CS4514 Assignment 2 Help Session
– Data Link Layer Client and Server Processes

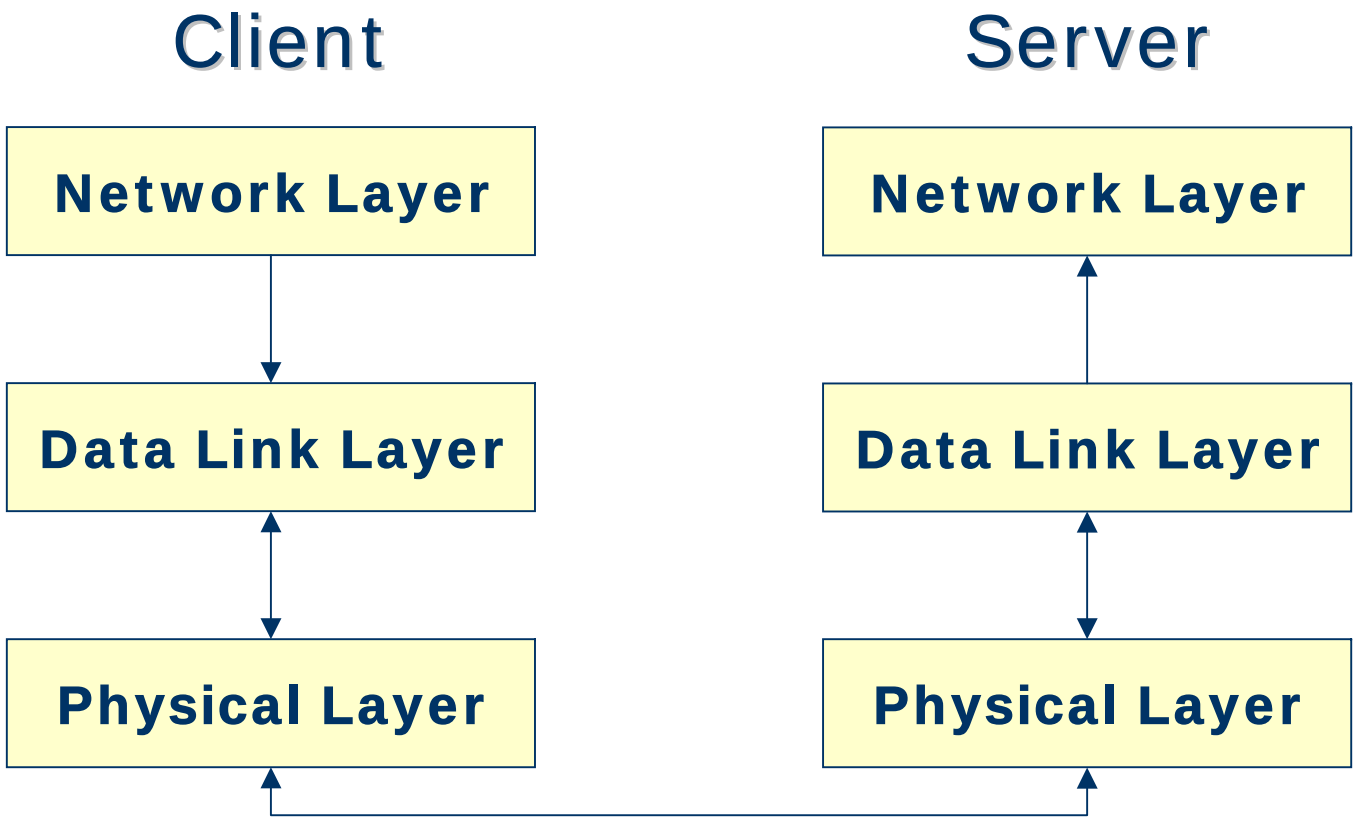
Speaker: Hao Shang

Date: Nov. 11th, 2004

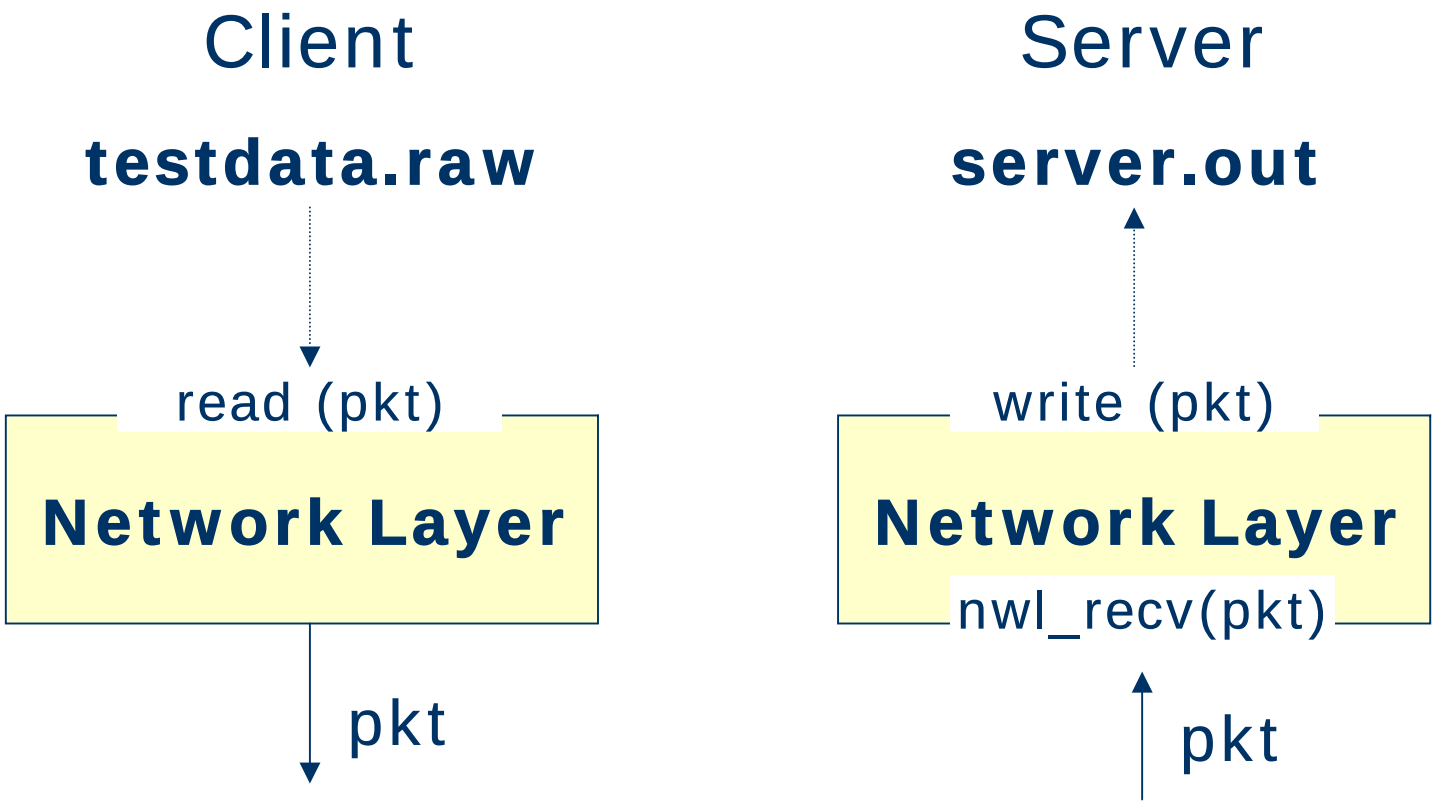
Goal

- Implement data link layer based on PAR protocol (Positive Acknowledgement with Retransmission)
  - PAR: a working stop-and-wait protocol over un unreliable channel. Ref to Figure3-12 in textbook.
  - Error detection: 1 byte xor
  - Framing: 80bytes max payload, flag bytes [0x7E] with byte stuffing
- Physical layer is emulated by a TCP connection plus an error module.

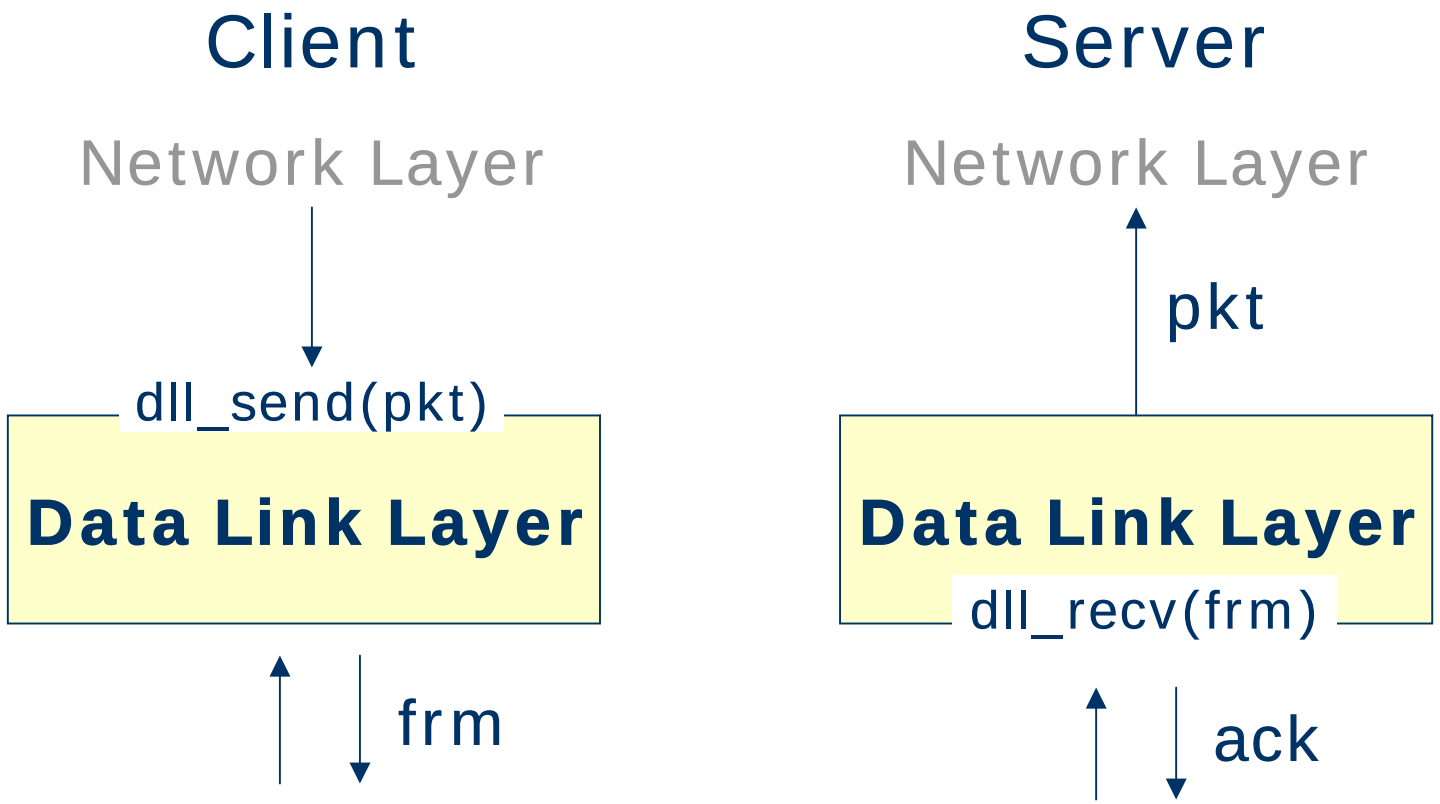
Framework


Client
Server
Network Layer
Network Layer
Data Link Layer
Data Link Layer
Physical Layer
Physical Layer

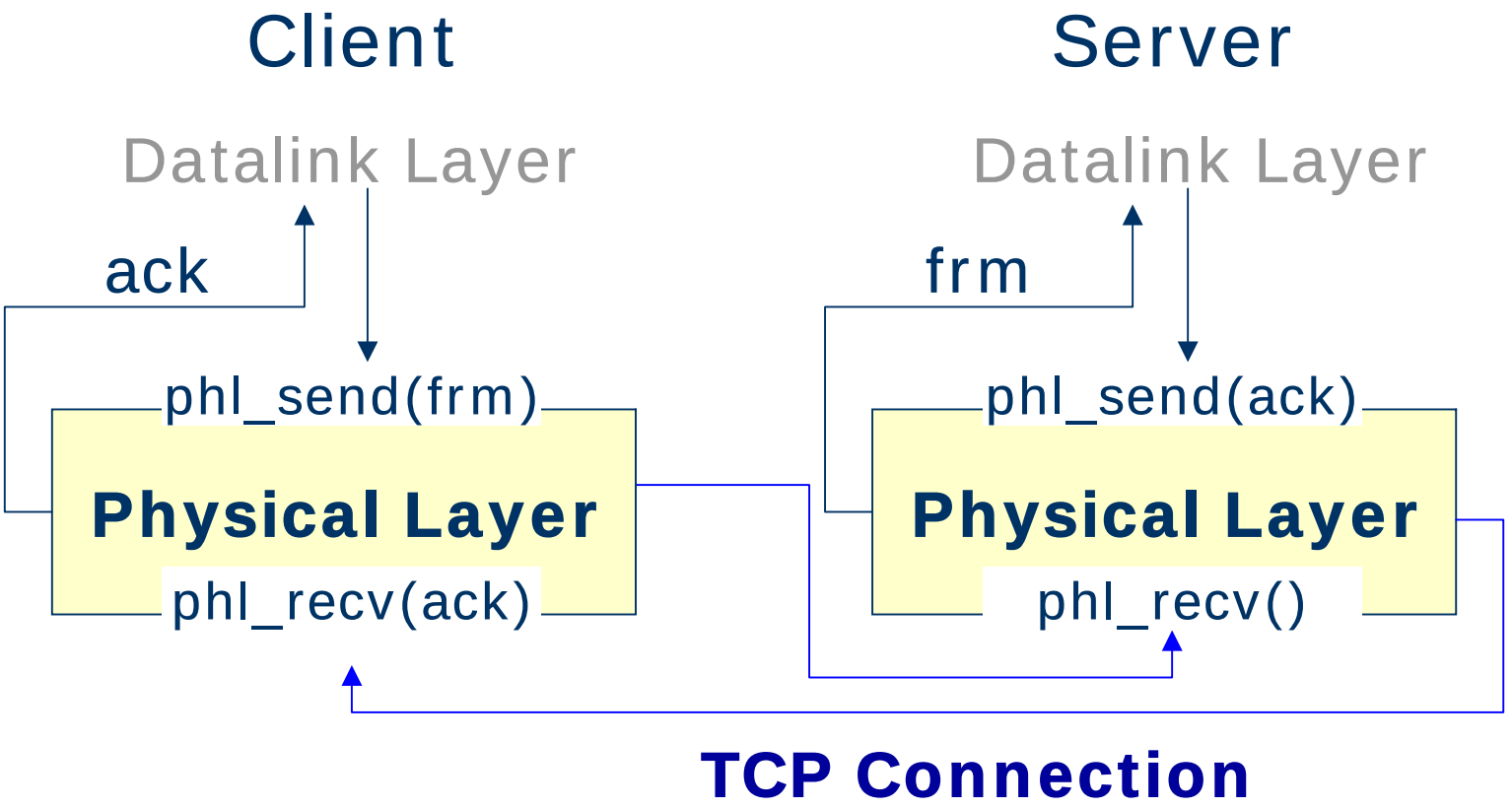
Network Layer



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_simple.raw

- ccc:/cs/cs4514/pub/proj2test/getData.c

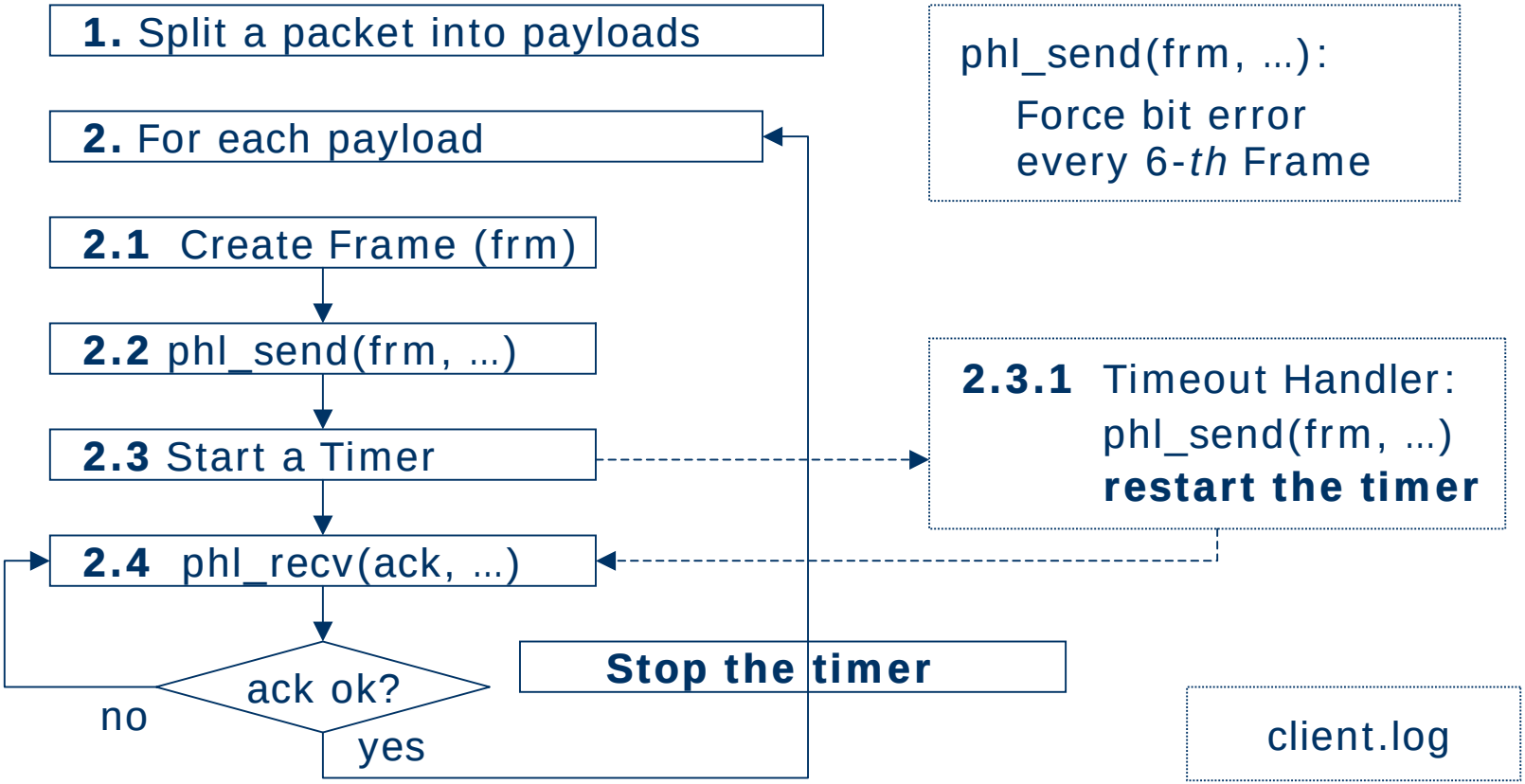
```
main(int argc, char **argv) {
    int fp, i;
    unsigned char packets[221];
    unsigned char byteNum;
    unsigned char p;

    if ((fp = open(argv[1], O_RDONLY)) < 0) {
        fprintf(stderr, "Open testData error!“);
        printf("Usage: %s filename\n", argv[0]);
        exit(-1);
    }
    read(fp, &p, 1);
    printf("The total number of packets is: %d\n\n", p);
```

Example: Read testdata_simple.raw (continued)

```
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);
        //the following two lines are applicable for printable
packets only
        packets[byteNum] = '\0';
        printf("The content of %dth packet : %s\n\n",
i+1,packets);
}
close(fp);
}
```

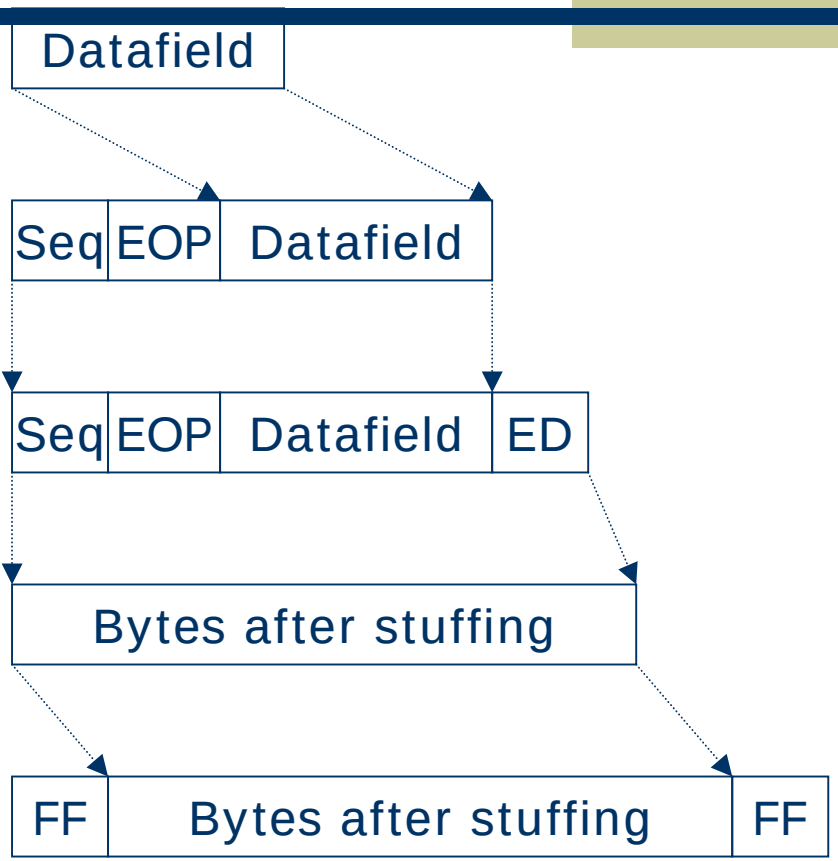
- Raw file: ccc:/cs/cs4514/pub/proj2test/testData_simple.raw
- Another example: ccc:/cs/cs4514/pub/proj2test/getData_v1.c
- NOTE: Packets may not be printable characters

Client: dll_send(pkt, ...)



Create Frame

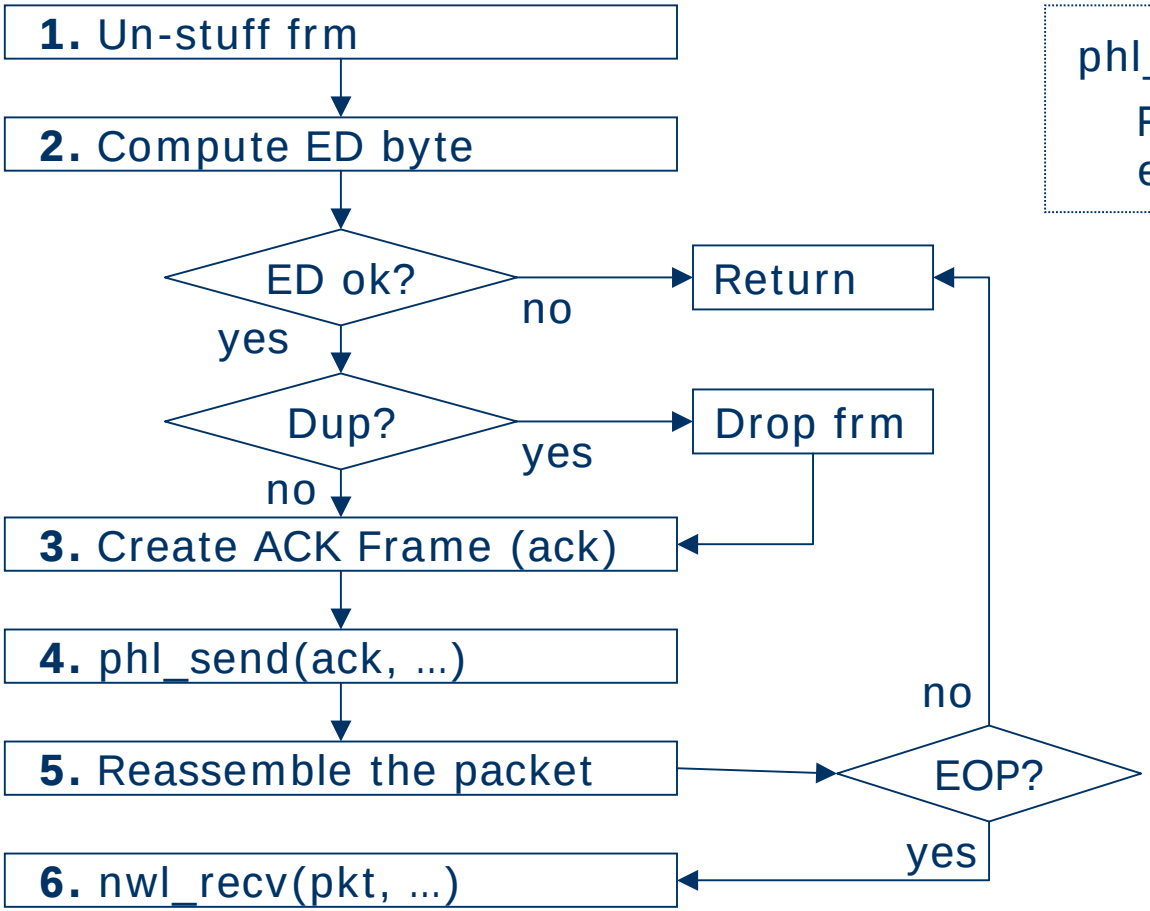
1. Compute Seq Number and End-Of-Packet (EOP) byte
2. Error-Detection (ED) byte (XOR on Seq + EOP + Data)
3. Byte Stuffing on Seq + EOP + Data + ED
4. Add Frame-Flag (FF) 0x7E at both ends



EOP: End of Packet
FF: frame flag

ED: Error Detection

Server: dll_recv(frm, …)



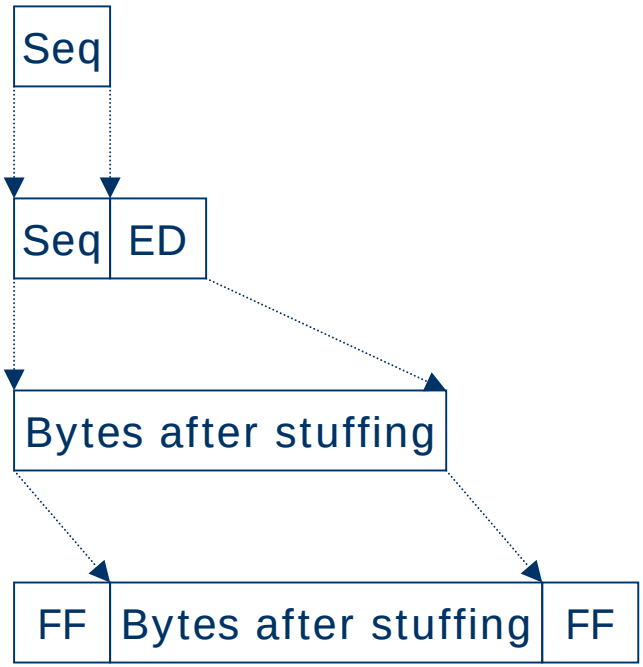
phl_send(frm, …):

Force bit error every 5-*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 Frame-Flag (FF) 0x7E at both ends



EOP: End of Packet
FF: Frame Flag

ED: Error Detection

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 : block
  - Signal and Timer: non-block

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 <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;
/* set both fields of it_value 0 to
    disarm the alarm */
    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);
}
```

Questions