CS4514 HELP Session 3

Concurrent Server Using Go-Back-N

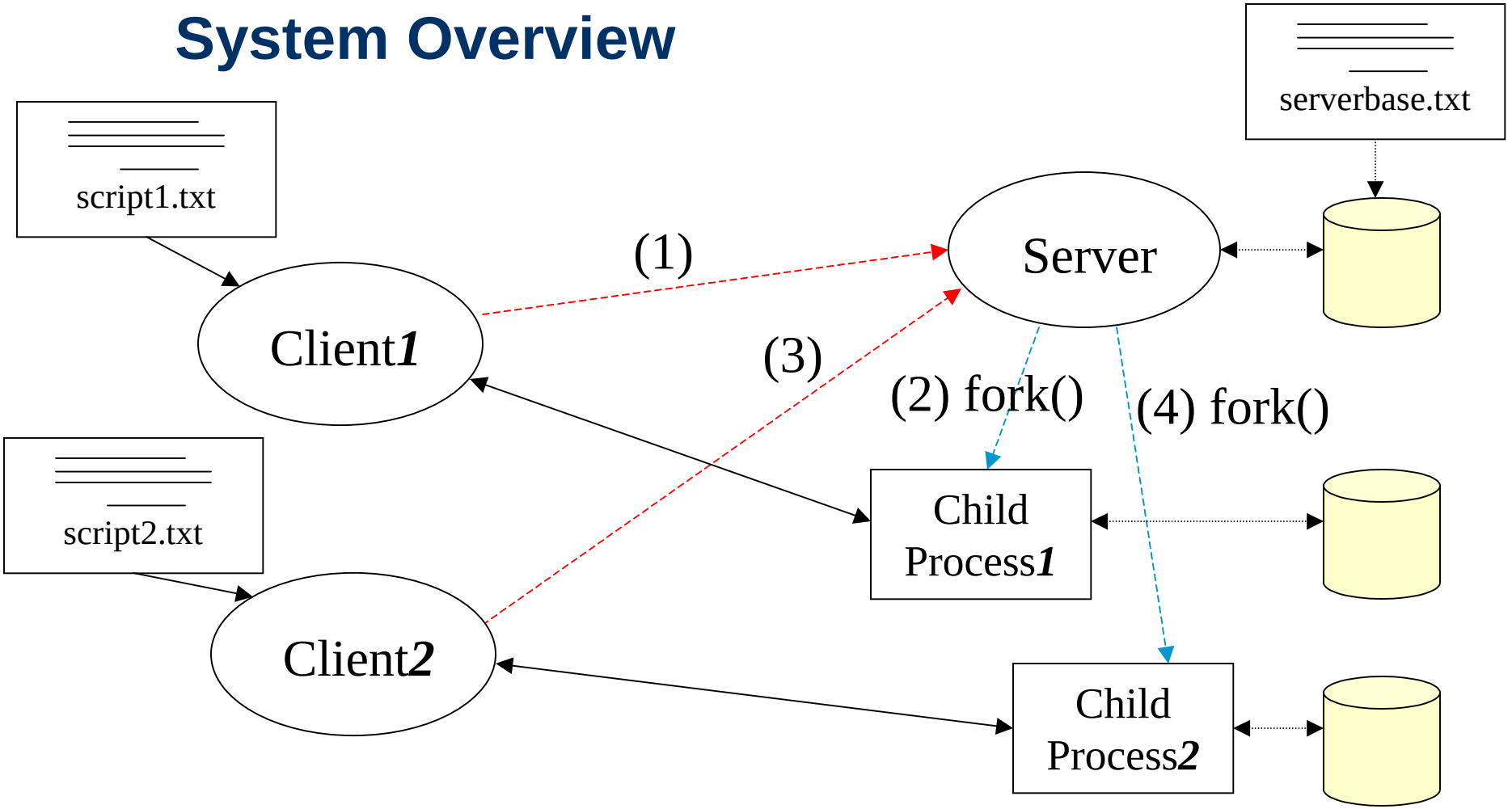
Song Wang

02/17/2004

Description

- You are supposed to implement a simple concurrent server and client having four emulated network protocol stack.
  - 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 one of ccc.WPI.EDU.

System Overview

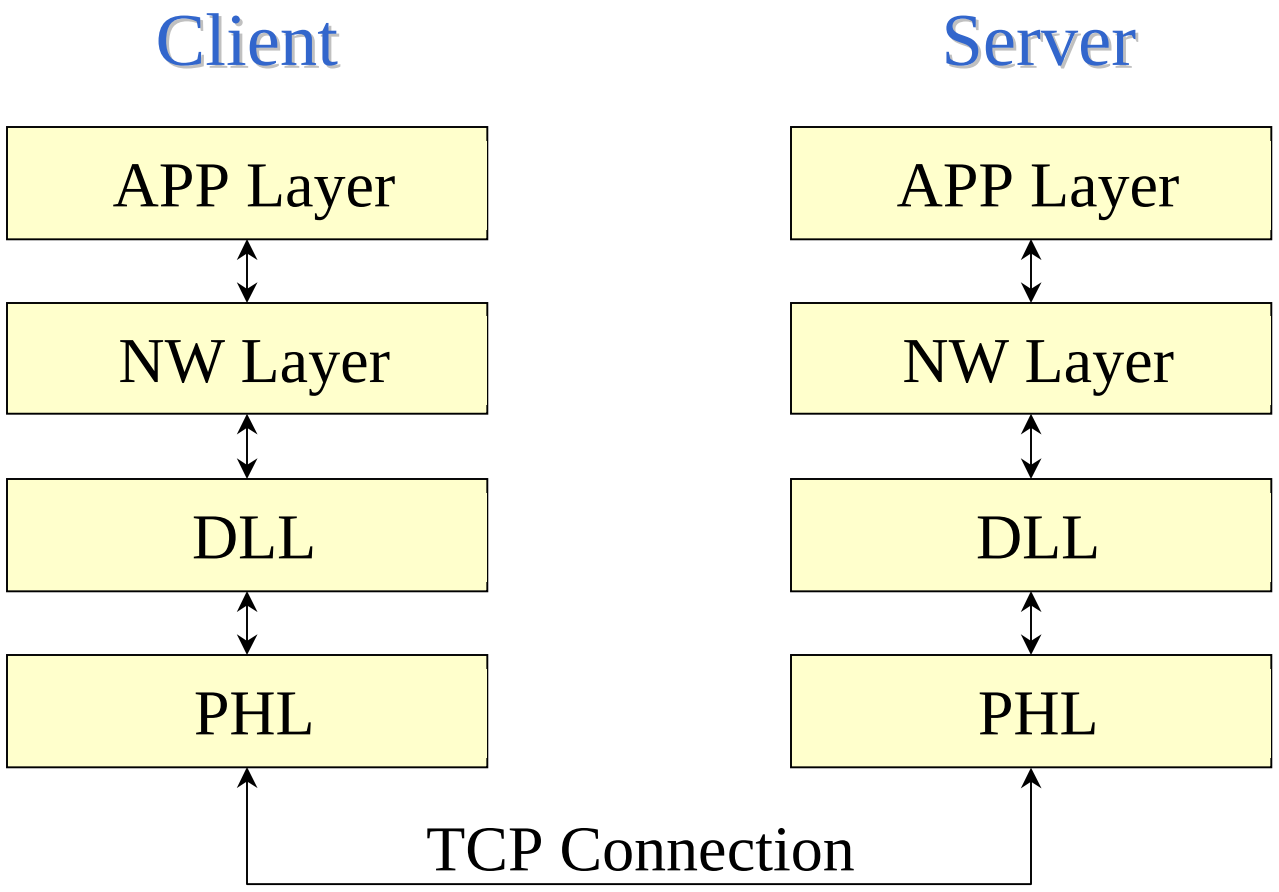


Note: each child process keeps a separate copy of the DB.

we do not keep data consistency for the serverbase

This is automatically done by using fork()

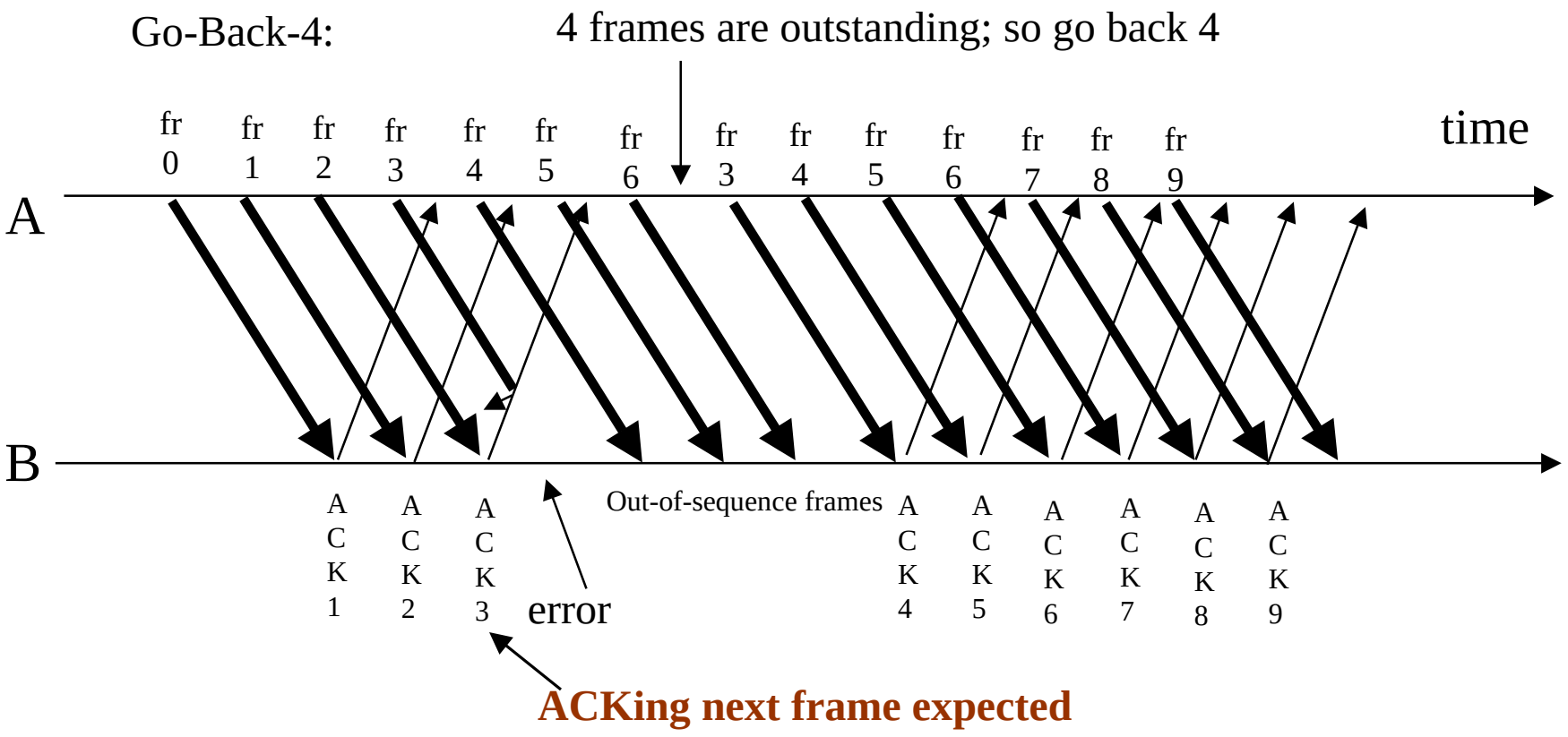
System Framework



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.

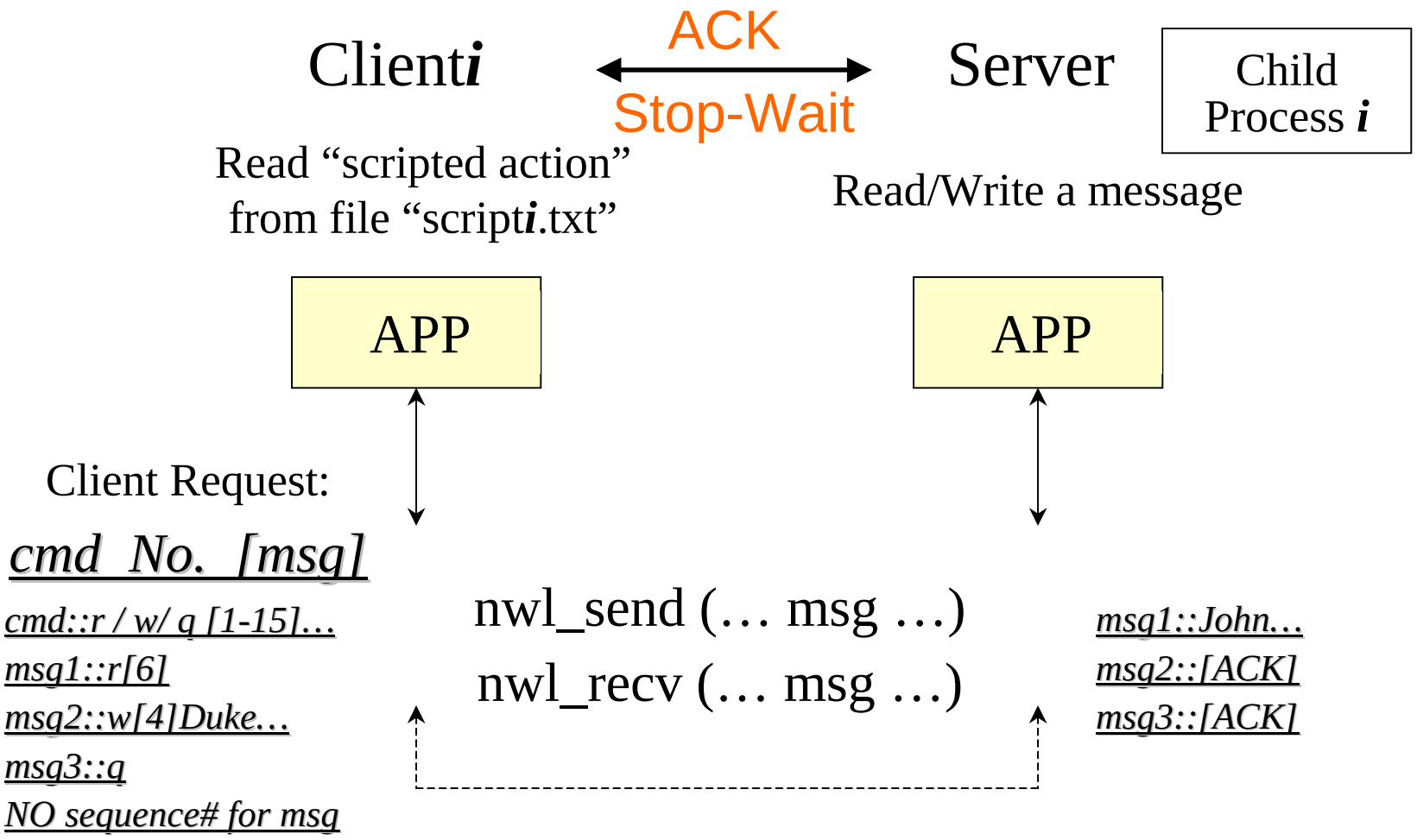
Go Back N



Leon-Garcia & Widjaja: *Communication Networks*

Figure 5.13

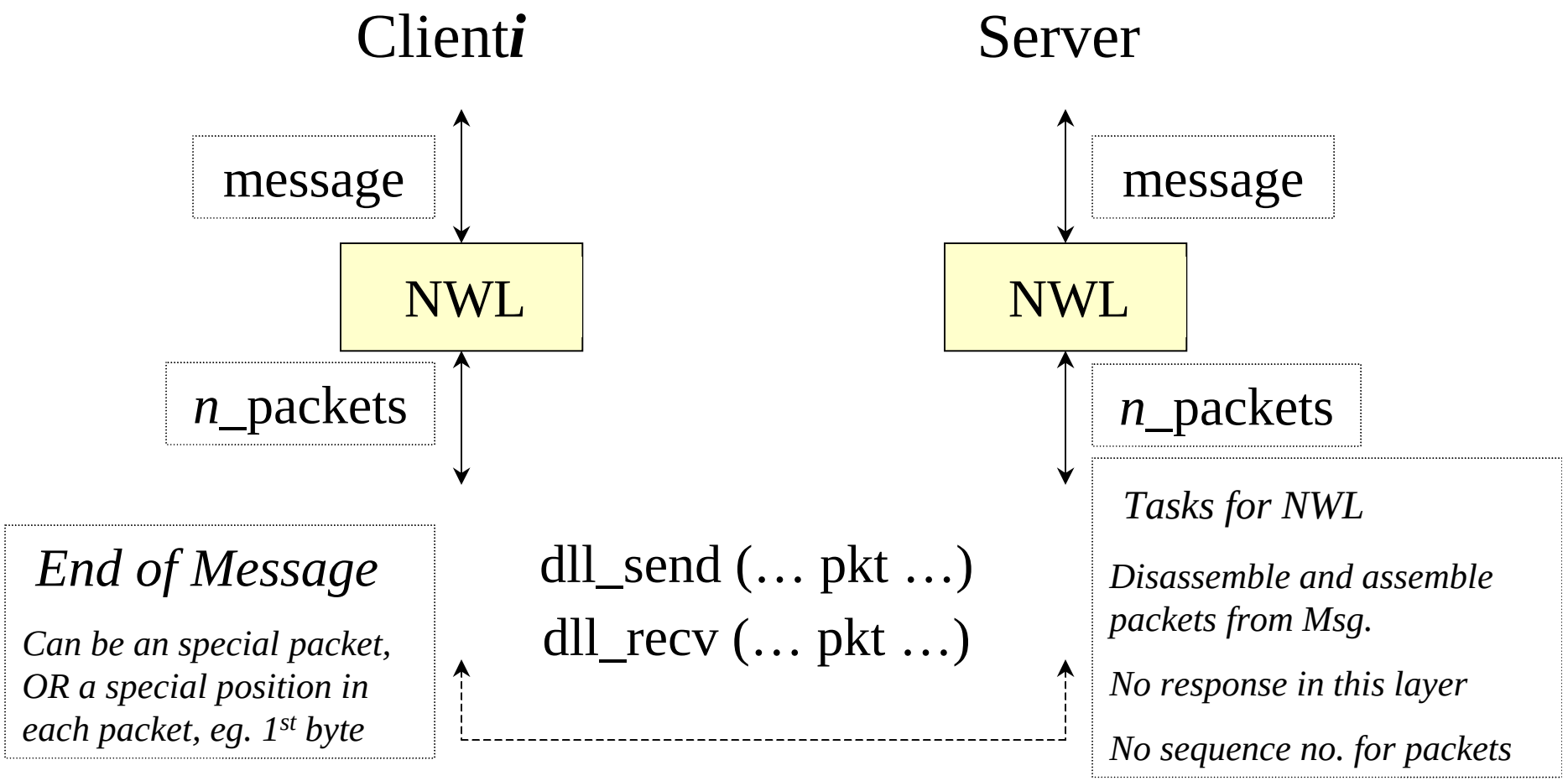
How the System Works: Layer by Layer



Note: The max_size of a message is 285 bytes

The number referring to tuple position is 1 to 15

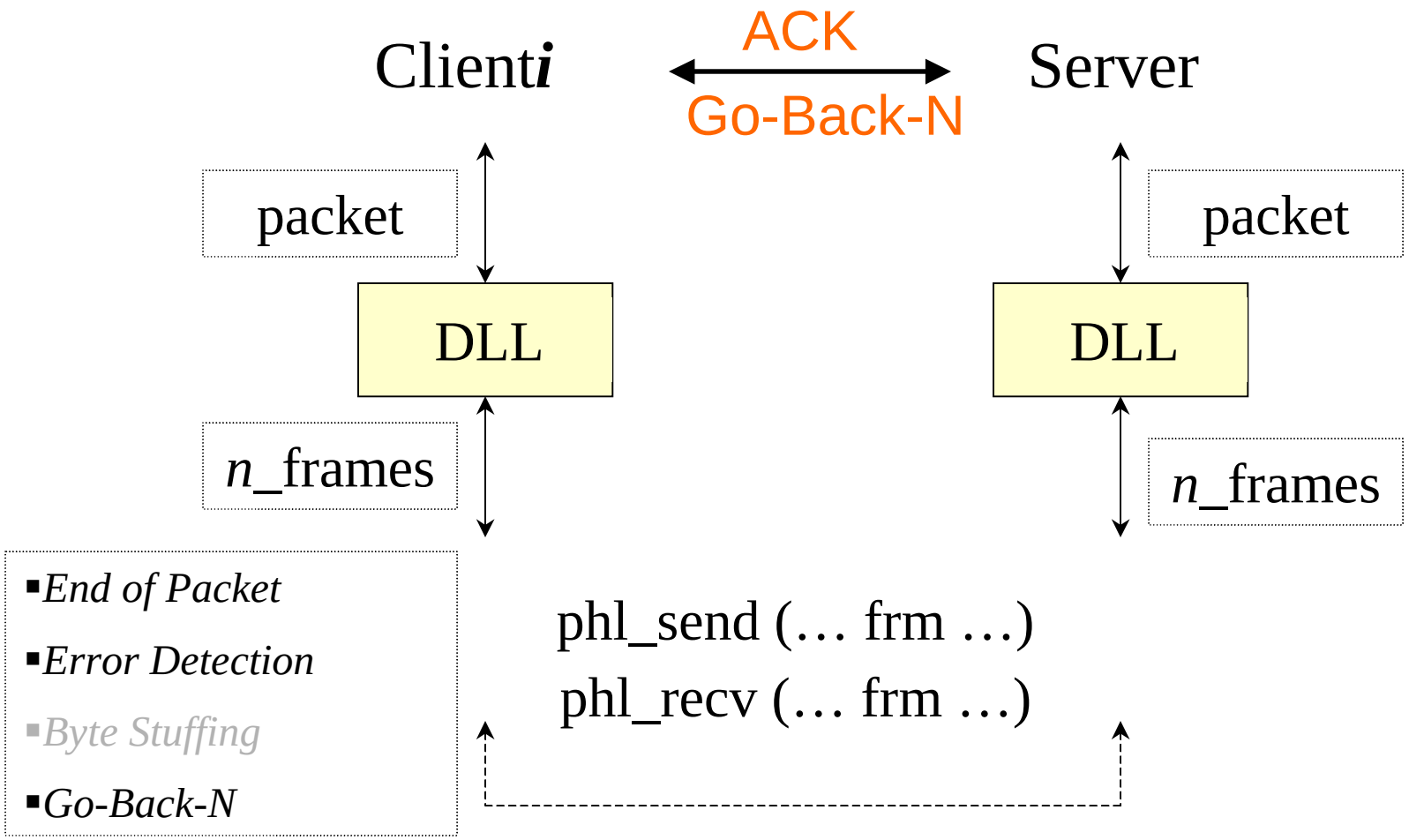
How the System Works: Layer by Layer



Note: The max_size of a packet is 70 bytes

The network layer will send packets until blocked by the Data Link Layer

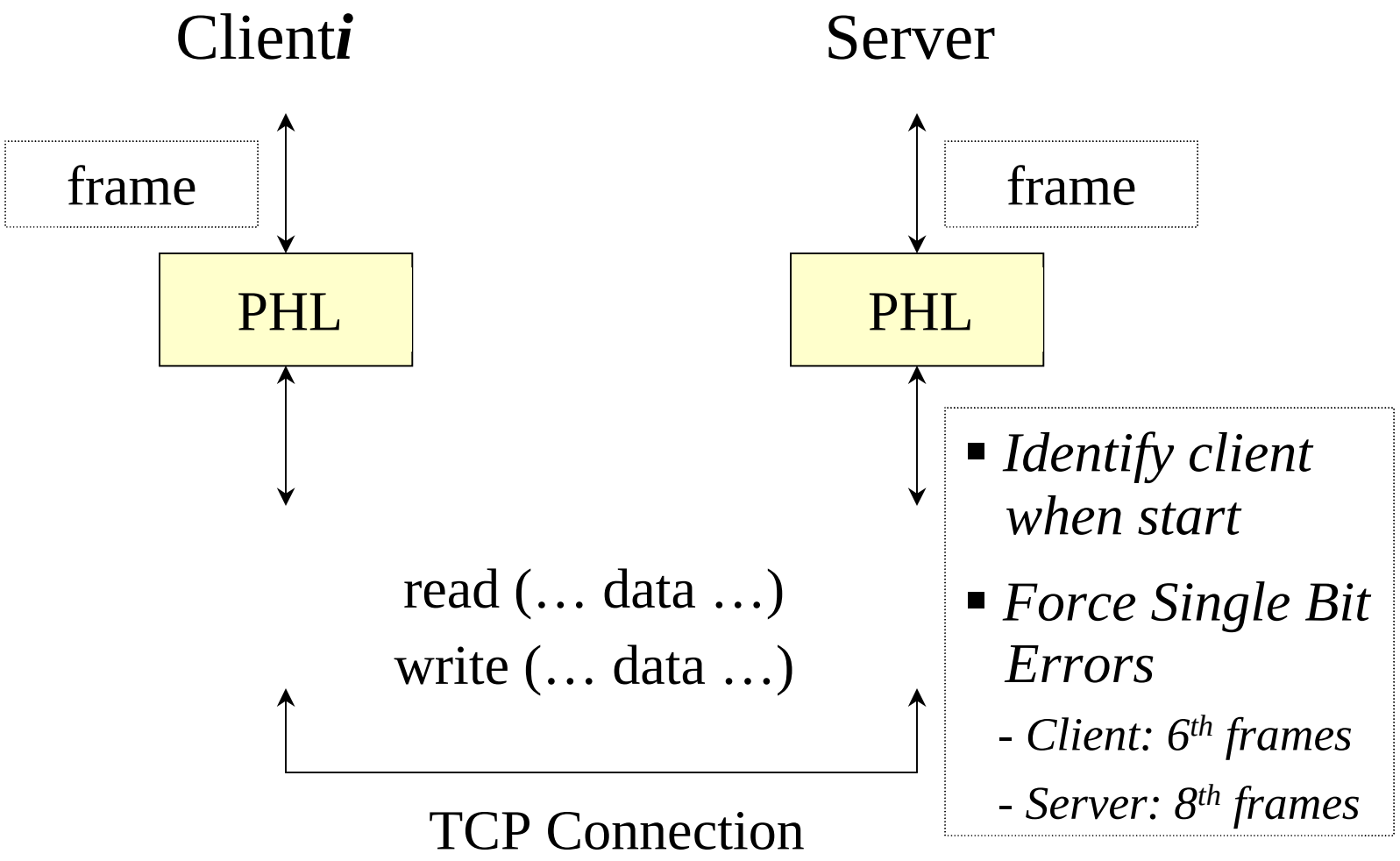
How the System Works: Layer by Layer



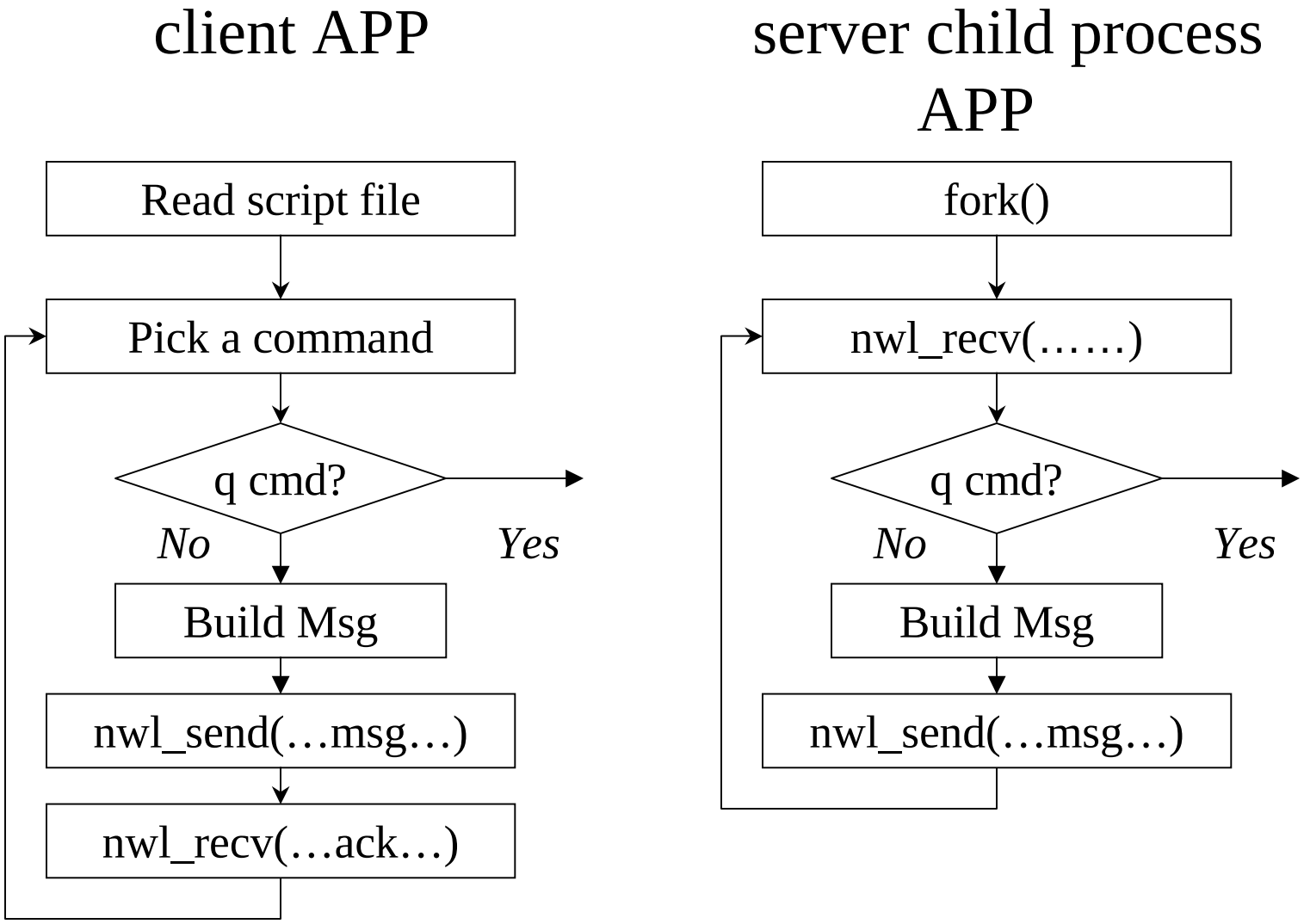
Note: The max_size of a frame payload is 45 bytes

Sliding window size >=3

How the System Works: Layer by Layer

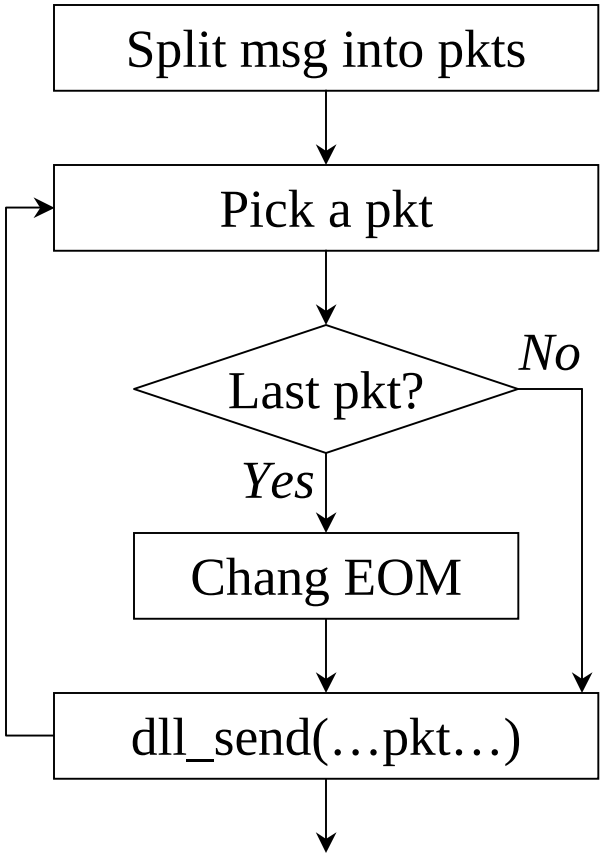


How the Functions Work: Layer by Layer

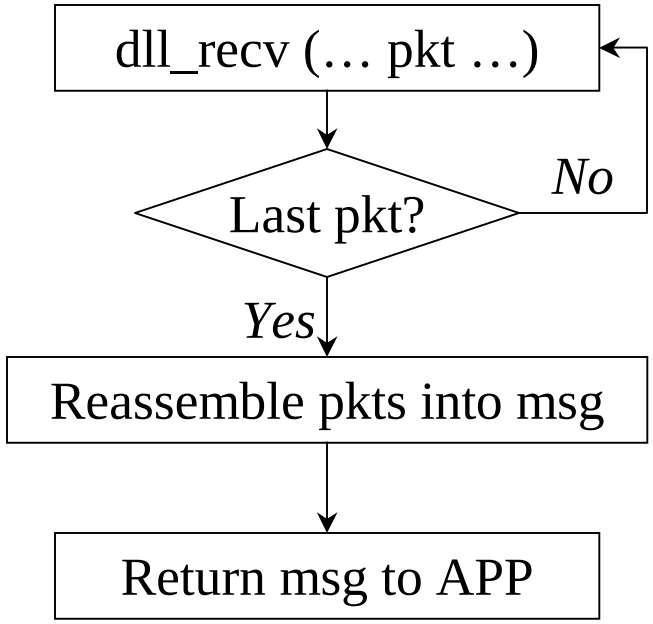


How the Functions Work: Layer by Layer

nwl_send (… msg …)

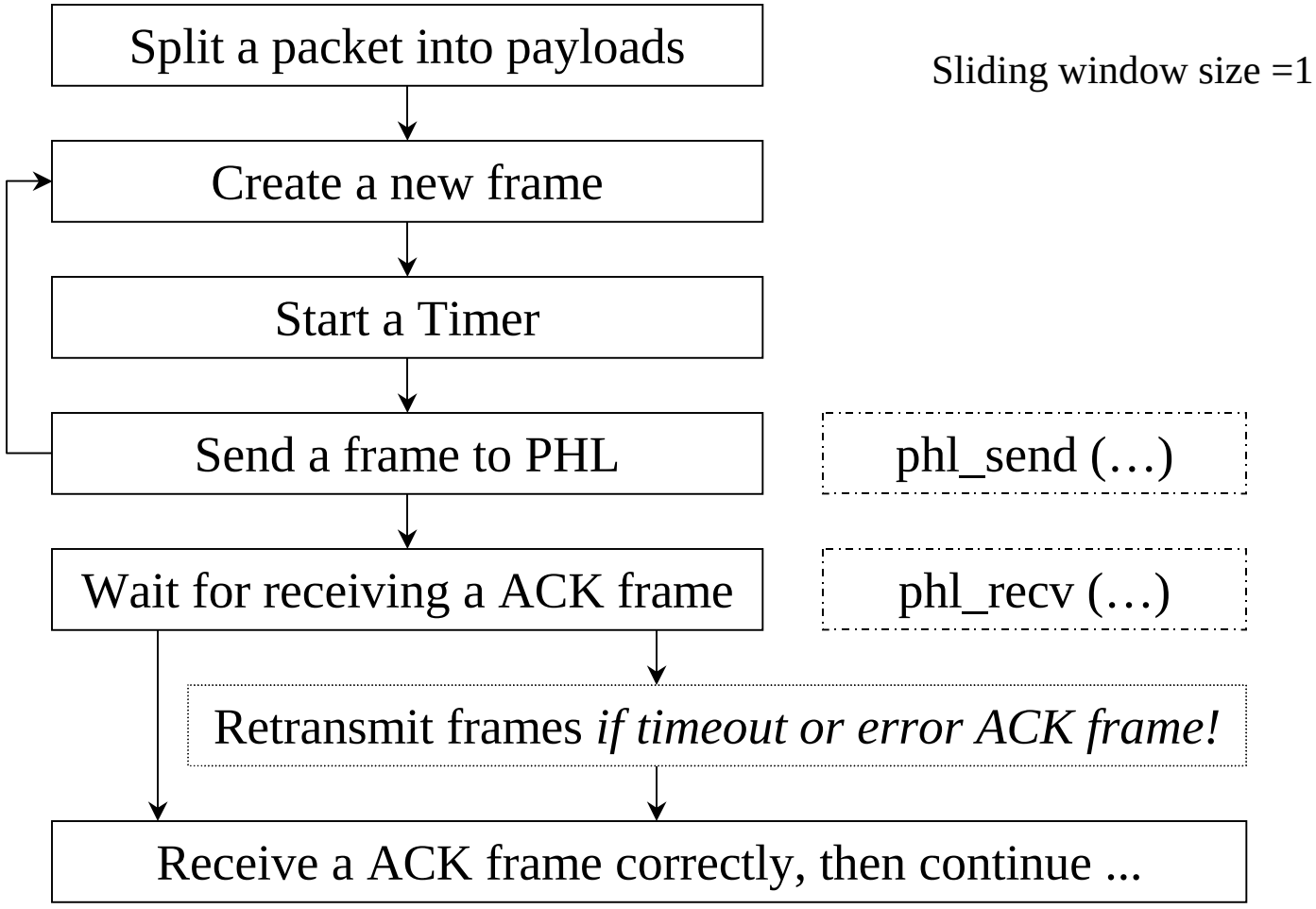


nwl_recv (… msg …)

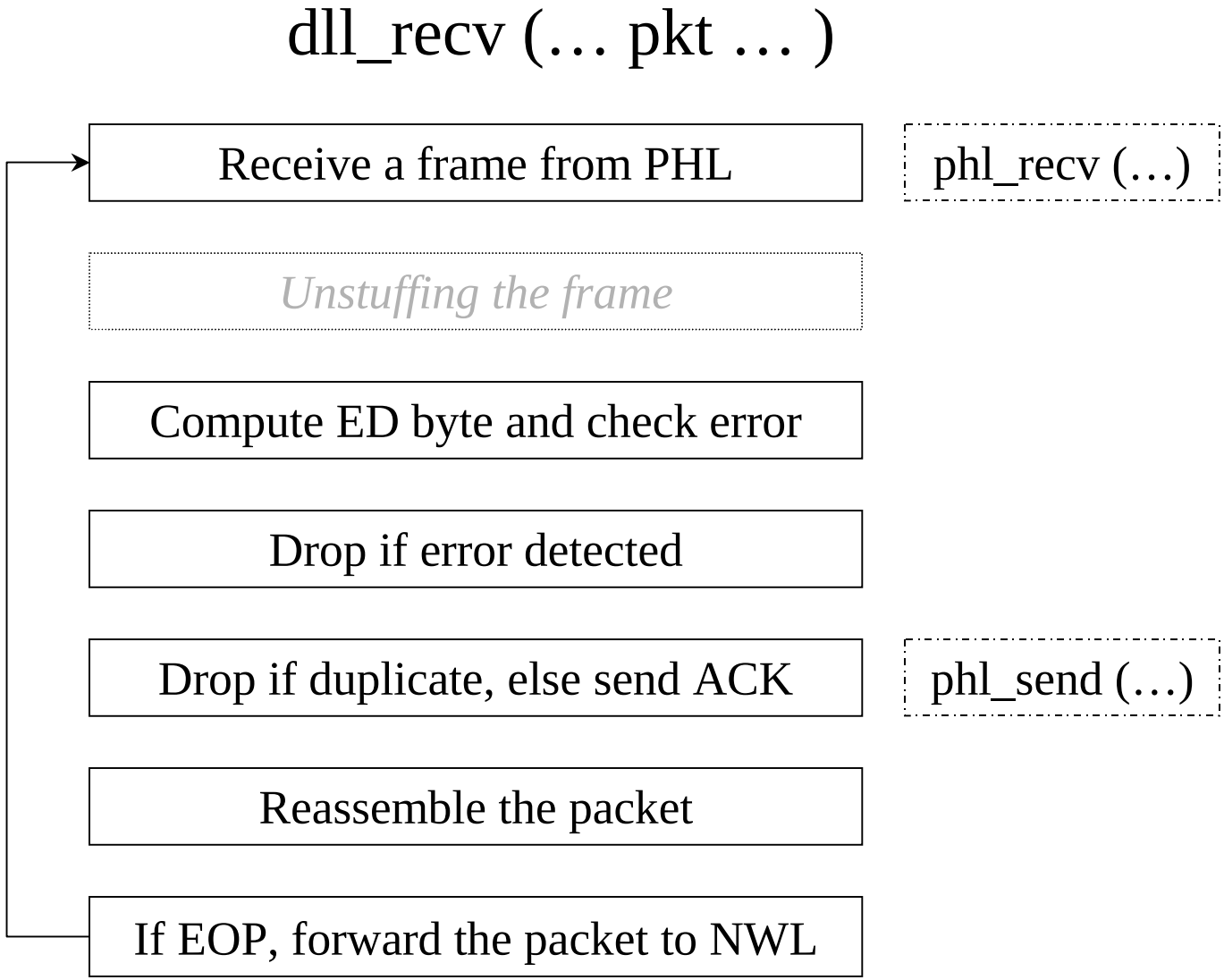


How the Functions Work: Layer by Layer

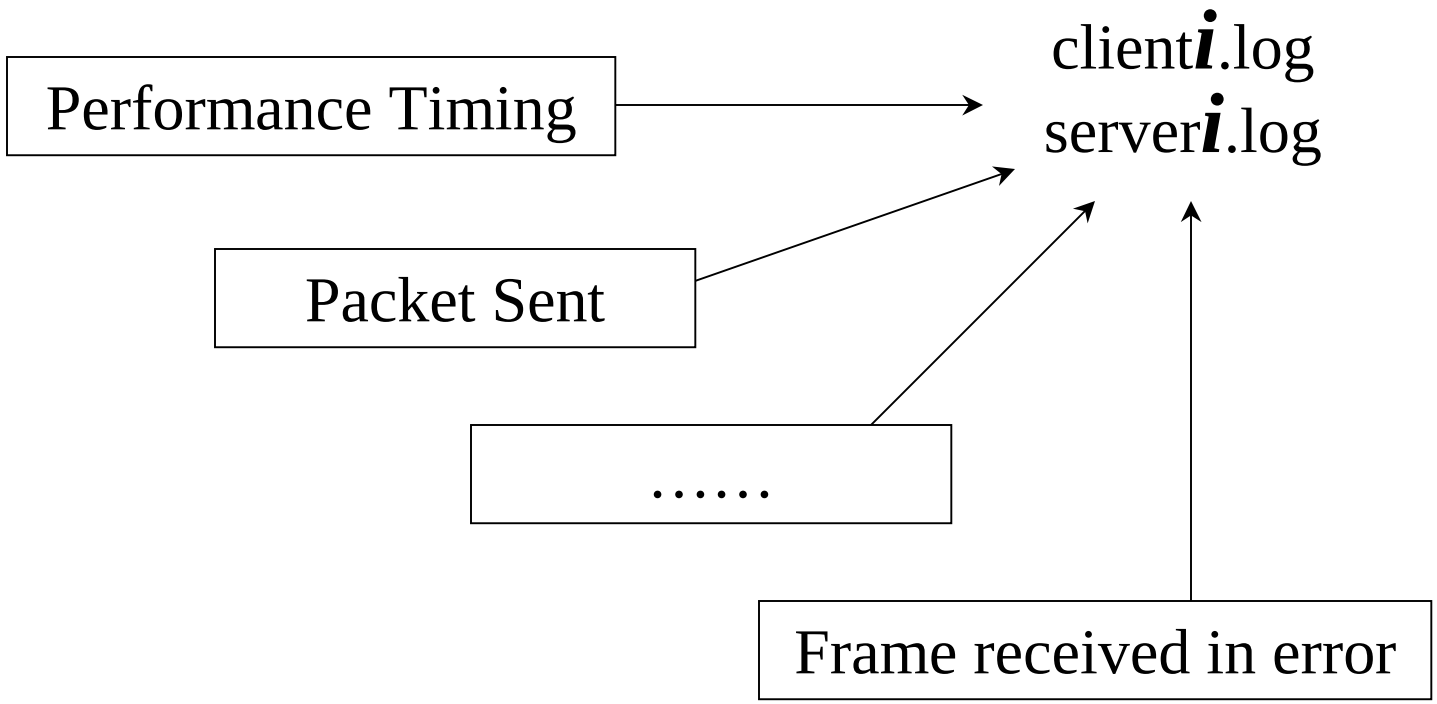
dll_send (… pkt …)



How the Functions Work: Layer by Layer



Log Significant Events



Project Tips

- 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

- 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

Concurrent TCP Server Example

```
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 */
}
```