CS4514-Help Session 2

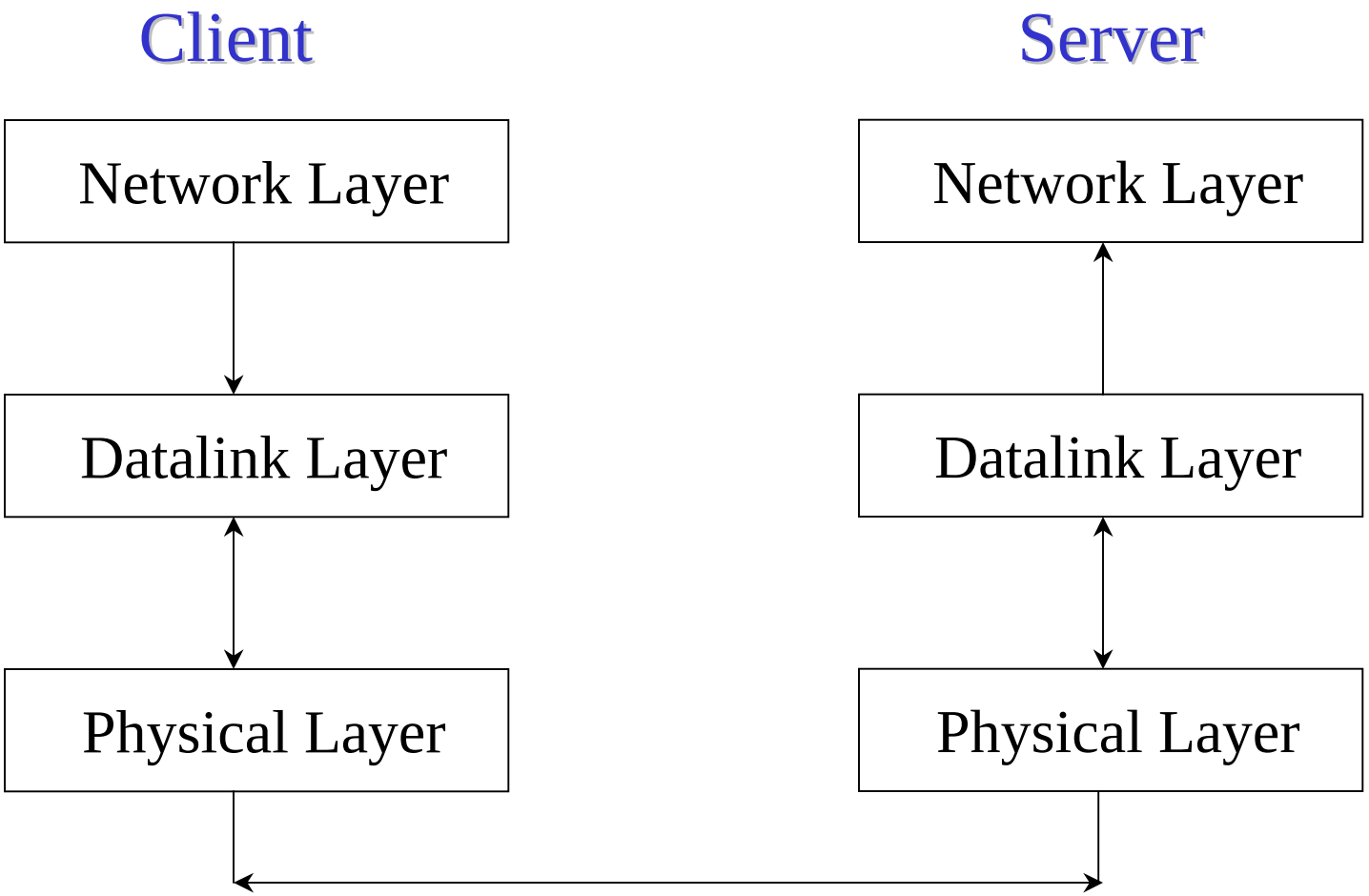
---- By Huahui Wu

Description

- In this project, you are supposed to implement a PAR (Positive Acknowledgement with Retransmission) Protocol on top of an emulated physical layer (It is actually the TCP here)

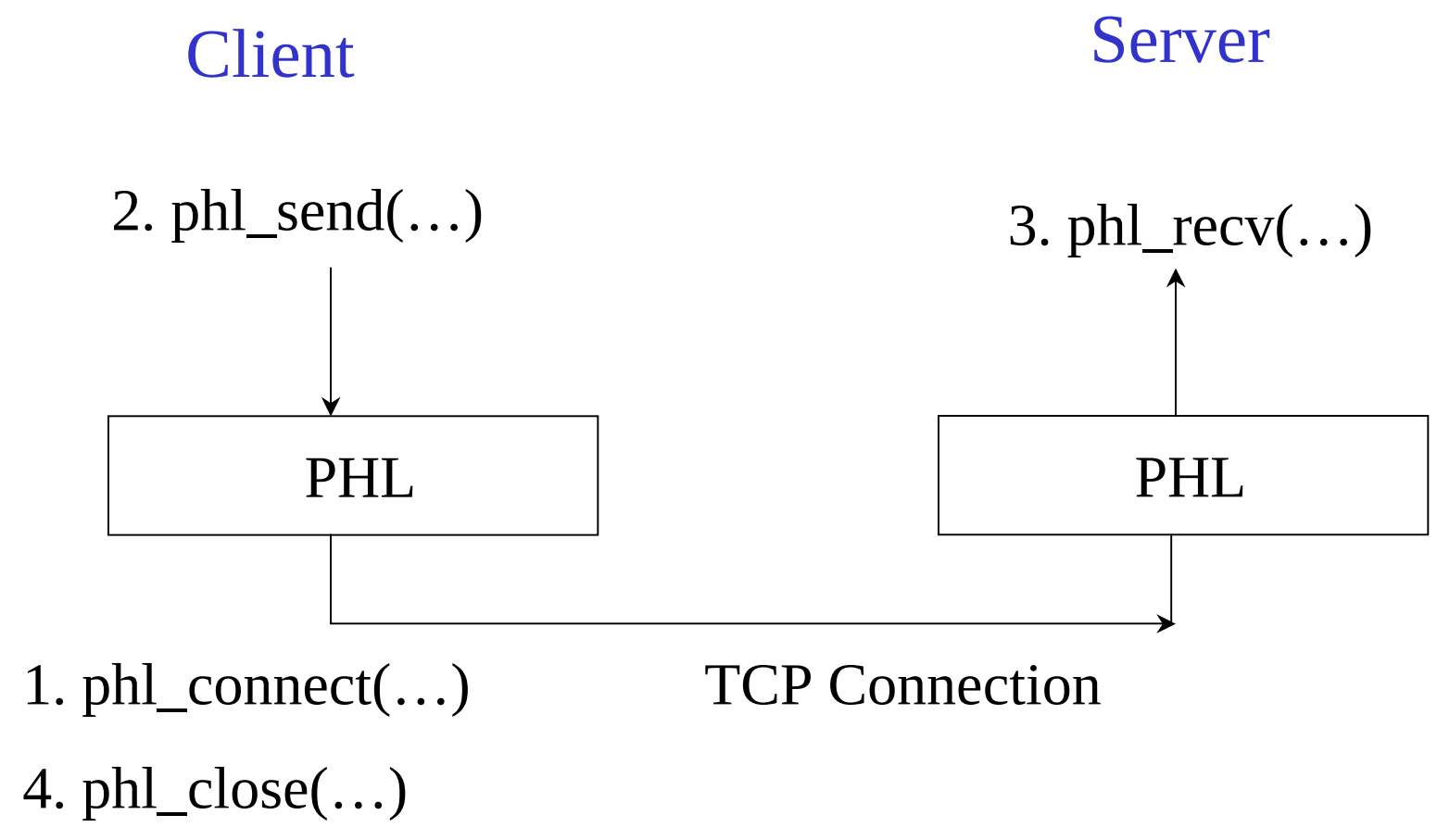
Framework

Don’t put everything in one Super main()



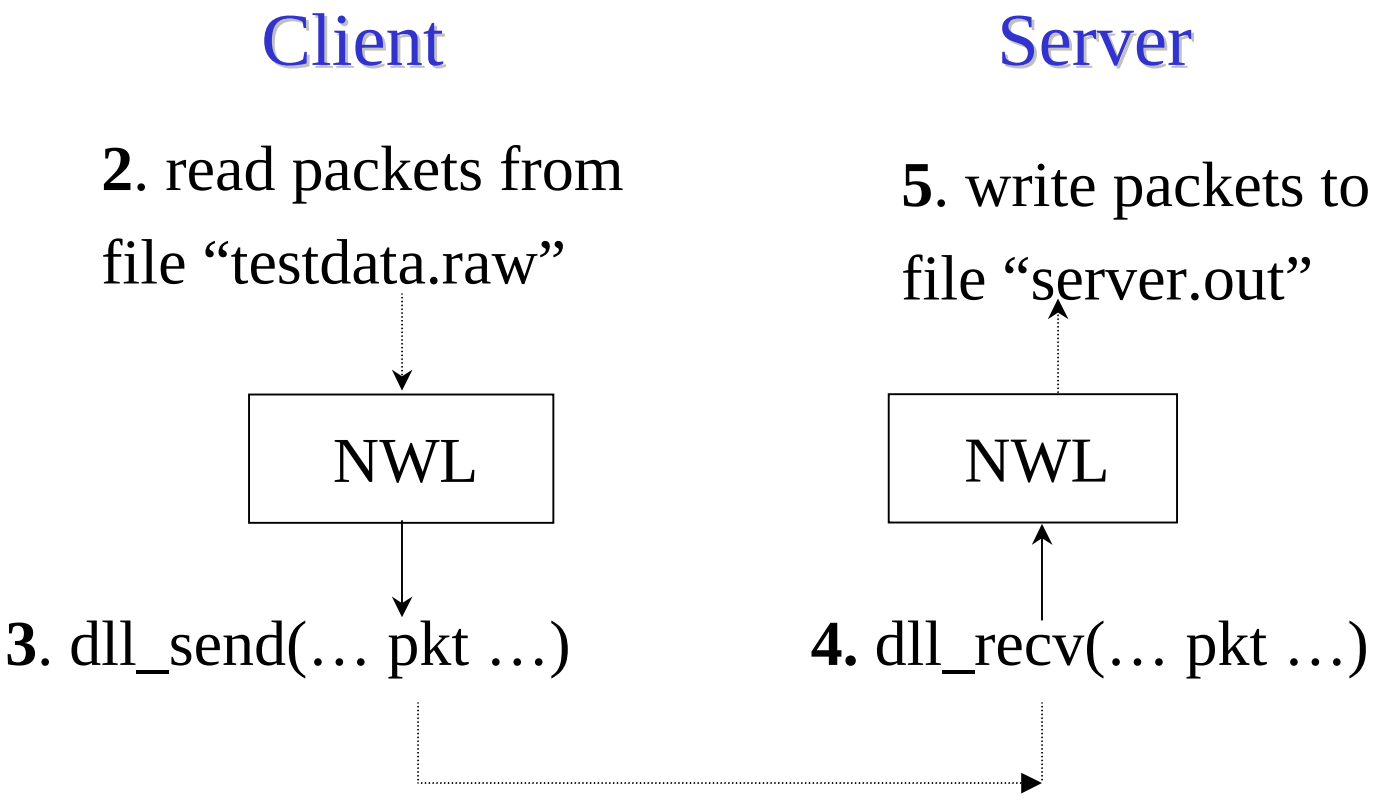
Threads?

Physical Layer



NOTE: *Real TCP does the actual transmission for your PHL.*

Network 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

Worcester Polytechnic Institute

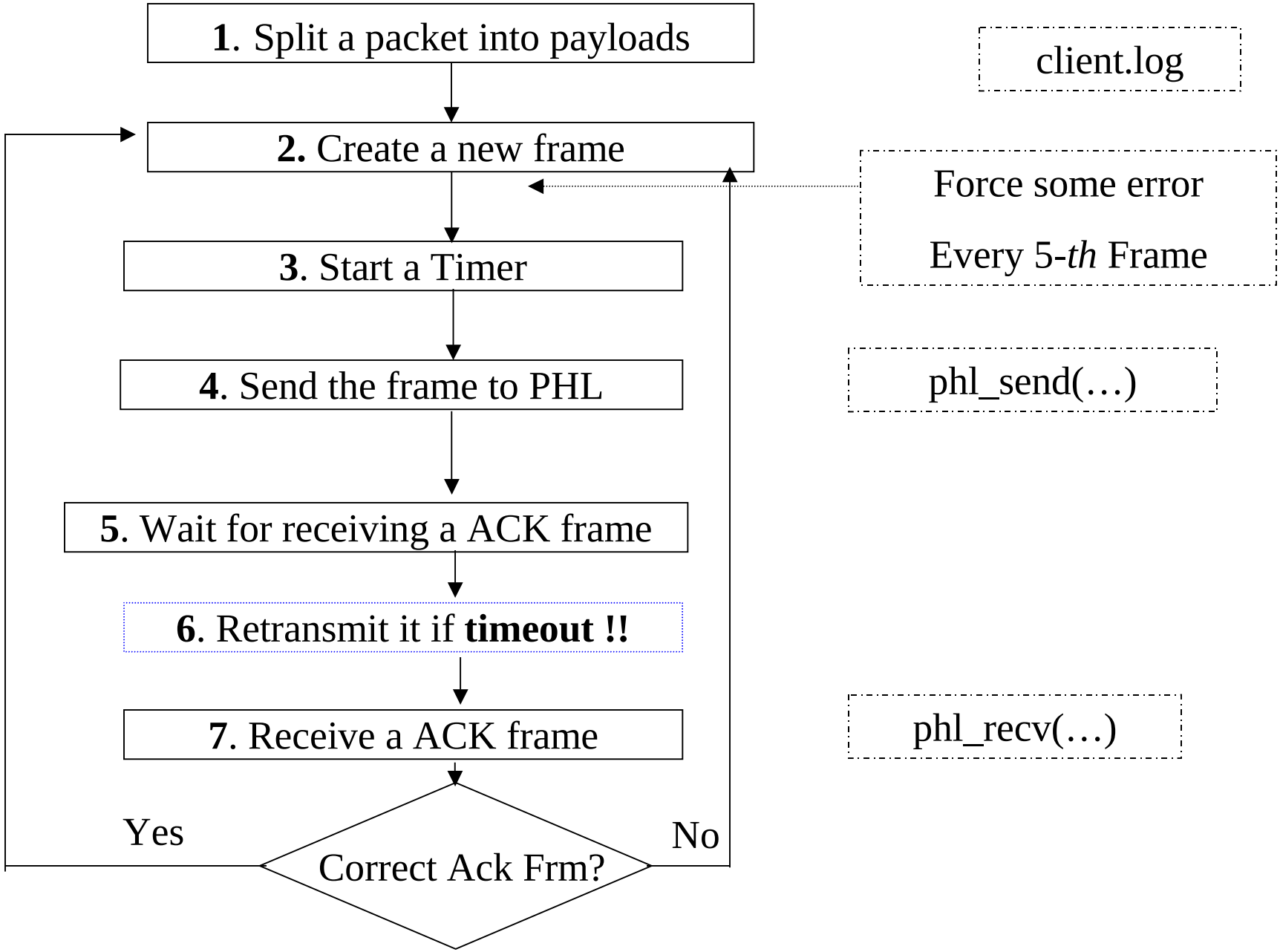
EX1: Read testdata.raw

```
#include <fcntl.h>
#include <stdio.h>
main(int argc, char* argv[]){
    int fp, i;
    unsigned char packets[205];
    unsigned char byteNum;
    unsigned char p;
    char
    if ((fp = open(argv[1], O_RDONLY)) < 0) {
        fprintf(stderr, "Open testData file error!");
        printf("Usage: %s filename\n", argv[0]);
        exit(-1);
    }
```

EX1: Read testdata.raw (Cont)

```
    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 a new frame

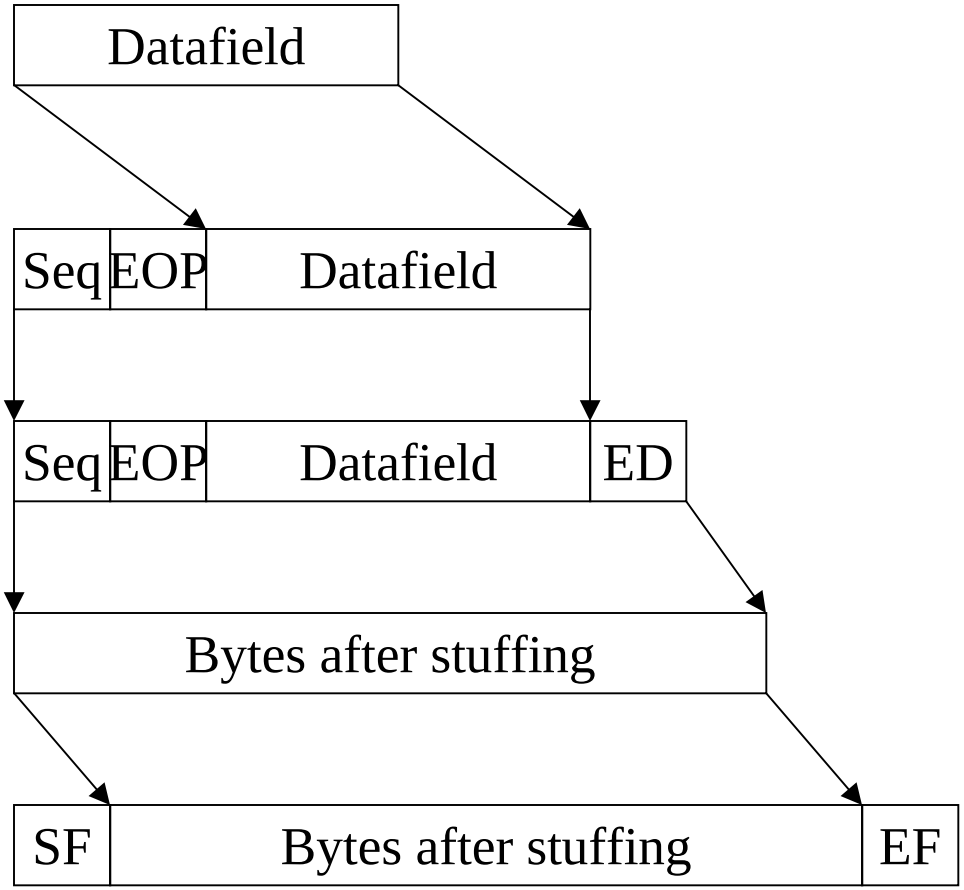
1. Generate the payload from the packet

2.compute sequence number and end-of-packet byte

3. Compute Error-Detection byte (Using XOR on Seq+EOP+Data)

4. Byte Stuffing on Seq+EOP+ED+Data

5. Add Start-flag&End-flag



- EOP: End of Packet
- SF: Start-flag

ED: Error Detection

EF: End-flag

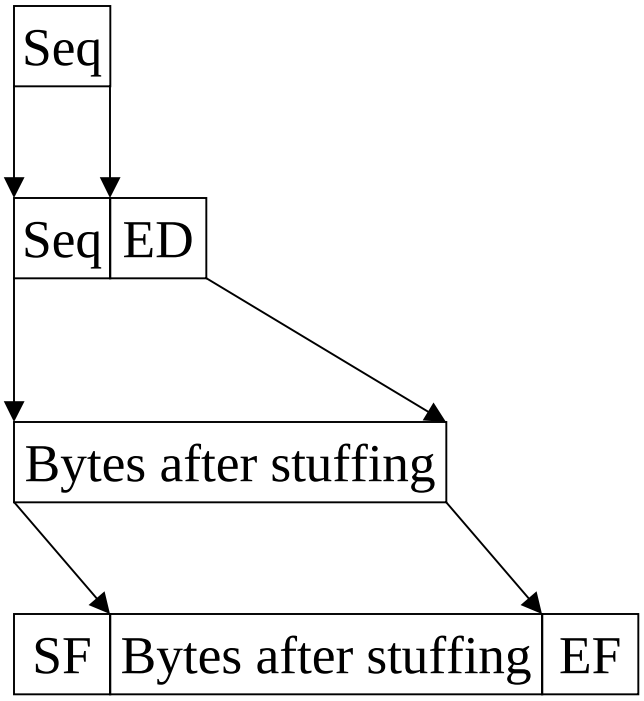
Creating a new Ack frame

1.Compute sequence number

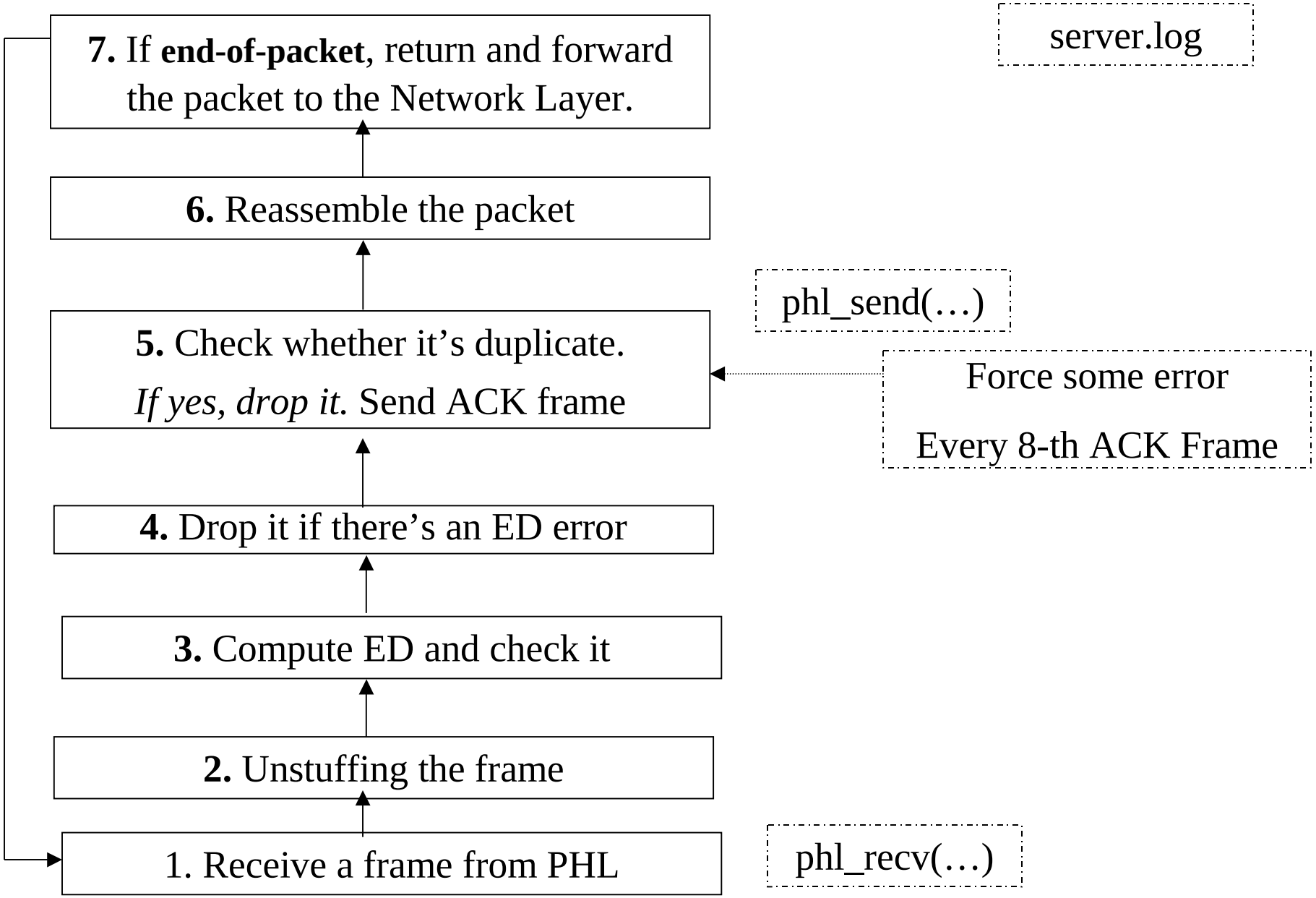
2. Compute Error-Detection byte by XOR (ED=Seq)

3. Byte Stuffing on Seq+ED

4. Add Start-flag&End-flag



Server dll_recv(…pkt…)



Timer

- The client can’t wait the server’s response forever. It will retransmit when timeout.
- Two kinds of timer can be used in proj2
  - 1. select(): Easy to use
  - 2. signal: need to use in proj3, maybe you want start using it now.

System call: select()

In this assignment, you can call select and tell the kernel to return only when:

- the socket descriptor in physical layer is ready for reading, or
- a preset timer timeout

System call: select()

```
# 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

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

EX2: how to use select()

```
{
 fd_set bvfdRead;                    FD_SET(sockfd, &bvfdRead);
 int readyNo;                         readyNo = select(sockfd+1,
 struct timeval timeout;                 &bvfdRead, 0, 0, &timeout);
 int sockfd;                         if(readyNo < 0)  //some errors
                                           error_handler;
 while (true) {                      else if(readyNo = = 0)  // timeout
   timeout.tv_sec = 0;                     timeout_handler;
   timeout.tv_usec = 500;            else   // receive data from socket
   FD_ZERO(&bvfdRead);                     receive_handler;  } }
```

Time signal

- 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();

EX3: How to use Time Signal

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/signal.h>
#include <sys/time.h>
#include <sys/timers.h>
timer_t timer_id;
void timeout(){
        printf("\n Time out!!!!\n");
        exit(0);        }

void start_timer(){
     struct itimerspec time_value;
     signal (SIGALRM, timeout);
     timer_create(
             CLOCK_REALTIME,
             NULL, &timer_id);
//set timeout to 1 second
     time_value.it_value.tv_sec     = 1;
     time_value.it_value.tv_nsec    = 0;
     time_value.it_interval.tv_sec  = 0;
     time_value.it_interval.tv_nsec = 0;
     timer_settime(
             timer_id, 0,
             &time_value, NULL          );
                     }

main(){
     start_timer();
     for(;;);           }
```