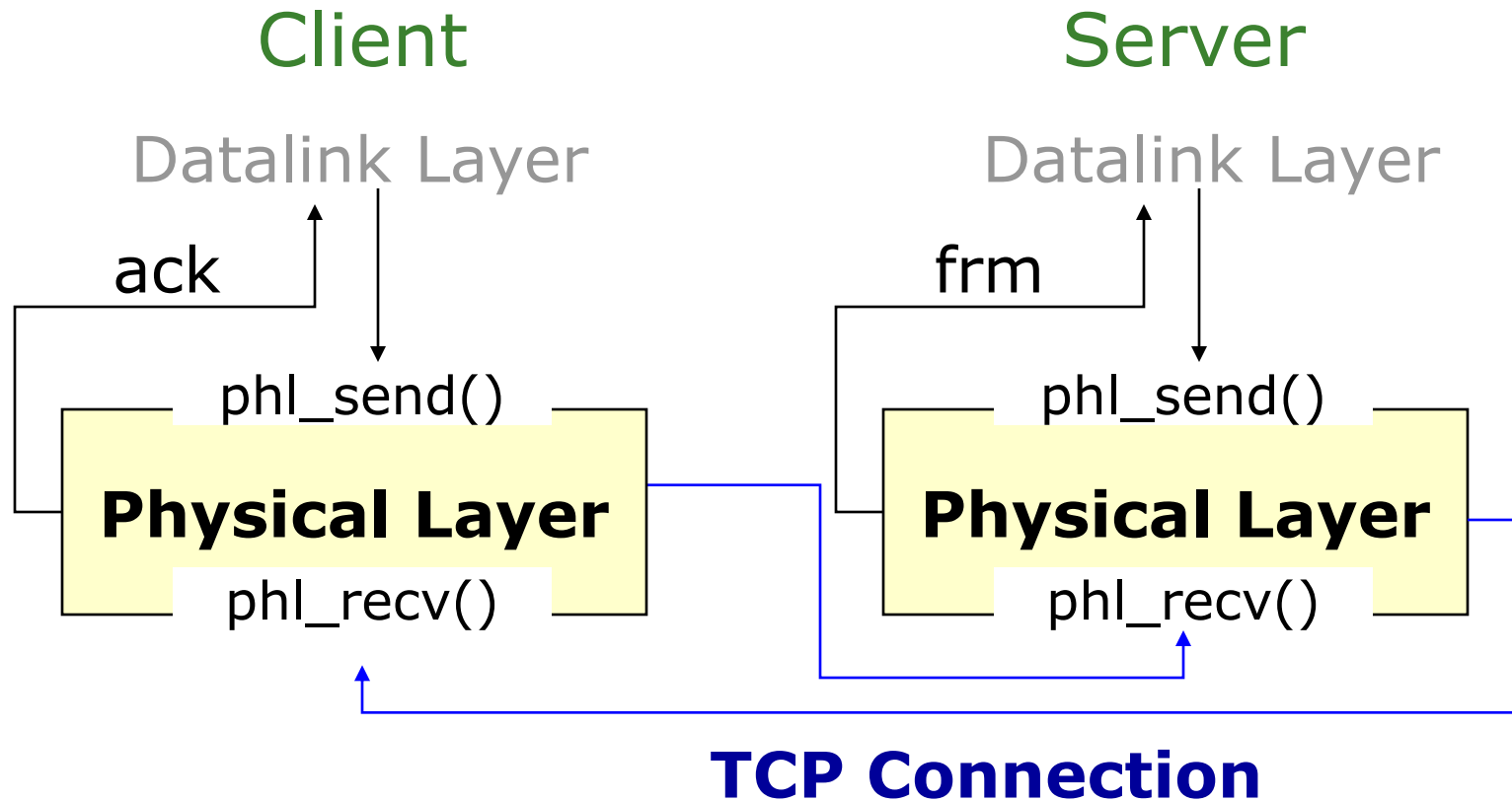
Client

Server

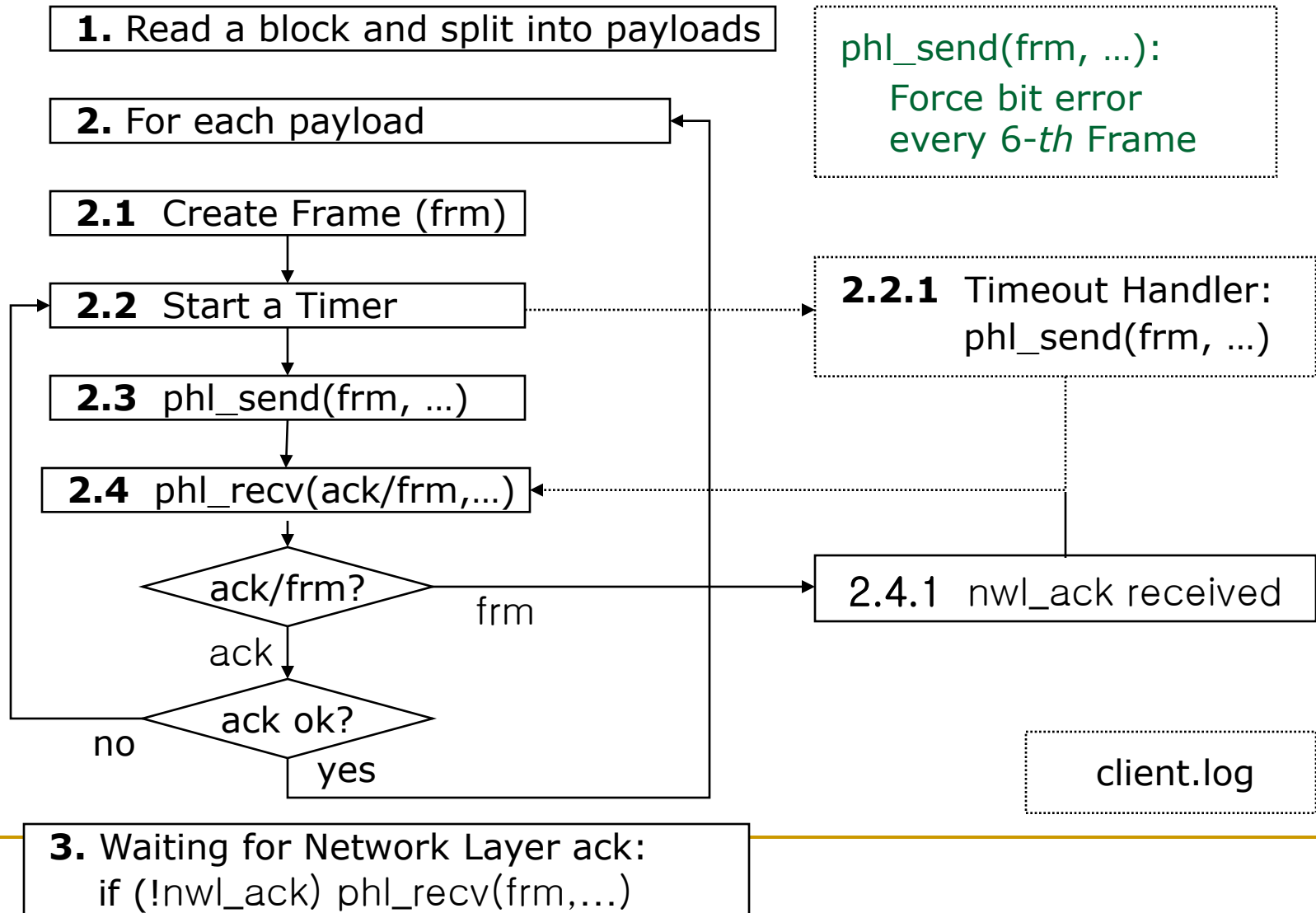


Client link layer does not need to ACK "Network Layer ACK" frame!

Physical Layer



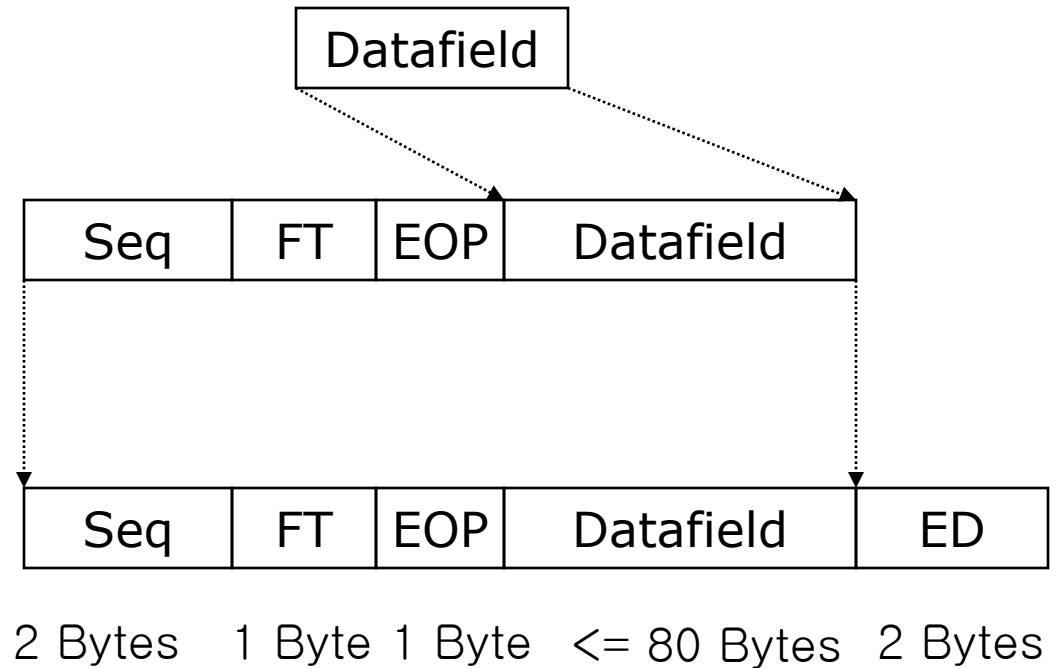
Client: dll_send(pkt, ...)



Create Frame

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

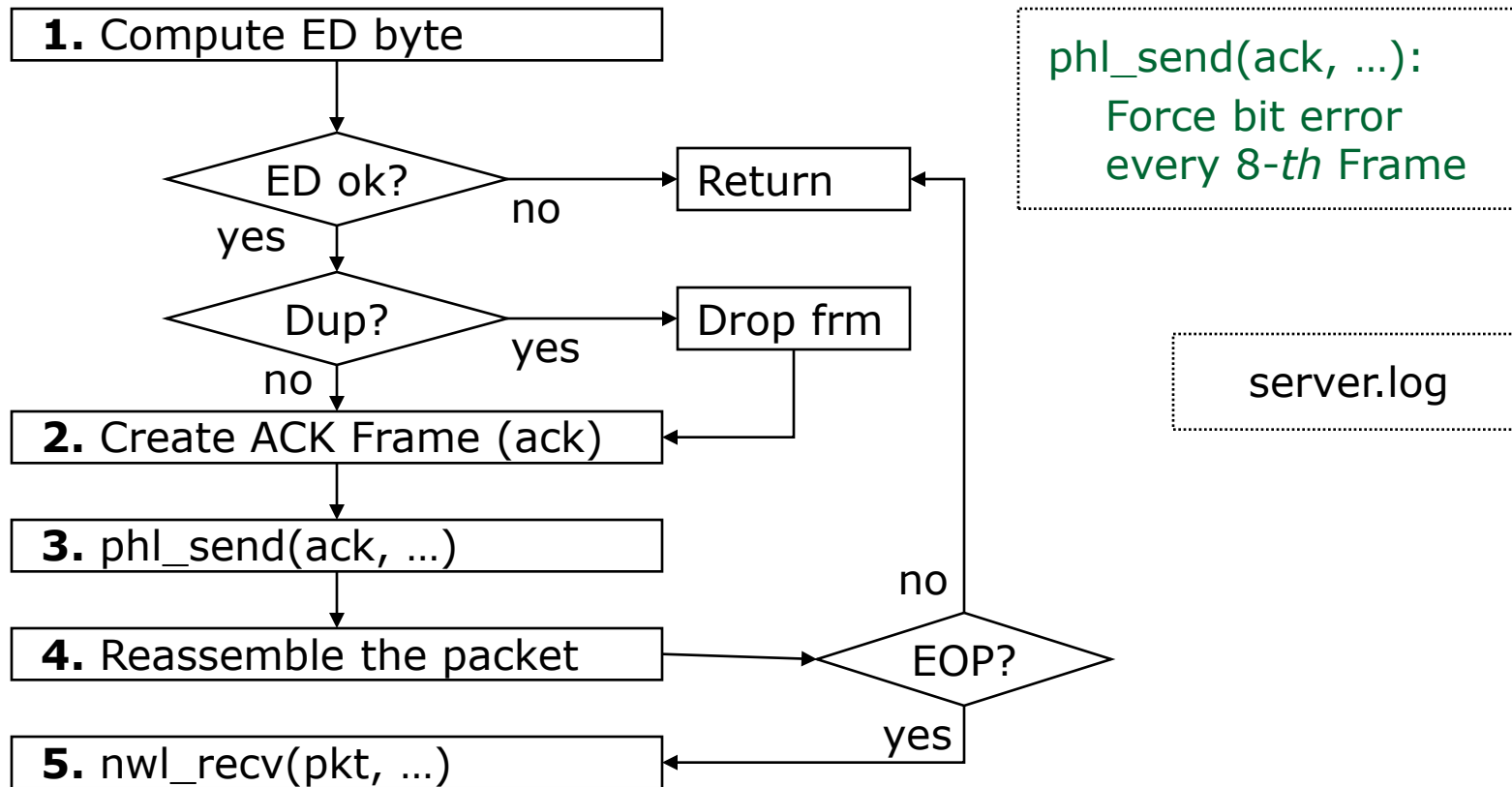
2. Error-Detection (ED) bytes
(XOR on Seq + FT + EOP + Data)



EOP: End of Packet
FT: Frame Type

ED: Error Detection
Seq: Sequence Num

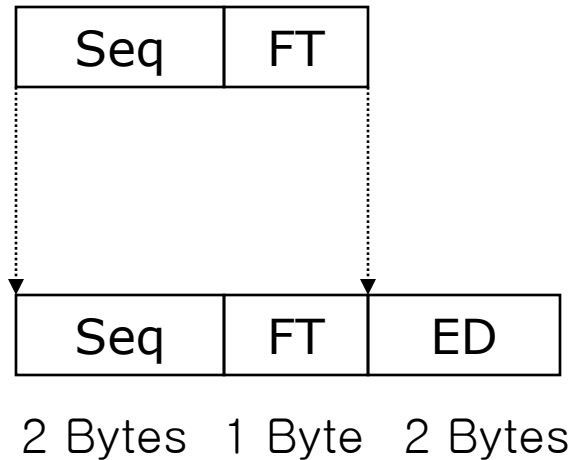
Server: dll_rcv(frm, ...)



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

Select: Monitor Given FDs (SDs)

```
# include <sys/select.h>
# include <sys/time.h>

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

struct timeval {
    long tv_sec;           /* seconds */
    long tv_usec;        /* microseconds */
}
```

Example: Select

```
fd_set bvfdrRead;
int readyNo;
struct timeval timeout;
int sockfd;

while (1) {
    timeout.tv_sec = 0;
    timeout.tv_usec = 500;
    FD_ZERO(&bvfdrRead);
    FD_SET(sockfd, &bvfdrRead);
```

```
    readyNo = select(sockfd+1,
                    &bvfdrRead, 0, 0, &timeout);

    if(readyNo < 0)
        error_handler();
    else if(readyNo == 0)
        timeout_handler();
    else {
        FD_ZERO(&bvfdrRead);
        receive_handler();
    }
}
```

Signal and Timer: Soft Interrupt

- Head files

```
#include <signal.h>
#include <time.h>
```
- Register a function to TIMEOUT signal

```
signal (SIGALRM, timeout);
```
- Create a timer and begin to run

```
timer_create();
timer_settime();
```
- Compile with option “-lrt” (link runtime library)

Example: Signal and Timer

```
timer_t timer_id;

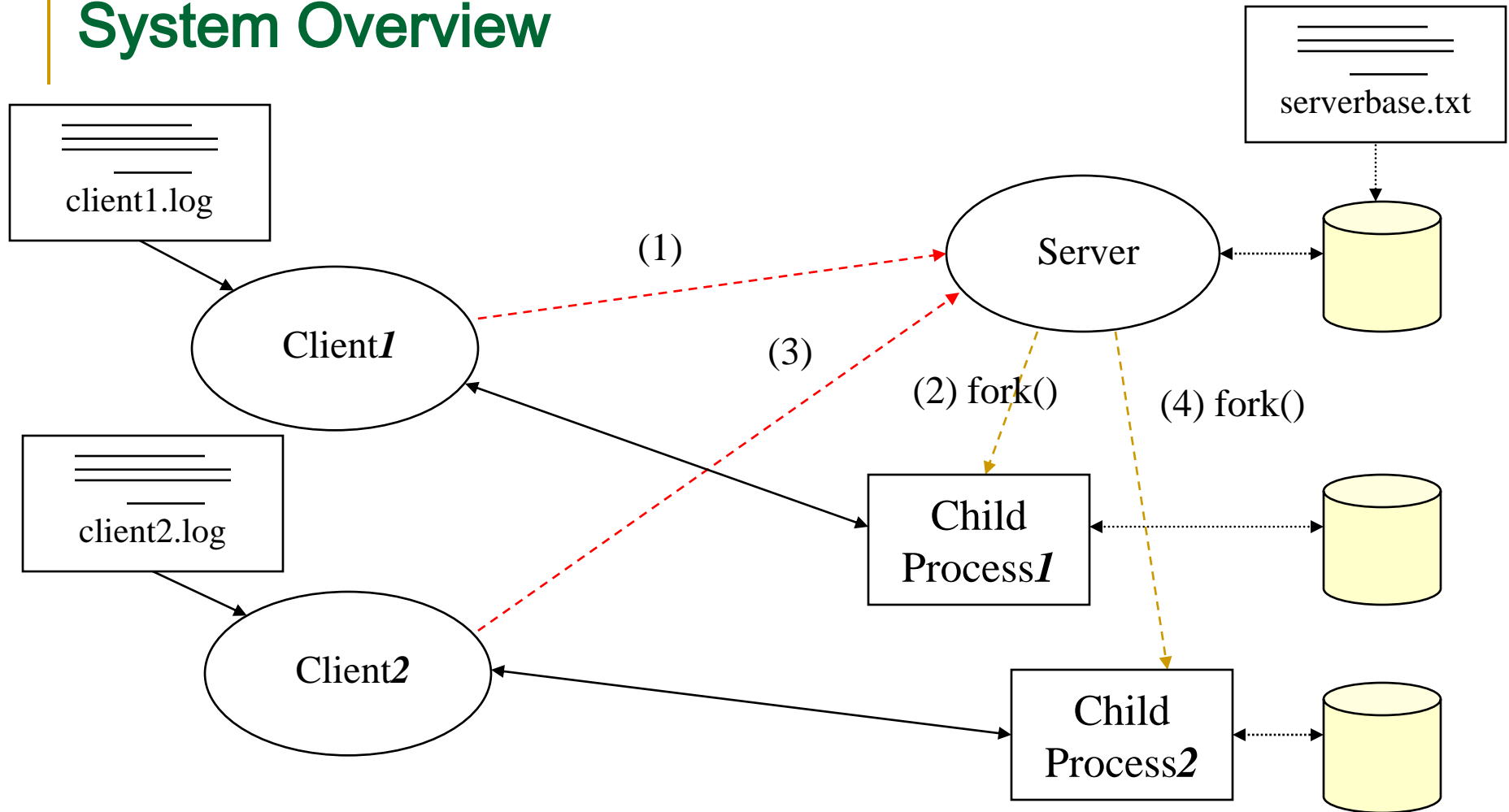
void timeout(int signal_number){
    printf("\n SIGNUM: %d\n",
           signal_number);
    exit(0);
}

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);
}
```

System Overview



Note: each child process keeps a separate copy of the DB.

we do not keep data consistency for the serverbase

This is automatically done by using `fork()`

Concurrent Photo Server

- Server will process requests in parallel (hence, concurrent server)
 - There are several ways to achieve concurrency
 - **Using fork()**
 - Using multiple threads
 - Using select()
-

fork()

- It splits current process into two processes: a parent and a child.
 - How does the program determine whether it is the parent or the child after the split?
 - The *fork()* command returns 0 if it is the child.
 - So all we need to do is check the return value from *fork()*.
 - Now you can program the child to handle the data transmission and the parent can continue on to accept other requests.
-

Example

```
pid_t pid, id;
int listenfd, connfd;
/* 1. create a socket socket() */
if ((listenfd = socket(AF_INET, SOCK_STREAM, 0)) < 0){
    perror("Error creating socket");
    exit(1); }
/* 2. fill in sockaddr_in{ } with server's well-known port */
...
/* 3. bind socket to a sockaddr_in structure bind() */
bind (listenfd, ...);
/* 4. specify the backlog of incoming connection requests listen() */
listen (listenfd, 5);
while(1){
    connfd = accept(listenfd, ... );
    if(( pid = fork()) == 0){
        close(listenfd); /* child closes listening socket */
        doit(connfd); /* process the request. this is where the work is done. */
        close(connfd); /* done with this client */
        exit(0);
    }
    close(listenfd); /* parent closes the socket */
    exit(0);
```

Open a File

- **Open a file for read:**

```
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):**

```
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);
}
```

File Read

- Read from file

```
while ((rd_size = read(rfile, buf, 190)) > 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

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

■ Close files

```
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 cccwork[1-2].wpi.edu**

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

Questions?