CS4514 – B03
Project 2 Help Session

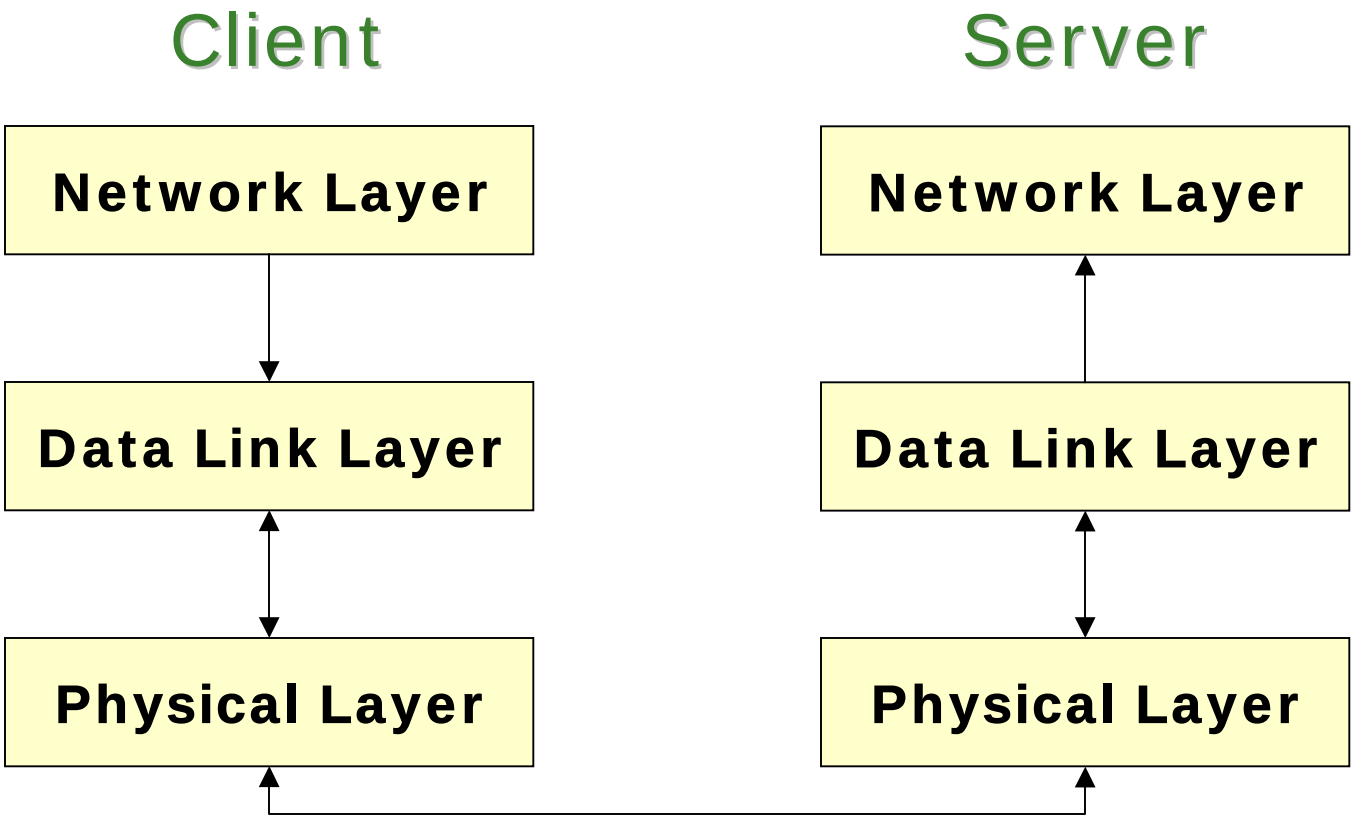
Choong-Soo Lee

November 24, 2003

Description

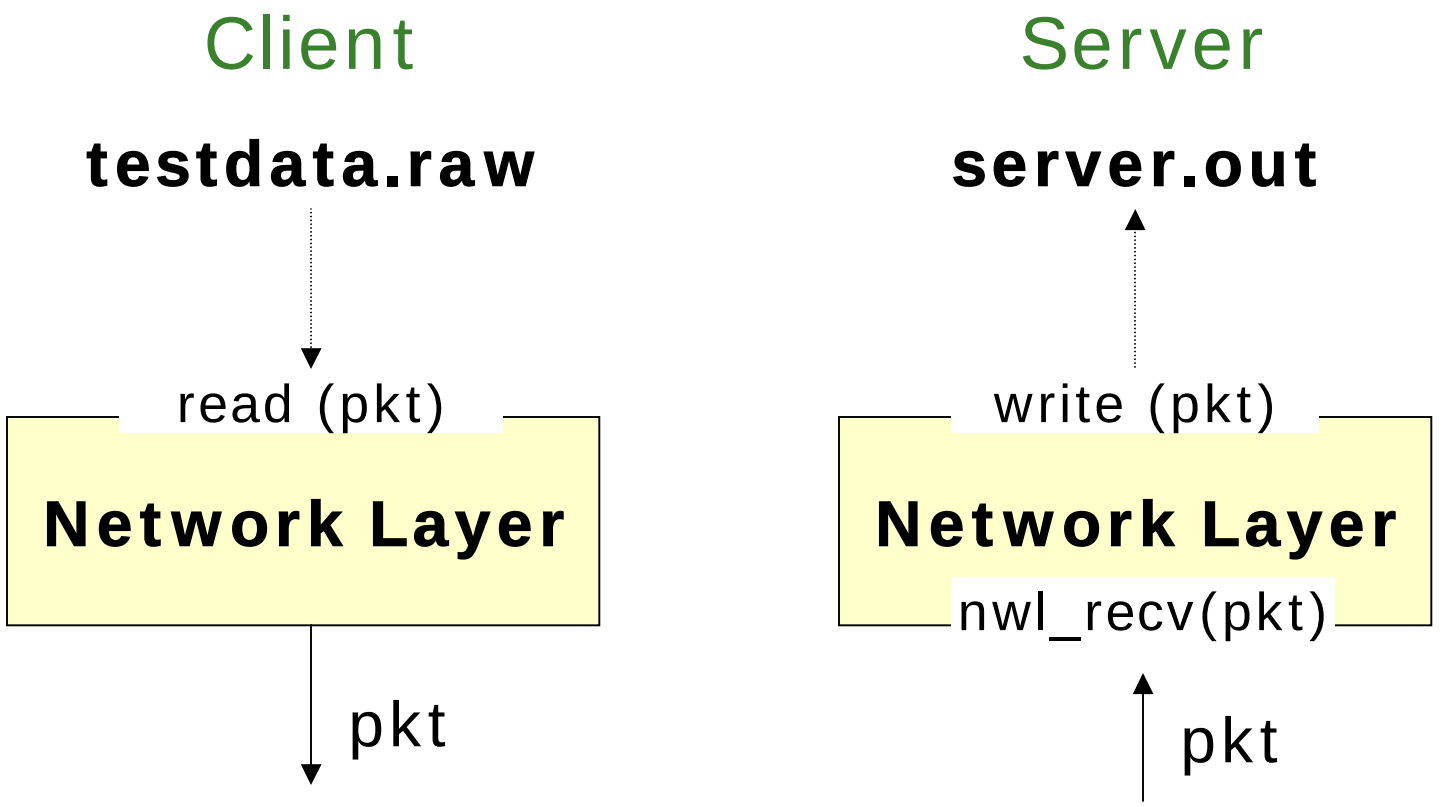
- The goal is to implement a Positive Acknowledgement with Retransmission (PAR) protocol on top of an emulated physical layer.
  - The receiver acknowledges only the correctly received segments and the sender uses timeout to detect and send the lost segment.
  - Physical layer is emulated by a TCP connection plus an error module.

Framework

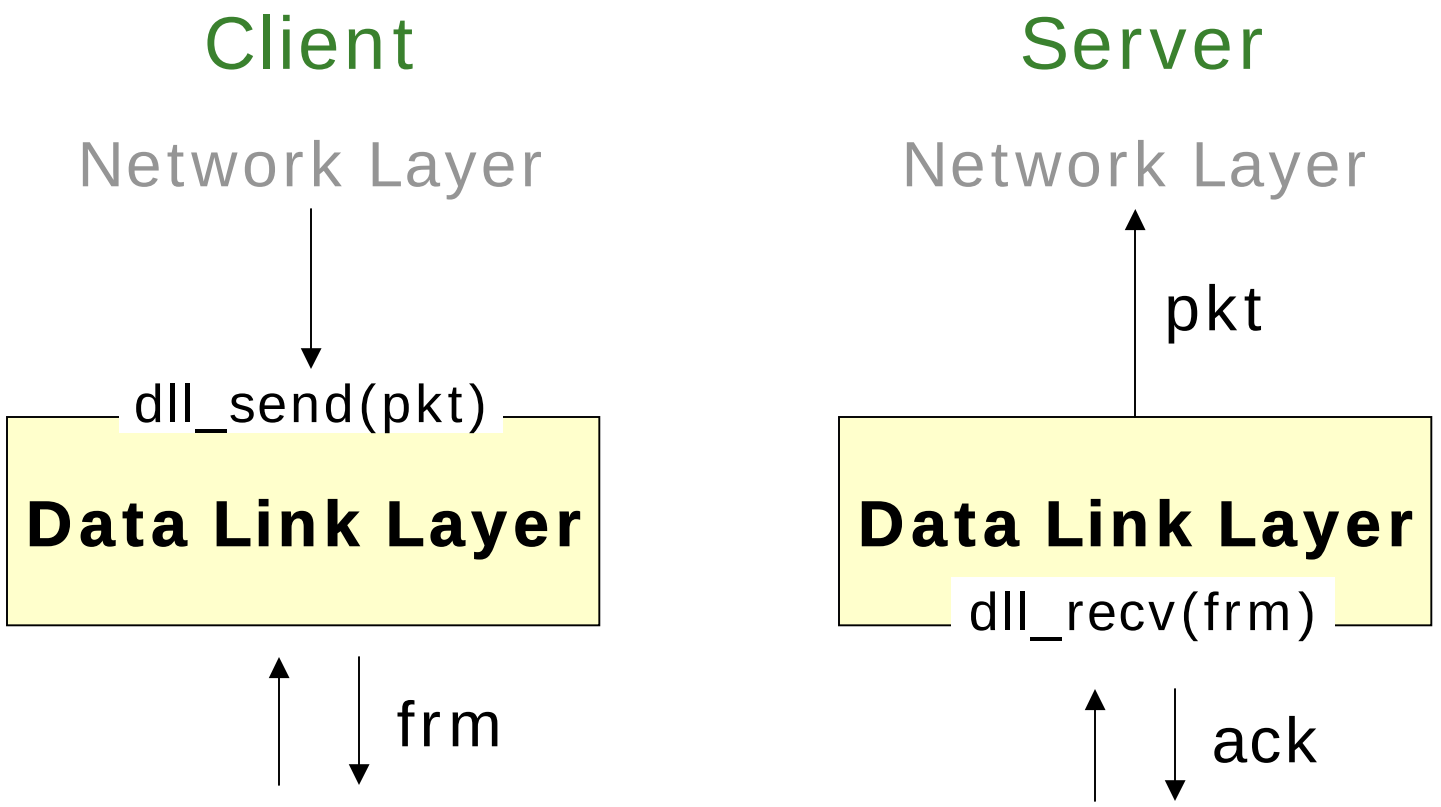


Do NOT attempt to put everything in one big main()

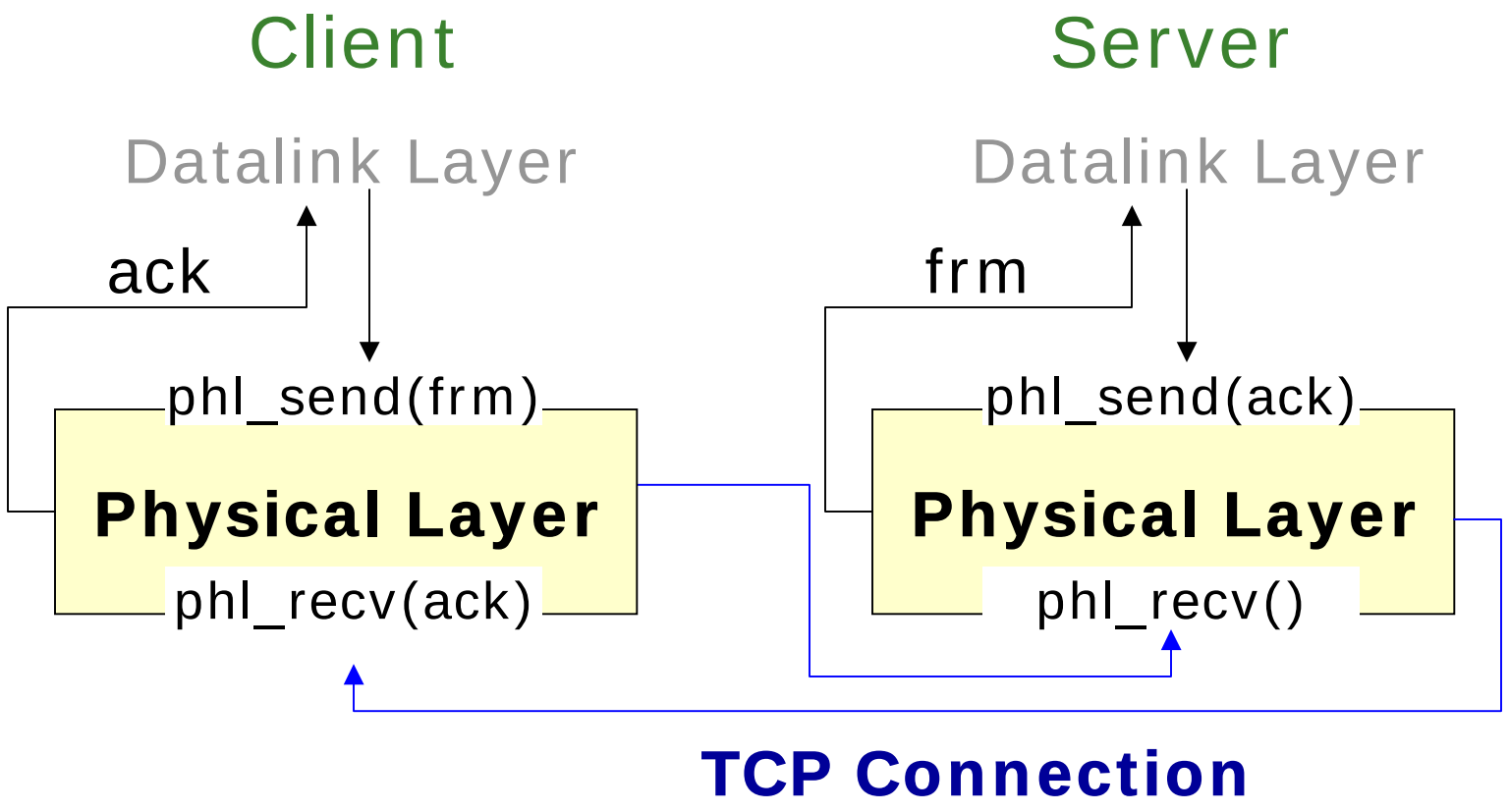
Network Layer



Data Link Layer



Physical Layer



Testdata File

- Pkt_num — the number of packets
- Packet_i_len — the byte number of the i-th packet
- Packet_i — the i-th packet in raw byte form

```
2                                          {one byte}
38                                         {one byte}
CS4514, computer network course, FL320     {38 bytes}
31                                         {one byte}
Worcester Polytechnic Institute            {31 bytes}
```

Example: Read testdata.raw

- /cs/cs4514/pub/example/getData.c

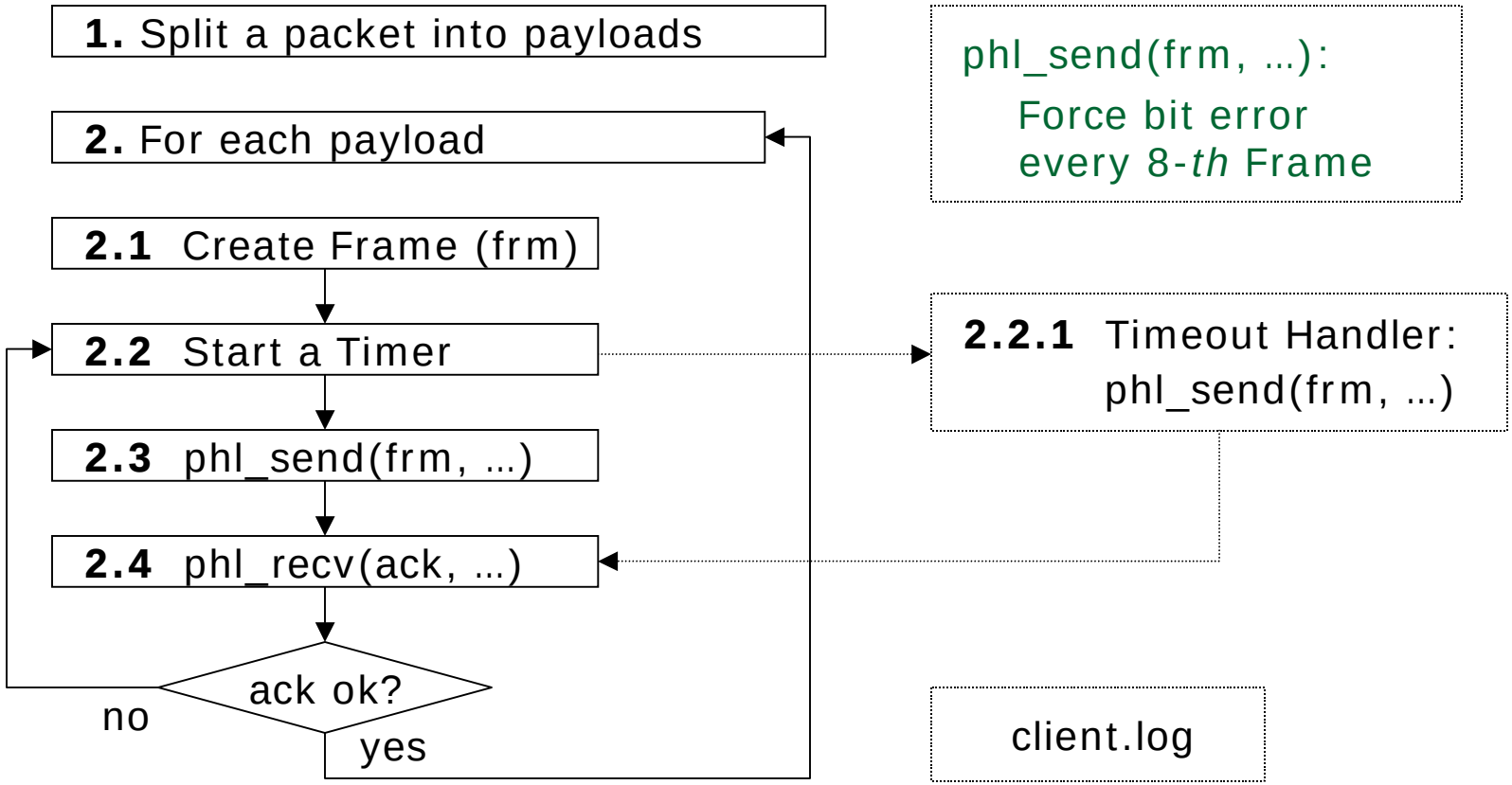
```
main(int argc, char **argv) {
    int fp, i;
    unsigned char packets[205];
    unsigned char byteNum;
    unsigned char p;

    if ((fp = open(argv[1], O_RDONLY)) < 0) {
        fprintf(stderr, "Open testData error!“);
        printf("Usage: %s filename\n", argv[0]);
        exit(-1);
    }
```

Example: Read testdata.raw (continued)

```
    read(fp, &p, 1);
    printf("The total number of packets is: %d\n\n", p);
    for (i = 0; i < p; i++) {
        read(fp,  &byteNum, 1);
        printf("The length of %dth packet : %d\n",
    i+1,byteNum);
        read(fp, packets, byteNum);
        packets[byteNum] = '\0';
        printf("The content of %dth packet : %s\n\n",
    i+1,packets);
    }
    close(fp);
}
```

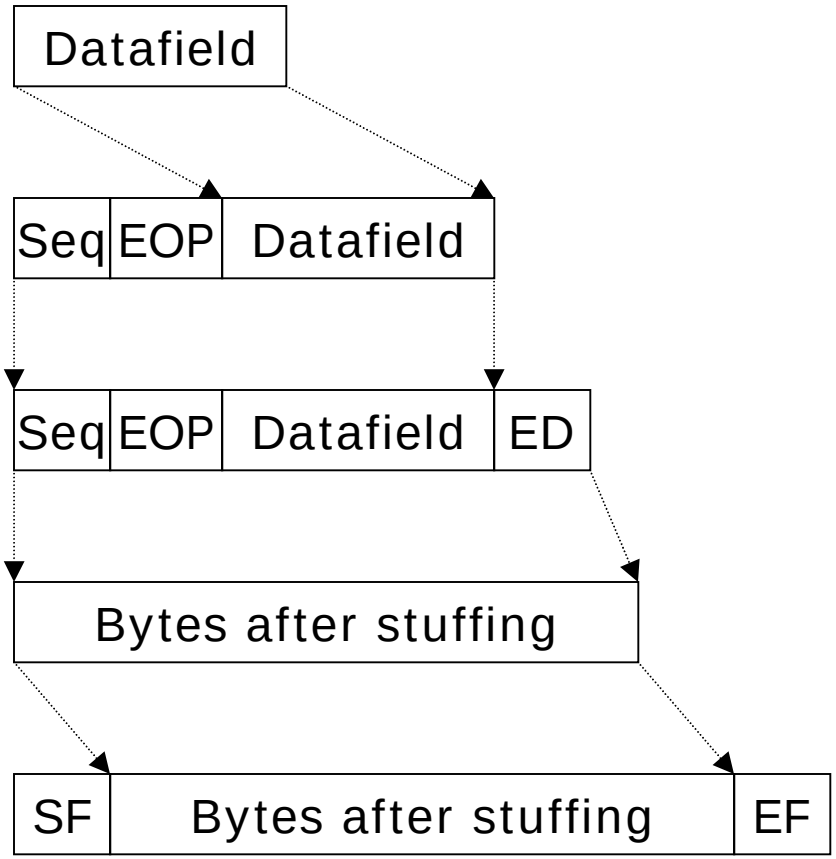
Raw file: /cs/cs4514/pub/C02_proj2/miniData.raw

Client: dll_send(pkt, …)



Create Frame

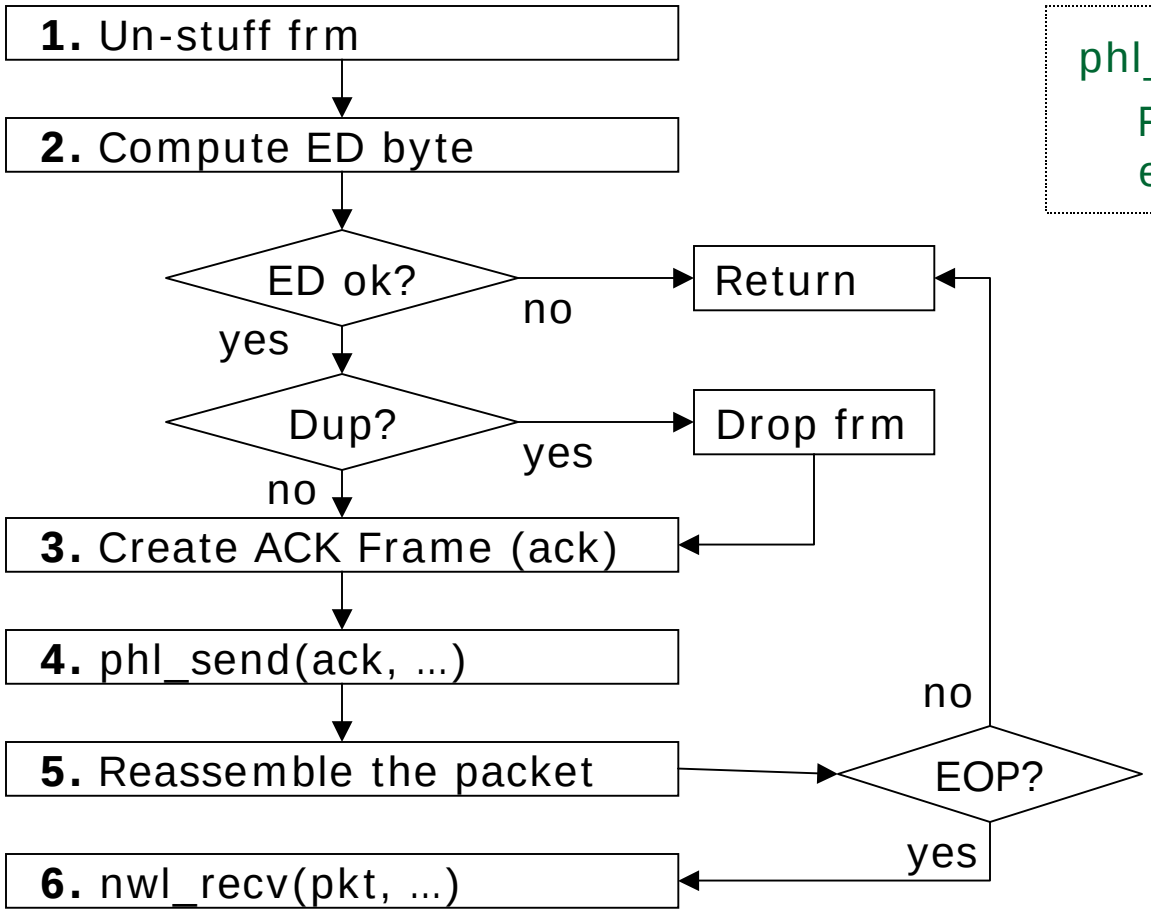
1. Compute Seq Number and End-Of-Packet (EOP) byte
2. Error-Detection (ED) byte (XOR on Seq + EOP + Data)
3. Byte Stuffing on Seq + EOP + Data + ED
4. Add Start-Flag (SF) and End-Flag (EF)



EOP: End of Packet
SF: Start-flag

ED: Error Detection
EF: End-flag

Server: dll_recv(frm, ...)



phl_send(frm, …):
Force bit error
every 7-*th* Frame

server.log

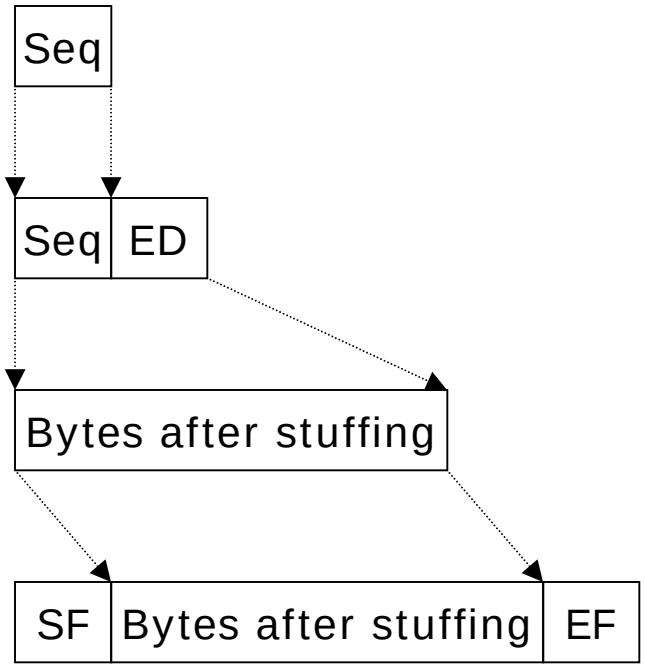
Create ACK Frame

1. Compute Seq Number

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

3. Byte Stuffing on Seq+ED

4. Add Start-Flag (SF) and
   End-Flag (EF)



EOP: End of Packet
SF: Start-flag

ED: Error Detection
EF: End-flag

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 *writeset,
            fd_set *exceptset, const struct timeval *timeout);

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

Example: Select

```
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

```
#include <sys/signal.h>
#include <sys/time.h>
#include <sys/timers.h>
```

- Register a function to TIMEOUT signal

```
signal (SIGALRM, timeout);
```

- Create a timer and begin to run

```
timer_create();
timer_settime();
```

Example: Signal and Timer

```
timer_t timer_id;

void timeout(){
        printf("\n Time out!!!!\n");
        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);
}
```