CS4514 HELP Session 3

Concurrent Server Using Go-Back-N

Feng Li
lif@cs.wpi.edu
12/01/2004

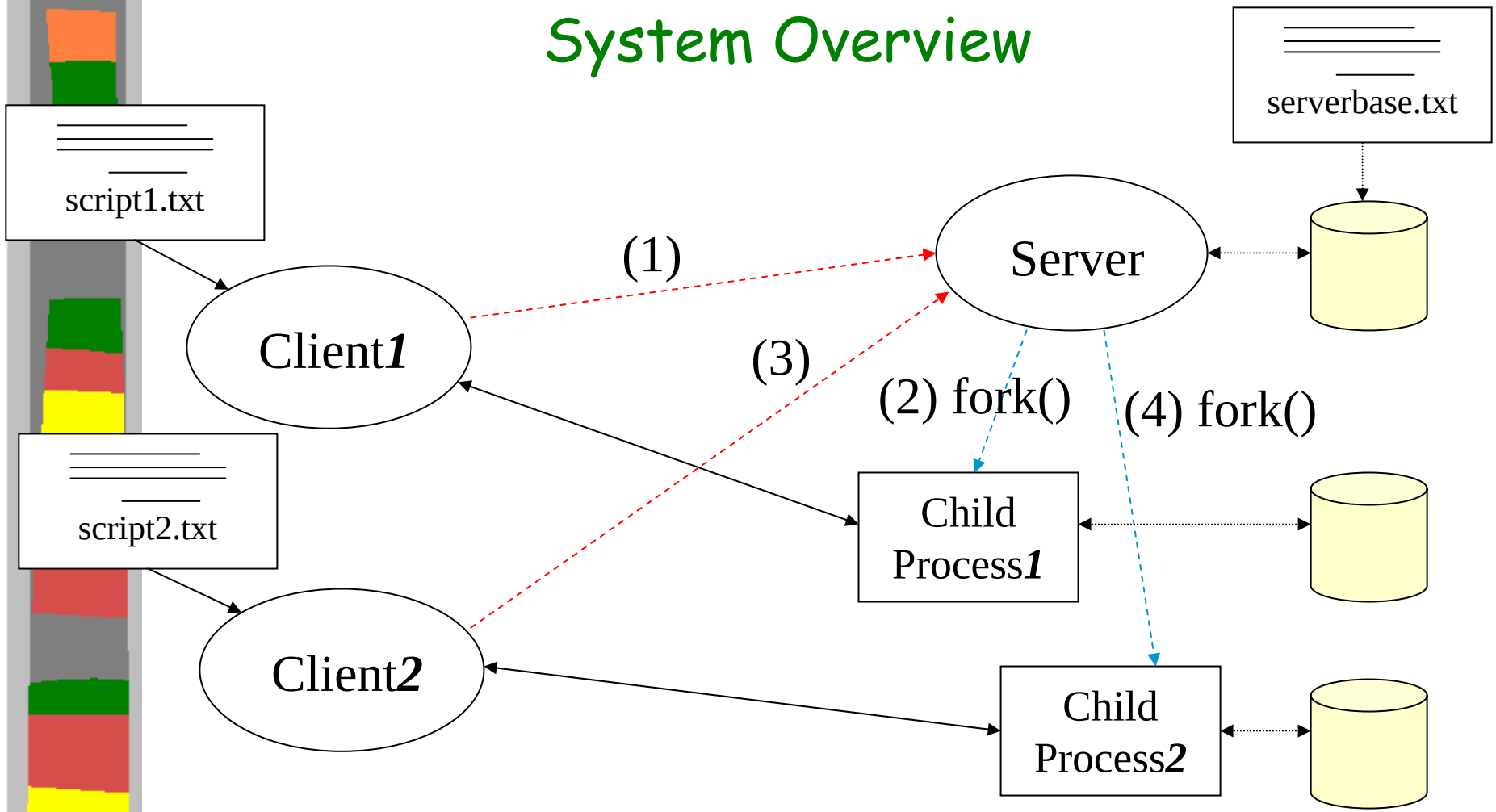
WPI
WORCESTER POLYTECHNIC INSTITUTE

Description

- Objective:

  To implement a simple concurrent server and clients having four emulated network protocol stacks.
  - Application layer: Read and execute commands
  - Network layer: Message←→Packet (send&recv)
  - Datalink layer: Packet ←→ Frame and Go-Back-N sliding window protocol
  - Physical layer: TCP connection.
- Your programs should compile and work on any host of ccc.WPI.EDU.

System Overview

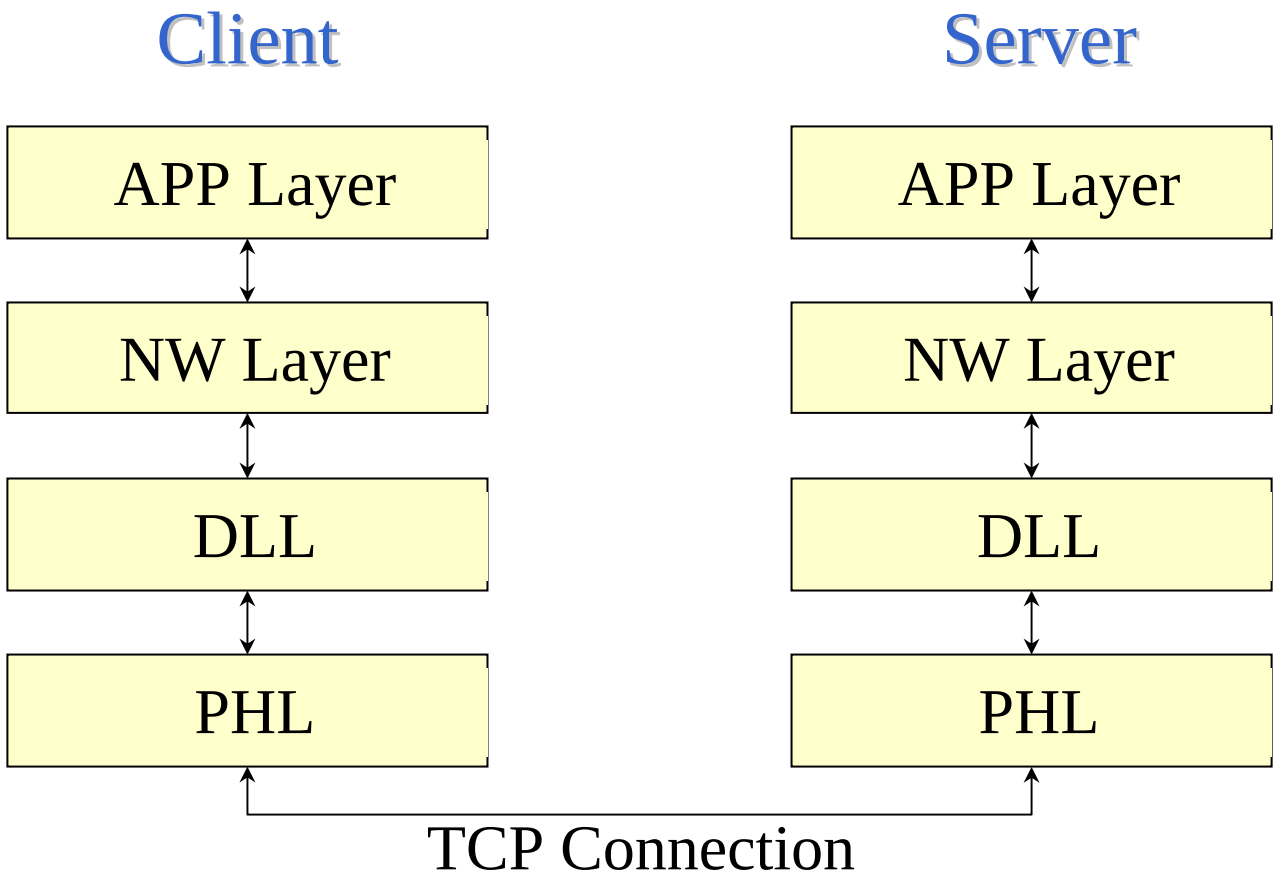


Note: each child process keeps a separate copy of the DB.

we do not keep data consistency for the serverbase

This is automatically done by using fork()

System Framework

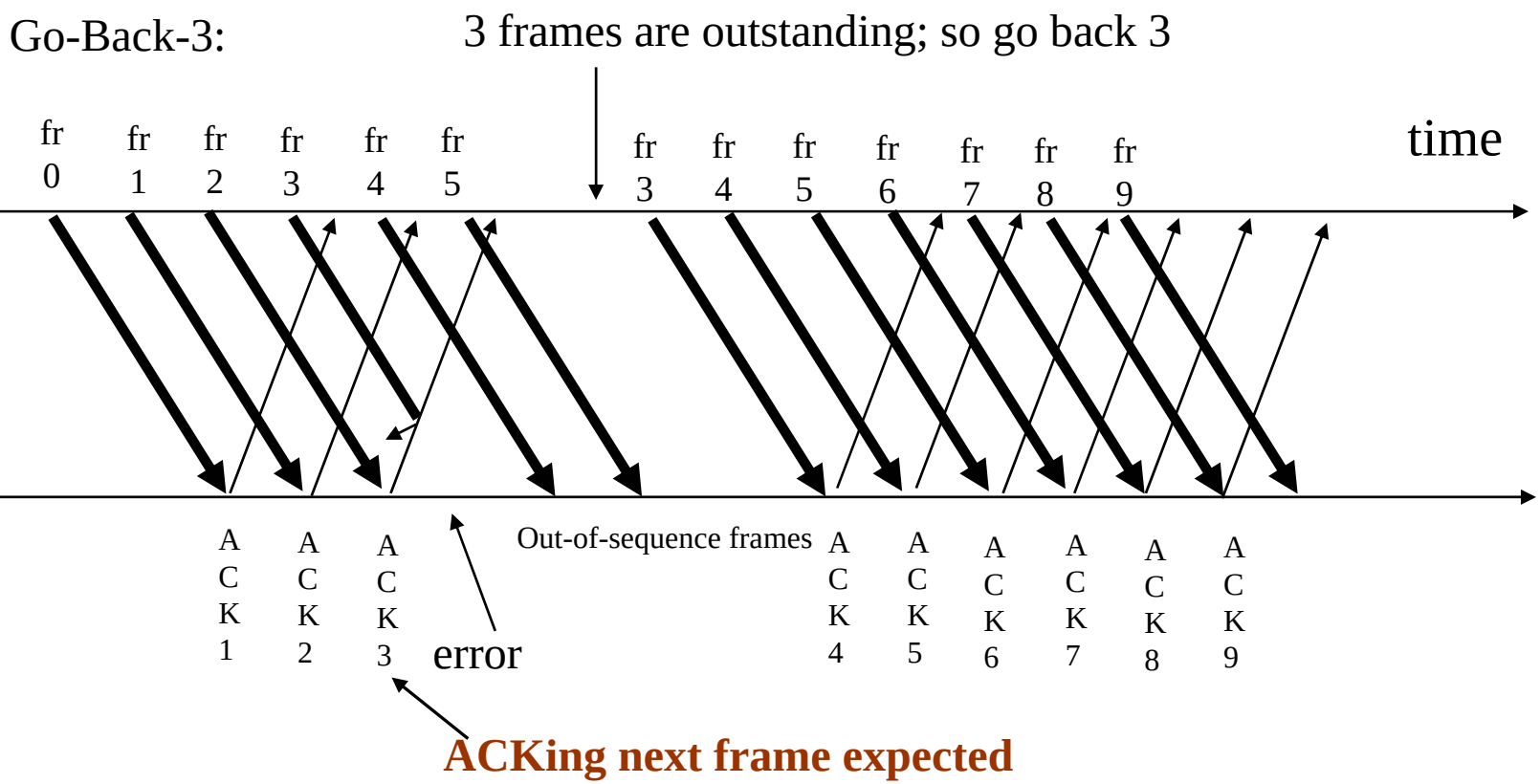


Four Layer stacks

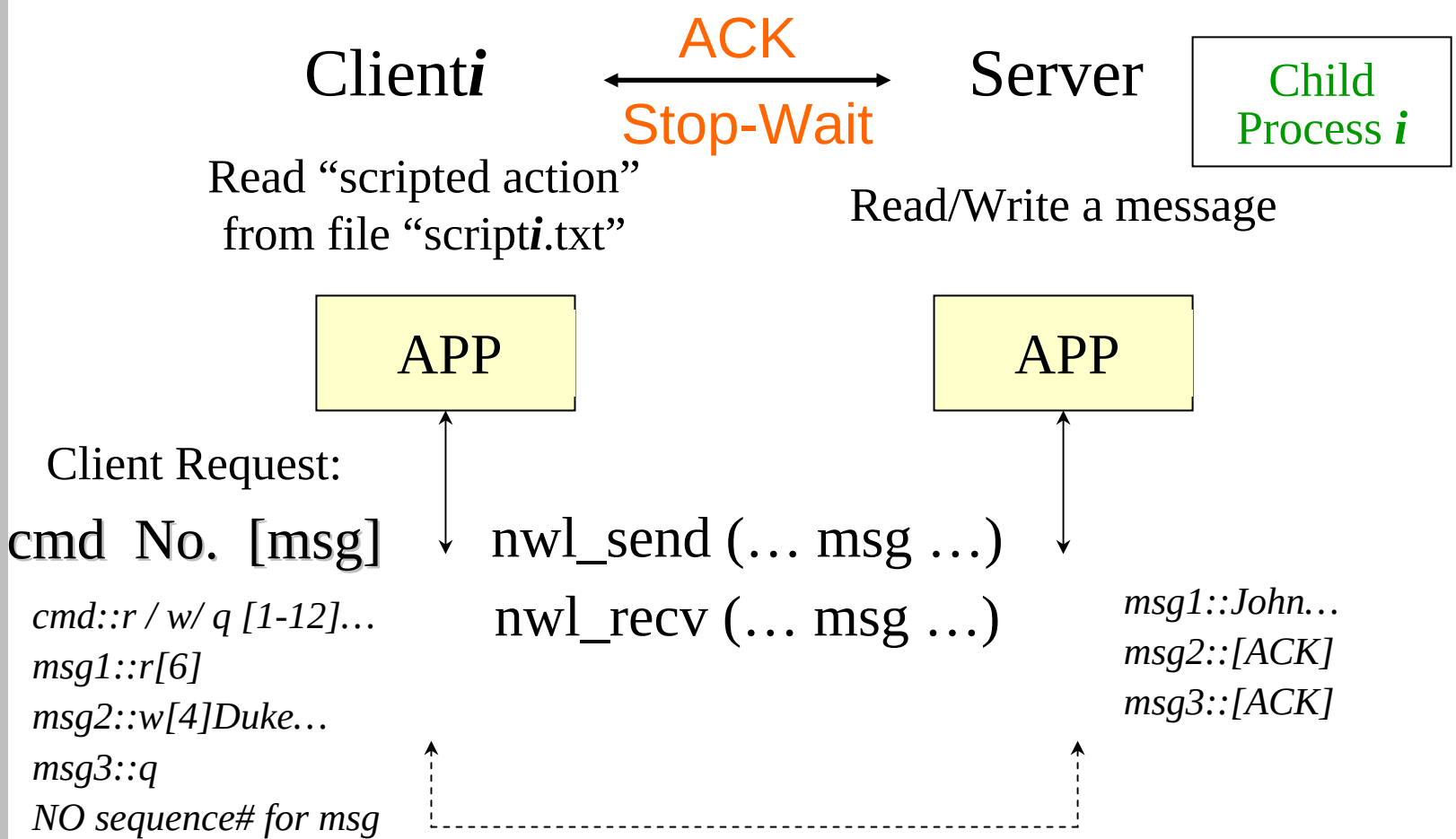
Concurrent Server (fork())

- fork() will make a child process with memory copy.
  - The initial serverbase will be copied to each child process.
  - fork() will return child pid in parent process and 0 in child process.
  - Remember to close socket after using.
- More information could be found at fork(2).

Go Back N



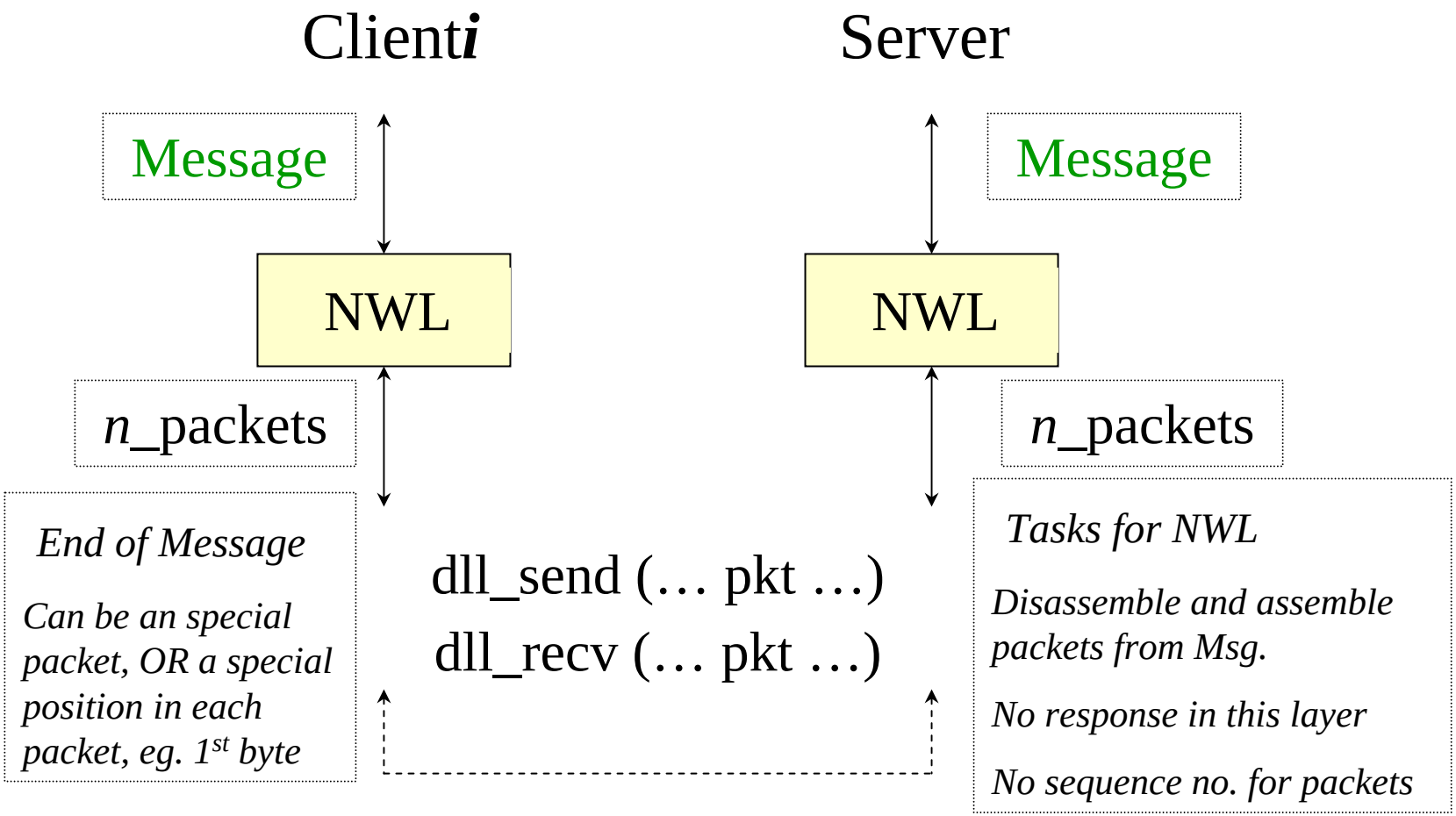
How the System Works: Layer by Layer
Application Layer



Note: The max_size of a message is 270 bytes

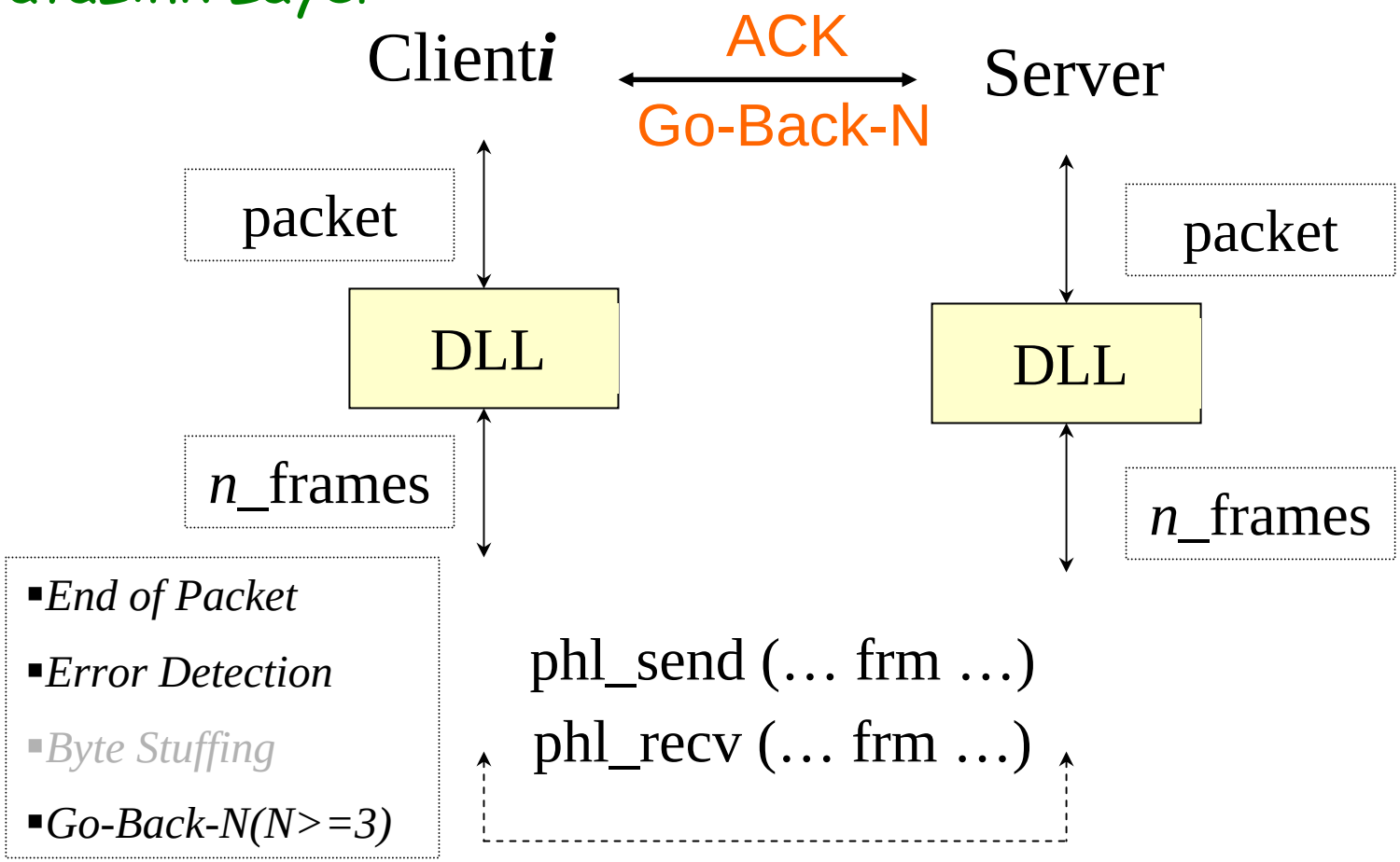
The number referring to triple position is 1 to 12

How the System Works: Layer by Layer
Network Layer



Note: The max_size of a packet is 90 bytes, The network layer will send packets until blocked by the Data Link Layer. But HOW?

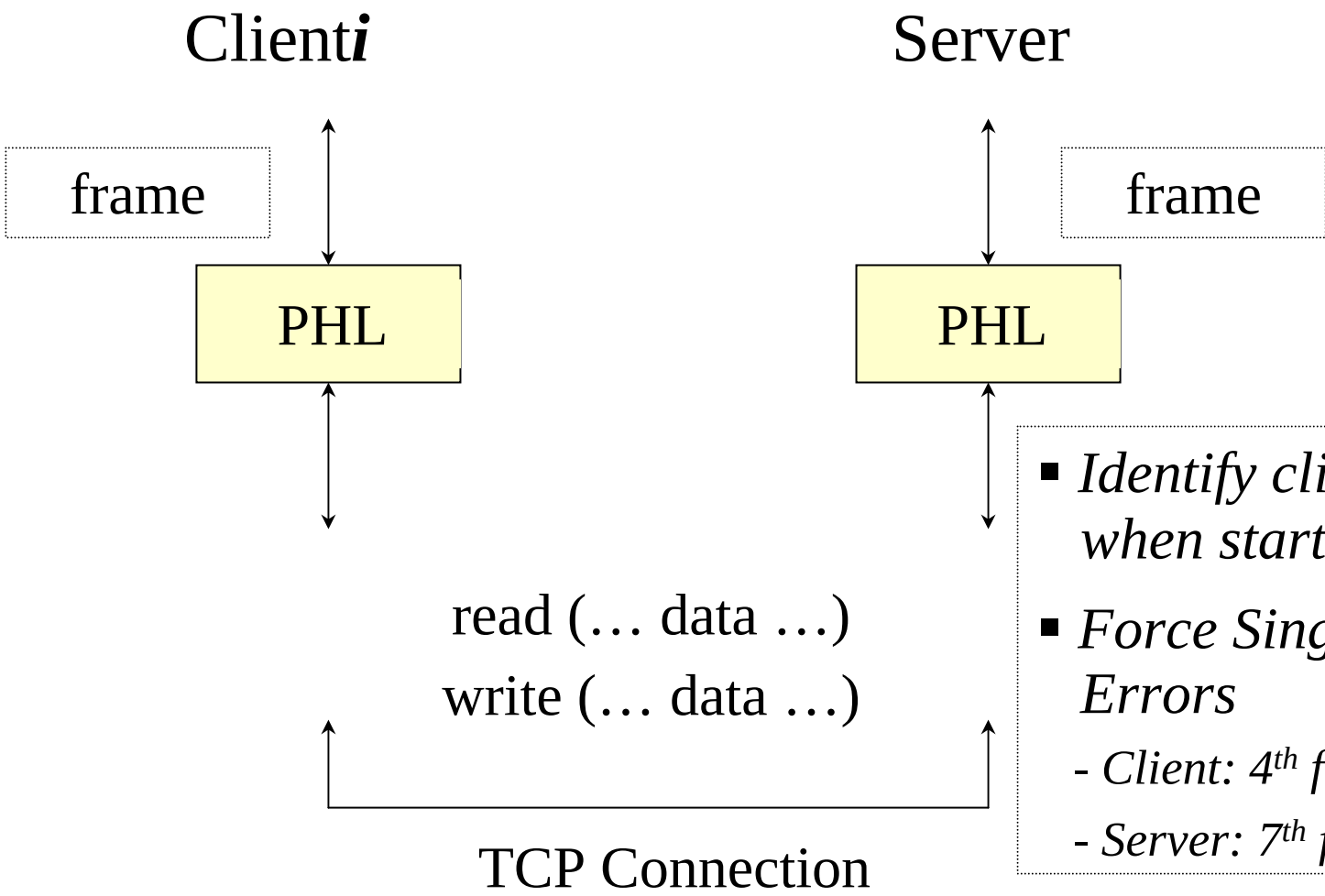
How the System Works: Layer by Layer
DataLink Layer



Note: The max_size of a frame payload is 40 bytes

Sliding window size >=3

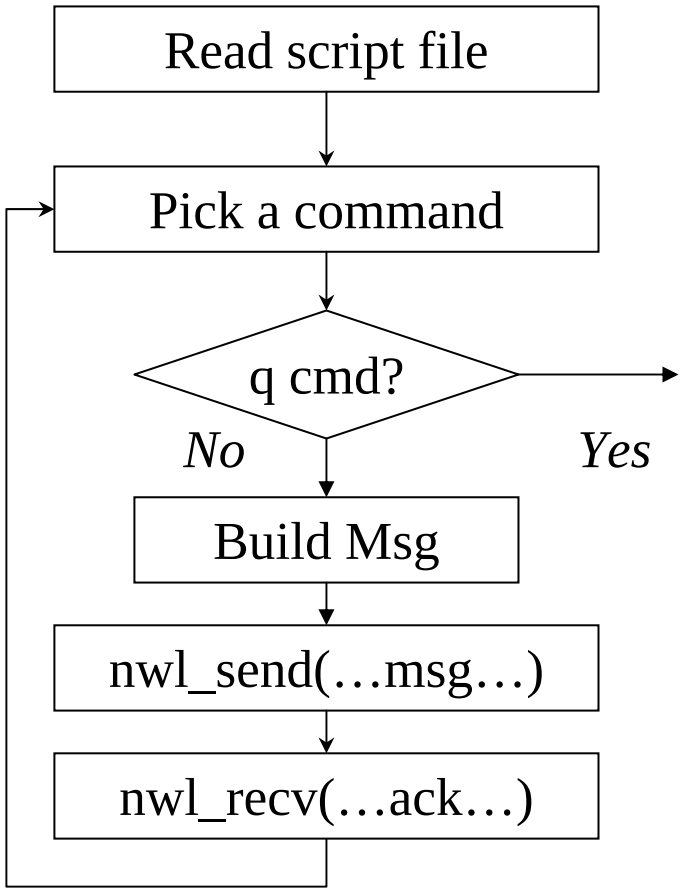
How the System Works: Layer by Layer



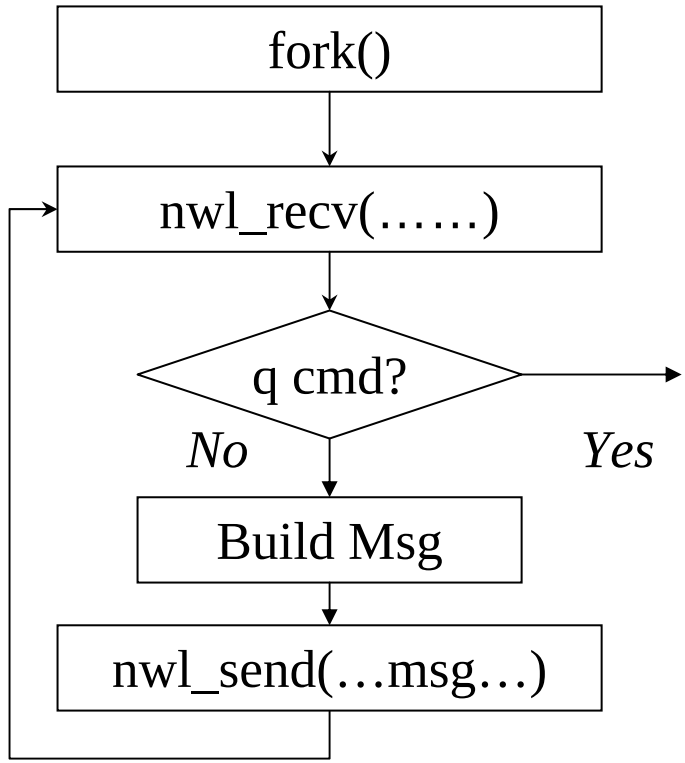
- *Identify client when start*
- *Force Single Bit Errors*
  - *Client: 4th frames*
  - *Server: 7th frames*

How the Functions Work: Layer by Layer

client APP

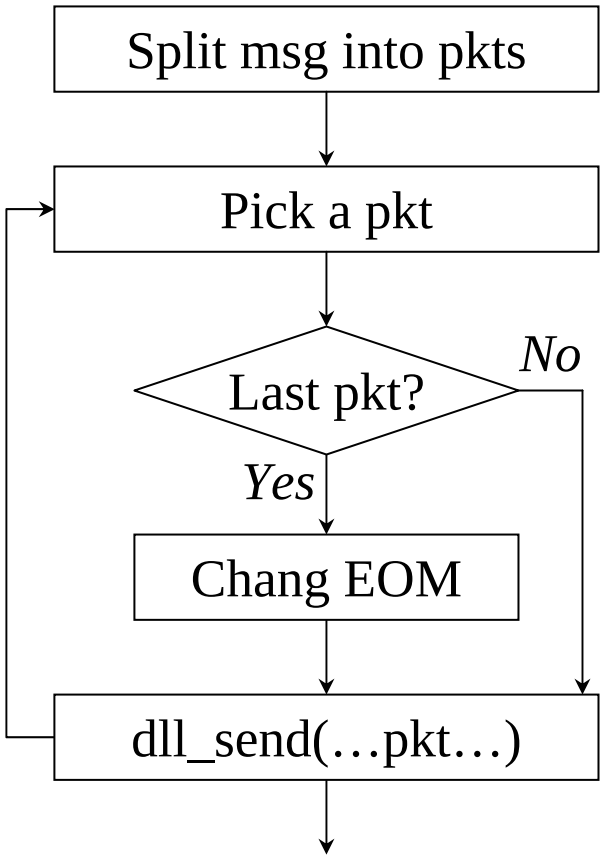


server child process APP

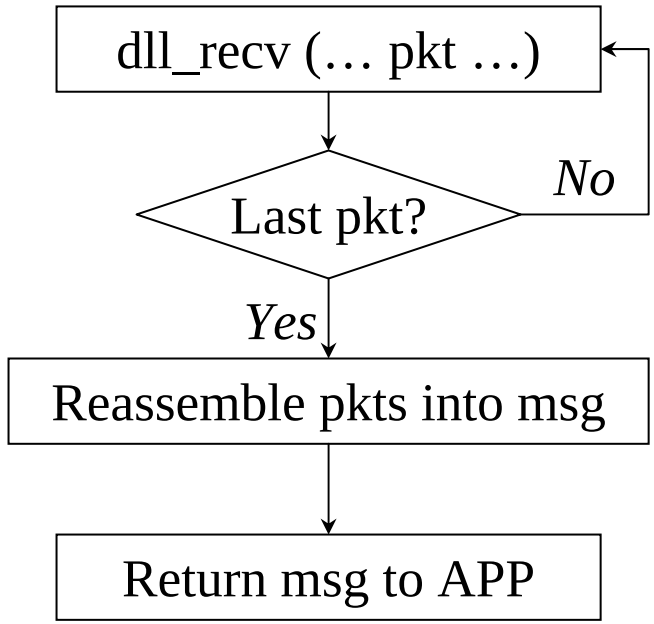


How the Functions Work: Layer by Layer

nwl_send (… msg …)



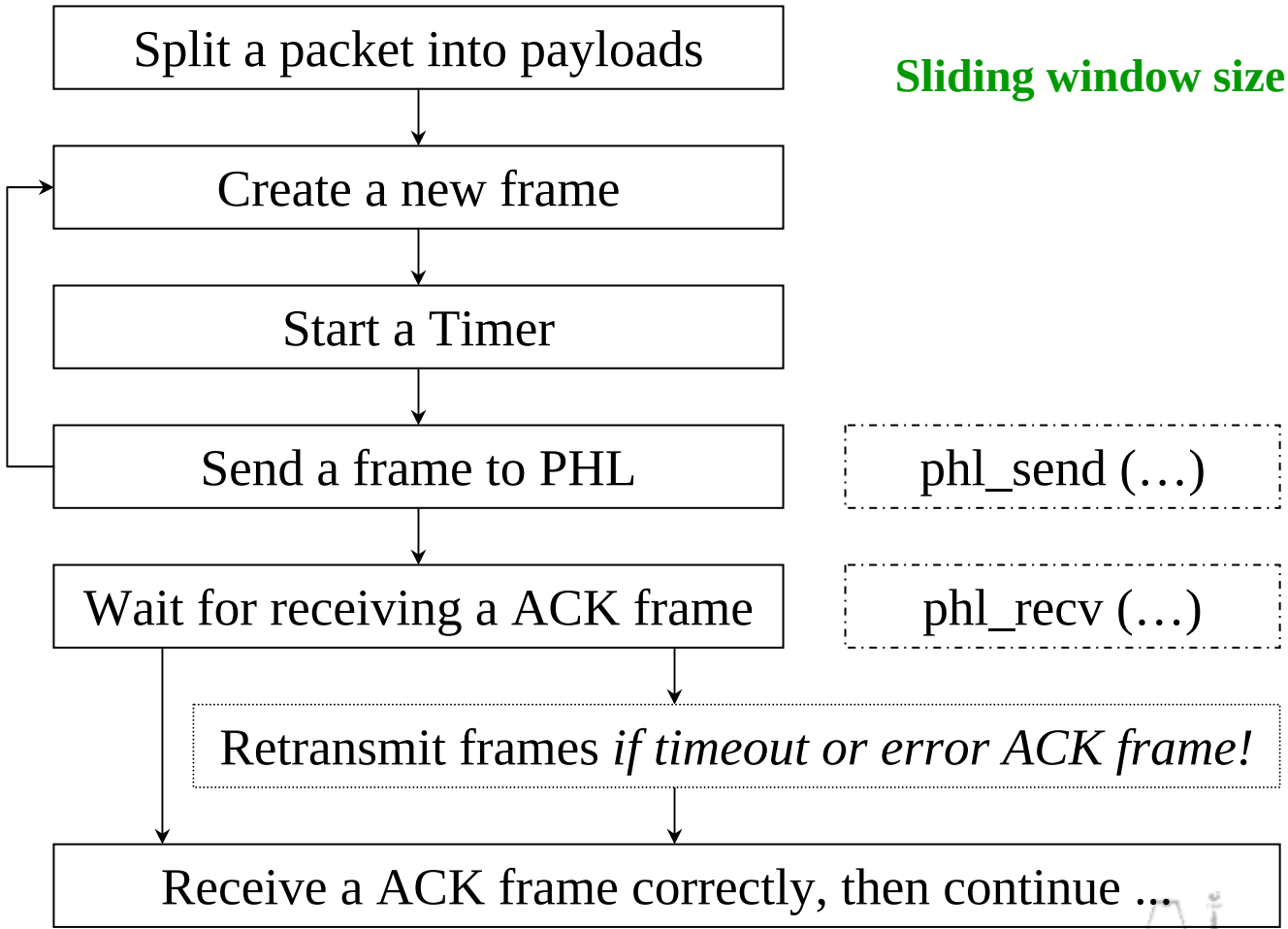
nwl_recv (… msg …)



Note: you need have a mechanism to decide the last packet in a frame.

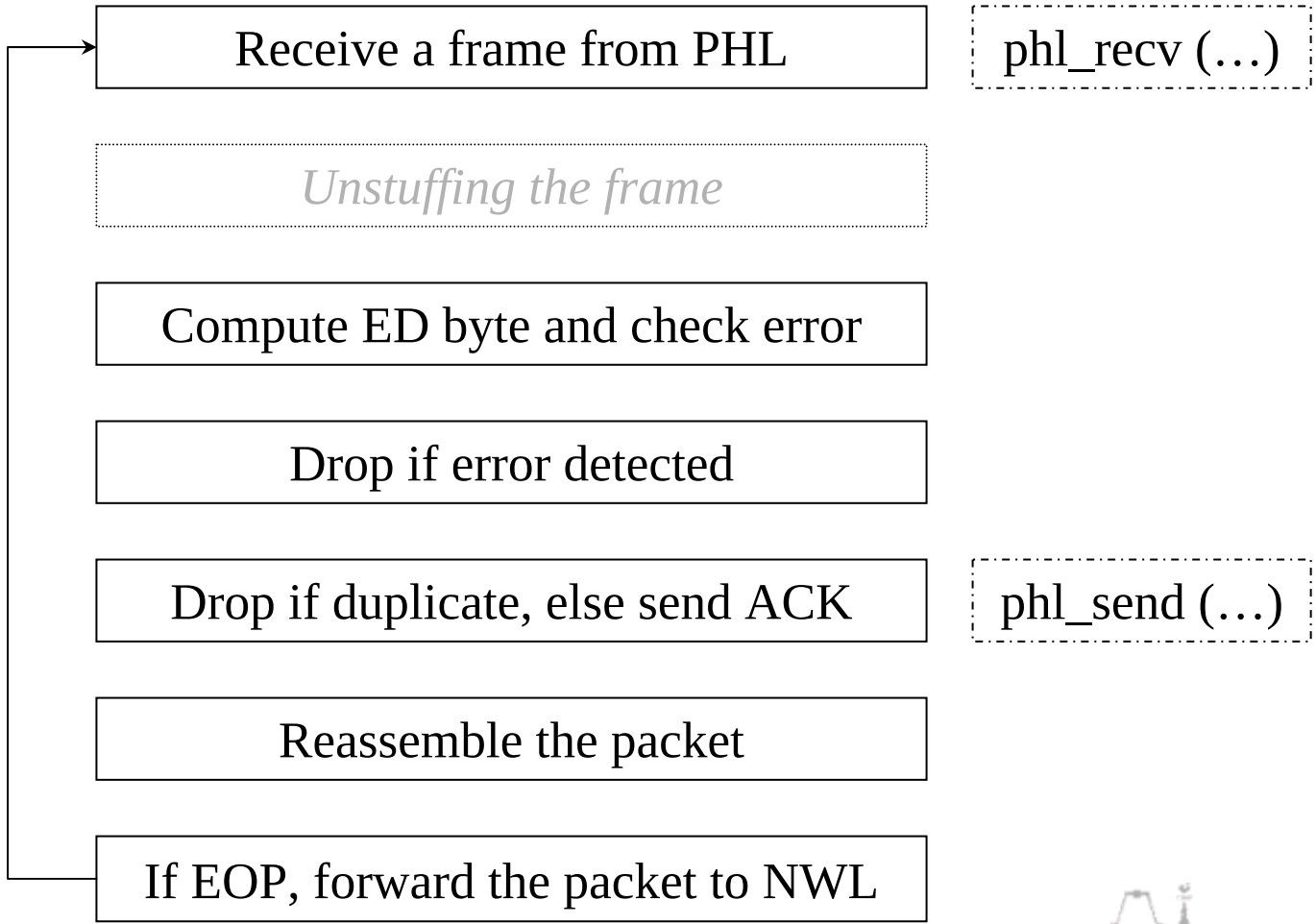
How the Functions Work: Layer by Layer

dll_send (… pkt …)

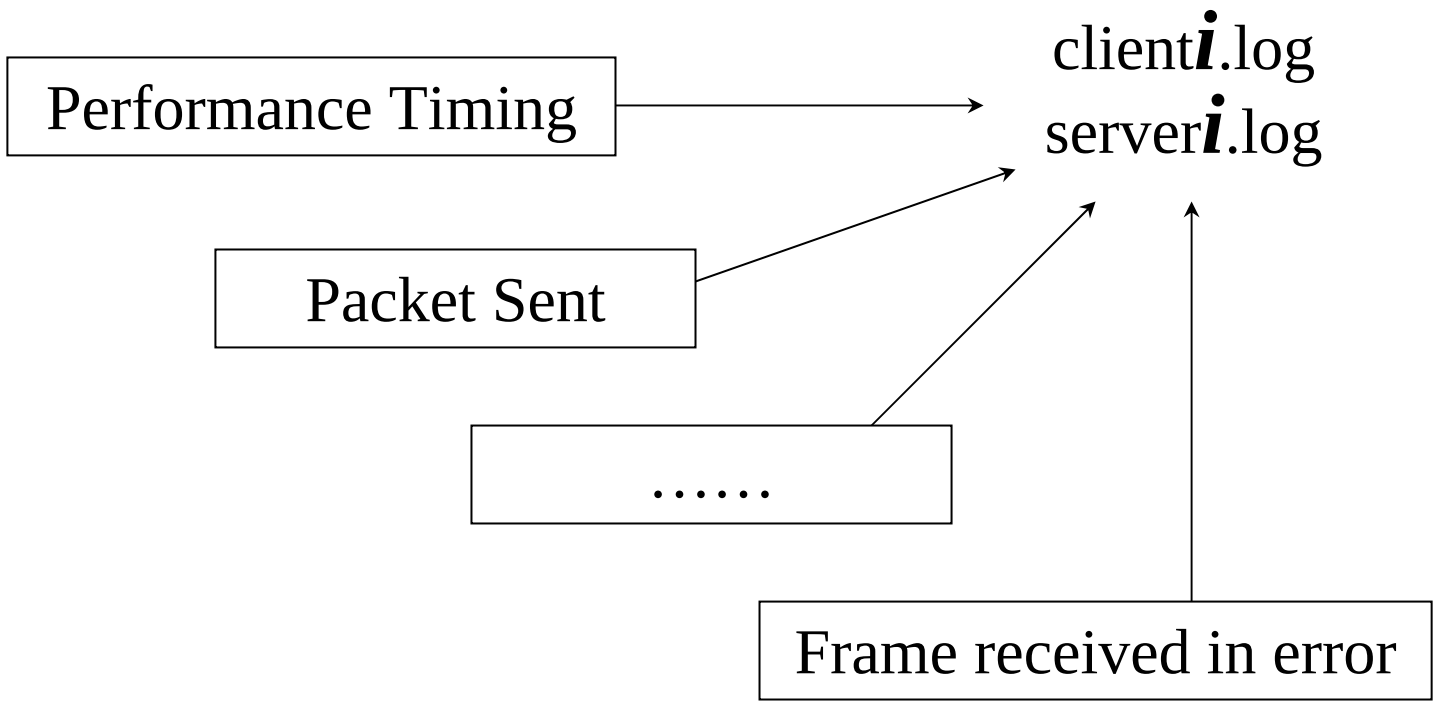


How the Functions Work: Layer by Layer

dll_recv (… pkt …)



Log Significant Events



Project Tips-1

- Sliding Window Protocol: Go-Back-N (N>=3)
  - Try to implement Go-Back-1 first
  - Then implement Go-Back-N (multiple timers)
- Maybe easier to merge PHL and DLL
- How to terminate client process:
  - When the client gets the response to the quit message
  - A "clean" way to terminate the server child process?

Project Tips-2

- Simulate multiple timer in software
  - Approach I
    - Using link list or array
    - pp.223 on textbook()
    - Need signal()
  - Approach II
    - Using link list or array
    - Update the *struct timeval* for next select() call

Project Tip3

- *How could the NWL Keep sending packets until blocked by the Data Link Layer ?*

  Our suggestion is that you could use pipe to implement it: NWL keeps writing packets to the pipe until the pipe is full.
- A simple code of pipe could be found at

  http://thor.prohosting.com/~nupshot21/Unix/sockets/node46.shtml
- Pipe is more like a socket between local processes.

Concurrent TCP Server Example (fork)

```
pid_t pid;
int listenfd, connfd;


/* 1. create a socket socket() */
if ((listenfd = socket(AF_INET, SOCK_STREAM, 0)) < 0 )
err_quit("build server socket error\n", -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, LISTENQ);
while(1){
    connfd = accept(listenfd, ... );  /* probably blocks */
    if(( pid = fork()) == 0){
        close(listenfd);  /* child closes listening socket */
        doit(connfd);     /* process the request */
        close(connfd);    /* done with this client */
        exit(0);
    }
    close(connfd);    /* parent closes connected socket */
}
```