Introduction

Introduction to Network Programming

Chun-Ying Huang <chuang@cs.nctu.edu.tw>

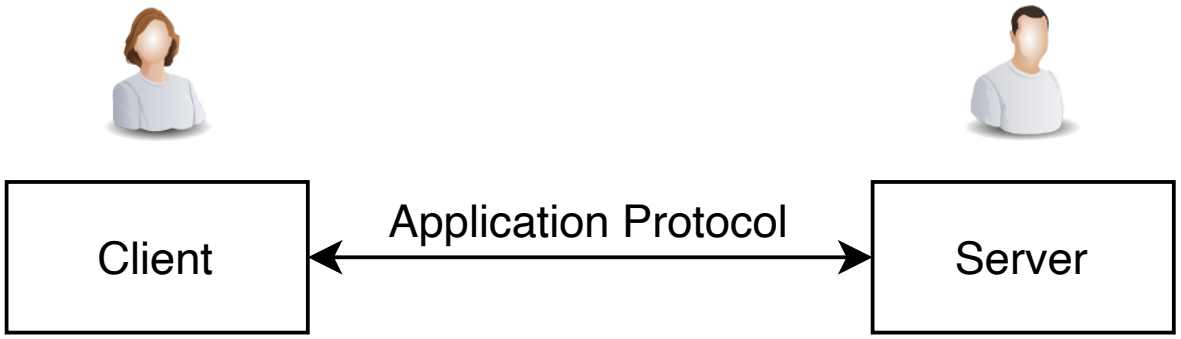
Outline

- Introduction
- Simple client
- Protocol independence
- Simple server
- OSI model

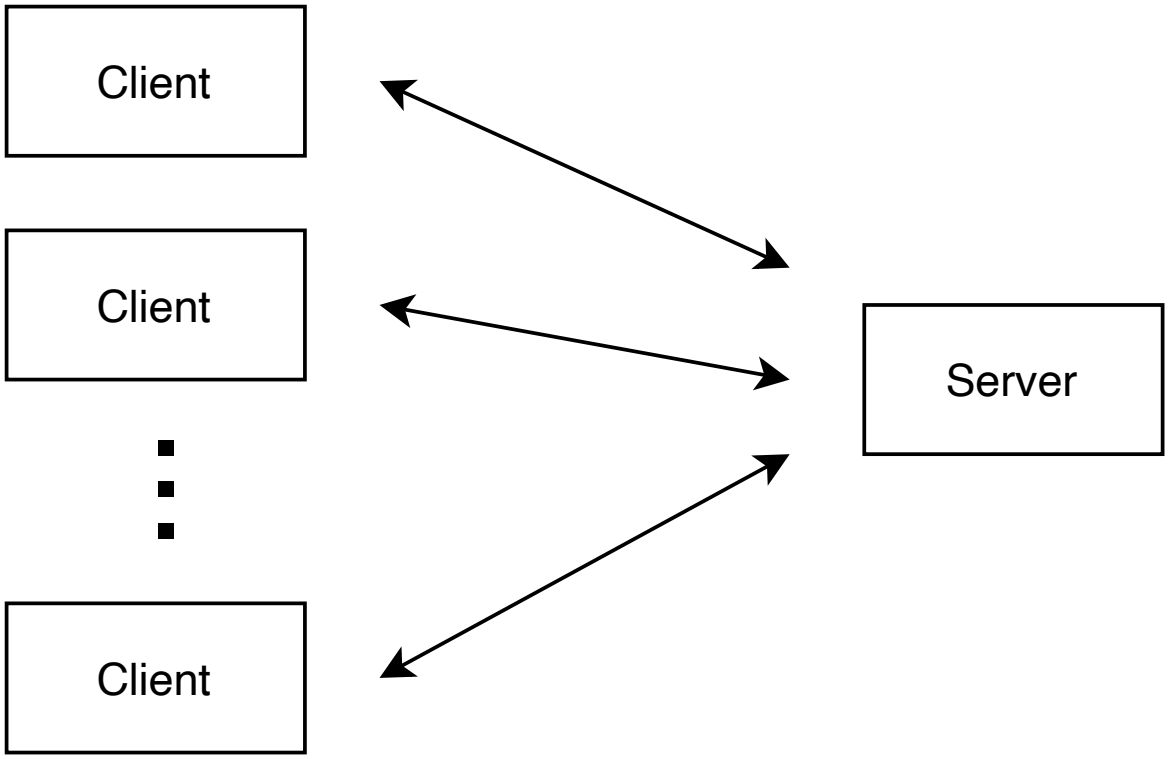
Introduction

- How do network application work?
- Architecture overview
- Sample codes in the textbook
- OSI and TCP/IP model
- Some fundamental commands

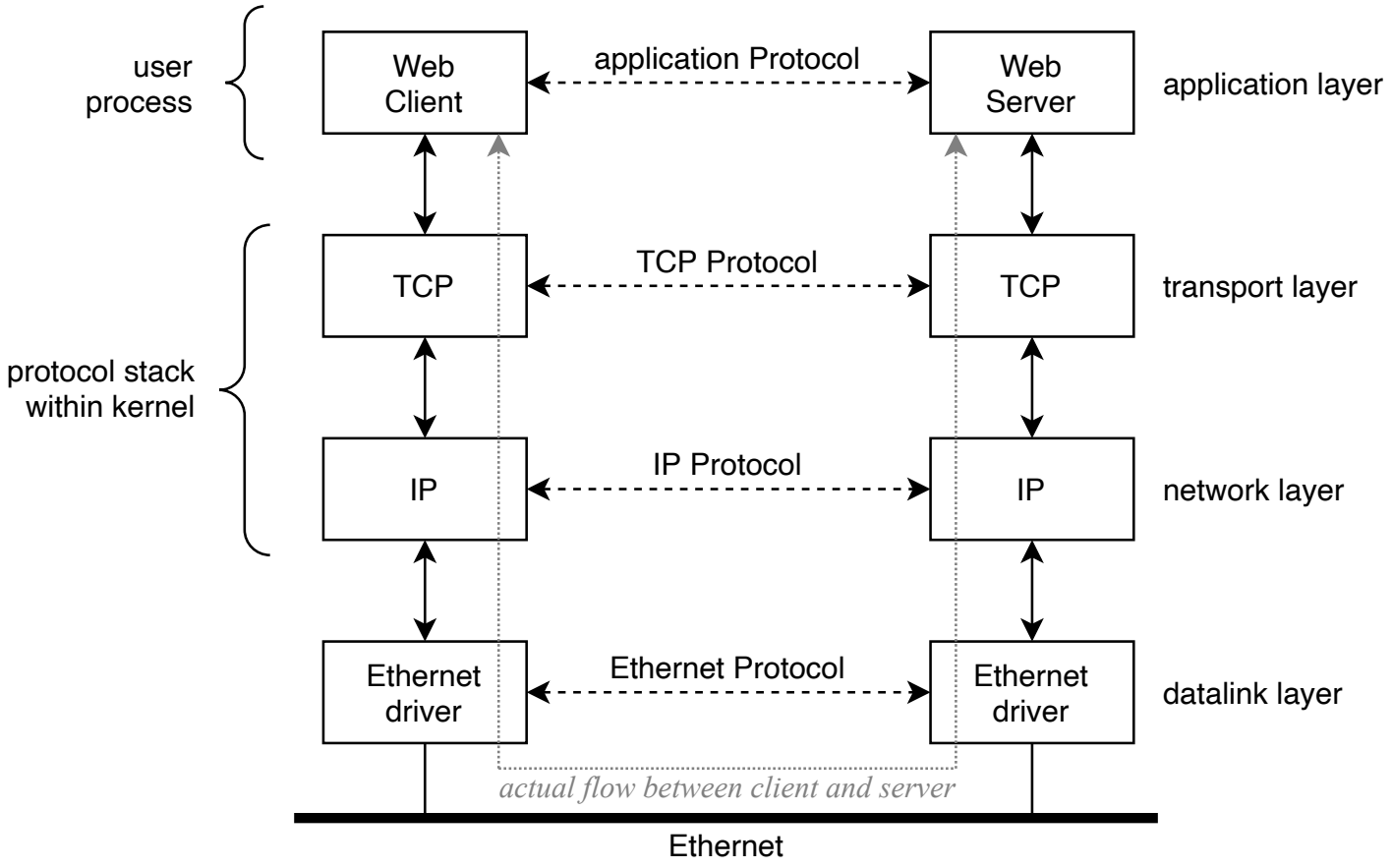
Network Application – The Simplest View



Network Application – Multiple Clients at a Time

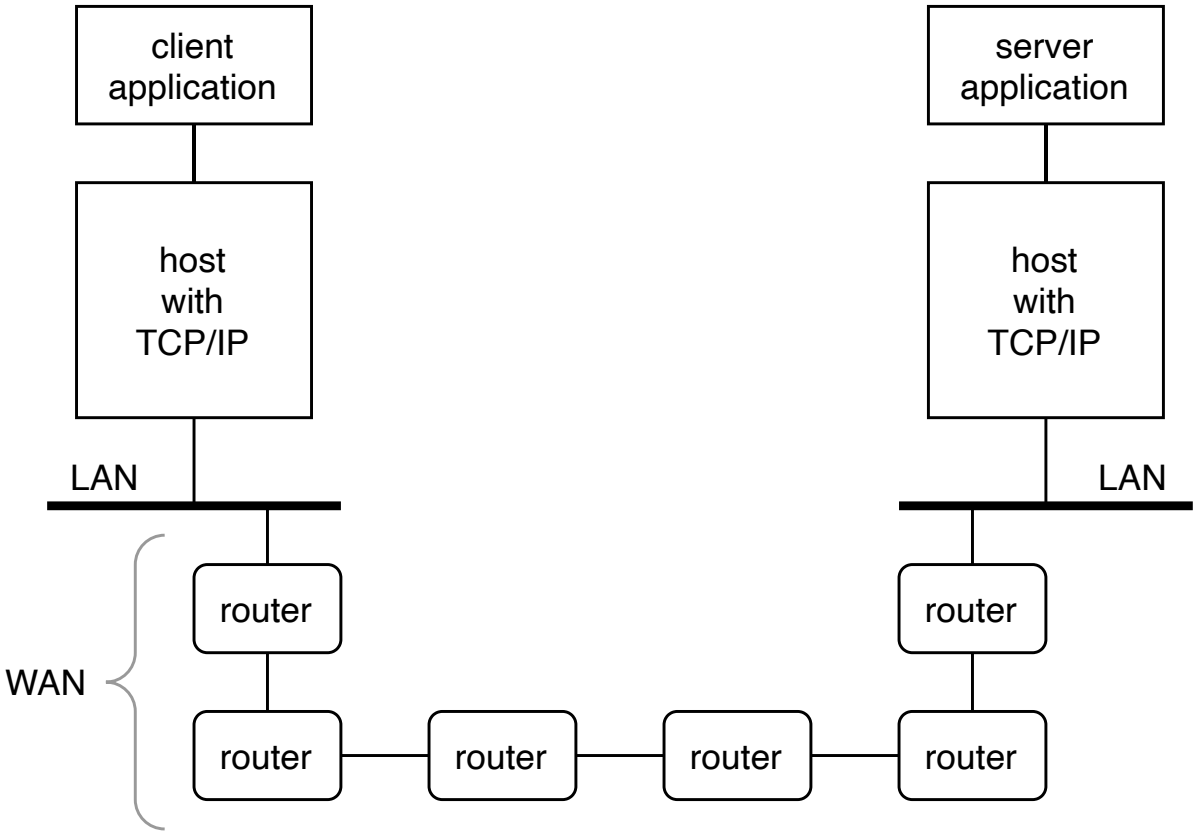


Zoom-In the Application Protocols



** unit for data transmission over the network: packet

Talk over the Internet



Textbook Sample Book: daytimecli

```c
#include	"unp.h"

int
main(int argc, char **argv)
{
	int					sockfd, n;
	char				recvline[MAXLINE + 1];
	struct sockaddr_in	servaddr;

	if (argc != 2)
		err_quit("usage: a.out <IPaddress>");

	if ( (sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
		err_sys("socket error");

	bzero(&servaddr, sizeof(servaddr));
	servaddr.sin_family = AF_INET;
	servaddr.sin_port   = htons(13);	/* daytime server */
	if (inet_pton(AF_INET, argv[1], &servaddr.sin_addr) <= 0)
		err_quit("inet_pton error for %s", argv[1]);

	if (connect(sockfd, (SA *) &servaddr, sizeof(servaddr)) < 0)
		err_sys("connect error");

	while ( (n = read(sockfd, recvline, MAXLINE)) > 0) {
		recvline[n] = 0;	/* null terminate */
		if (fputs(recvline, stdout) == EOF)
			err_sys("fputs error");
	}
	if (n < 0)
		err_sys("read error");

	exit(0);
}
```

Sample Running Scripts

```
$ git clone https://github.com/unpbook/unpv13e.git
$ cd unpv13e
$ ./configure
$ (cd lib; make)
$ (cd intro; make)
```

Alternative Running Scripts

(suppose libunp.a has been built)

```
(in intro directory)
$ gcc -I../lib -L.. daytimetcpcli.c -lunp
```

Source code availability: https://github.com/unpbook/unpv13e/blob/master/intro/daytimetcpcli.c

Source code availability: https://github.com/unpbook/unpv13e

daytimecli – Example

- You need to run the server first – need root?
- Other relevant commands
  - netstat -nap

daytimecli – Example

```
1  #include       "unp.h"
2
3  int
4  main(int argc, char **argv)
5  {
6          int                                     sockfd, n;
7          char                                recvline[MAXLINE + 1];
8          struct sockaddr_in      servaddr;
9
10         if (argc != 2)
11                 err_quit("usage: a.out <IPaddress>");
12
13         if ( (sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
14                 err_sys("socket error");
15
16         bzero(&servaddr, sizeof(servaddr));
17         servaddr.sin_family = AF_INET;
18         servaddr.sin_port   = htons(13);        /* daytime server */
19         if (inet_pton(AF_INET, argv[1], &servaddr.sin_addr) <= 0)
20                 err_quit("inet_pton error for %s", argv[1]);
21
22         if (connect(sockfd, (SA *) &servaddr, sizeof(servaddr)) < 0)
23                 err_sys("connect error");
24
25         while ( (n = read(sockfd, recvline, MAXLINE)) > 0) {
26                 recvline[n] = 0;        /* null terminate */
27                 if (fputs(recvline, stdout) == EOF)
28                         err_sys("fputs error");
29         }
30         if (n < 0)
31                 err_sys("read error");
32
33         exit(0);
34 }
```

Key Points

- argv
- socket
- struct socketaddr_in
- inet_pton
- connect
- read

Function Wrapper

- lib/wrapsock.c
- “Uppercase” naming convention in the textbook
- Enforced error check in wrappers

```
287  /* include Socket */
288  int
289  Socket(int family, int type, int protocol)
290  {
291          int             n;
292
293          if ( (n = socket(family, type, protocol)) < 0)
294                  err_sys("socket error");
295          return(n);
296  }
```

daytimesrv – Example

```
1  #include        "unp.h"
2  #include        <time.h>
3
4  int
5  main(int argc, char **argv)
6  {
7          int                                     listenfd, connfd;
8          struct sockaddr_in      servaddr;
9          char                            buff[MAXLINE];
10         time_t                          ticks;
11
12         listenfd = Socket(AF_INET, SOCK_STREAM, 0);
13
14         bzero(&servaddr, sizeof(servaddr));
15         servaddr.sin_family      = AF_INET;
16         servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
17         servaddr.sin_port        = htons(13);   /* daytime server */
18
19         Bind(listenfd, (SA *) &servaddr, sizeof(servaddr));
20
21         Listen(listenfd, LISTENQ);
22
23         for ( ; ; ) {
24                 connfd = Accept(listenfd, (SA *) NULL, NULL);
25
26         ticks = time(NULL);
27         snprintf(buff, sizeof(buff), "%.24s\r\n", ctime(&ticks));
28         Write(connfd, buff, strlen(buff));
29
30                 Close(connfd);
31         }
32 }
```

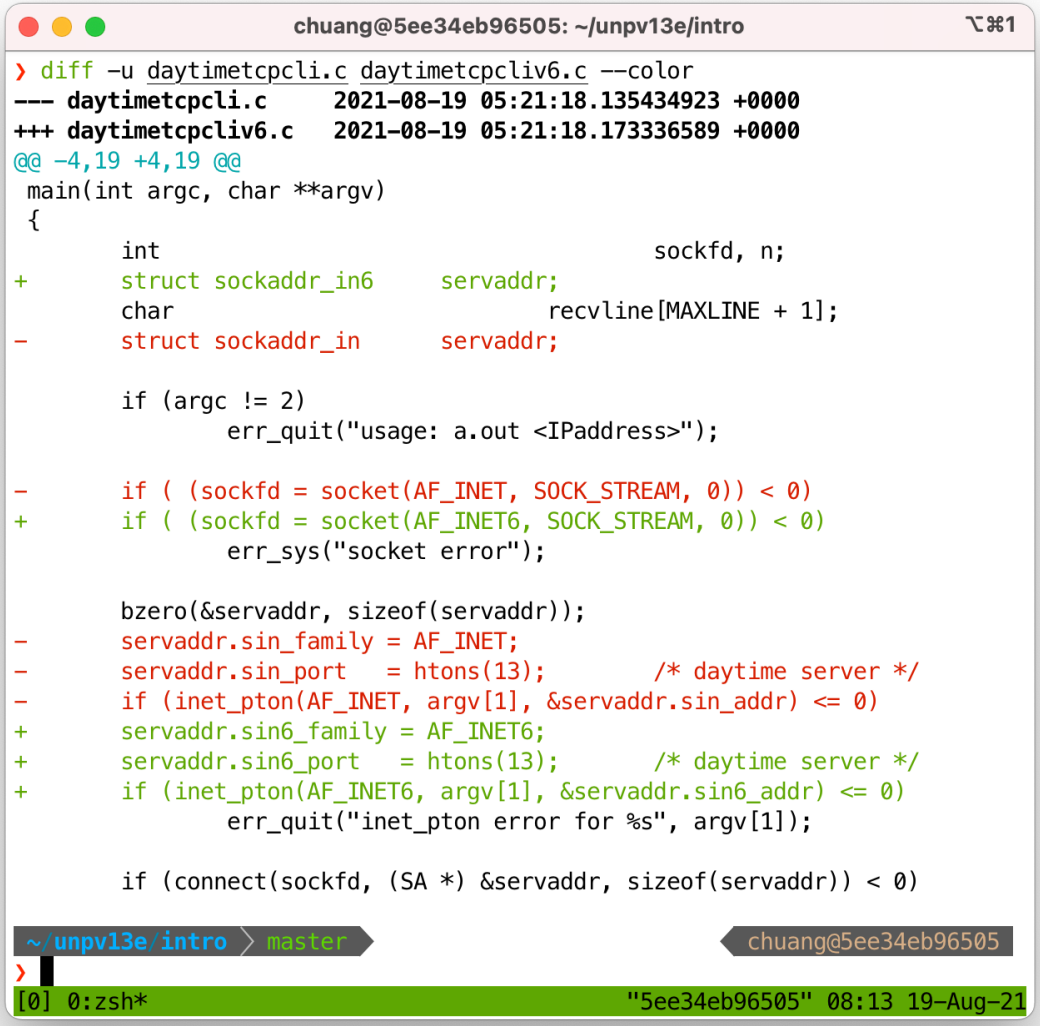
Key Points

- argv
- socket => Socket
- struct socketaddr_in
- bind => Bind
- listen => Listen
- (in an infinite loop)
- accept => Accept
- write => write
- close

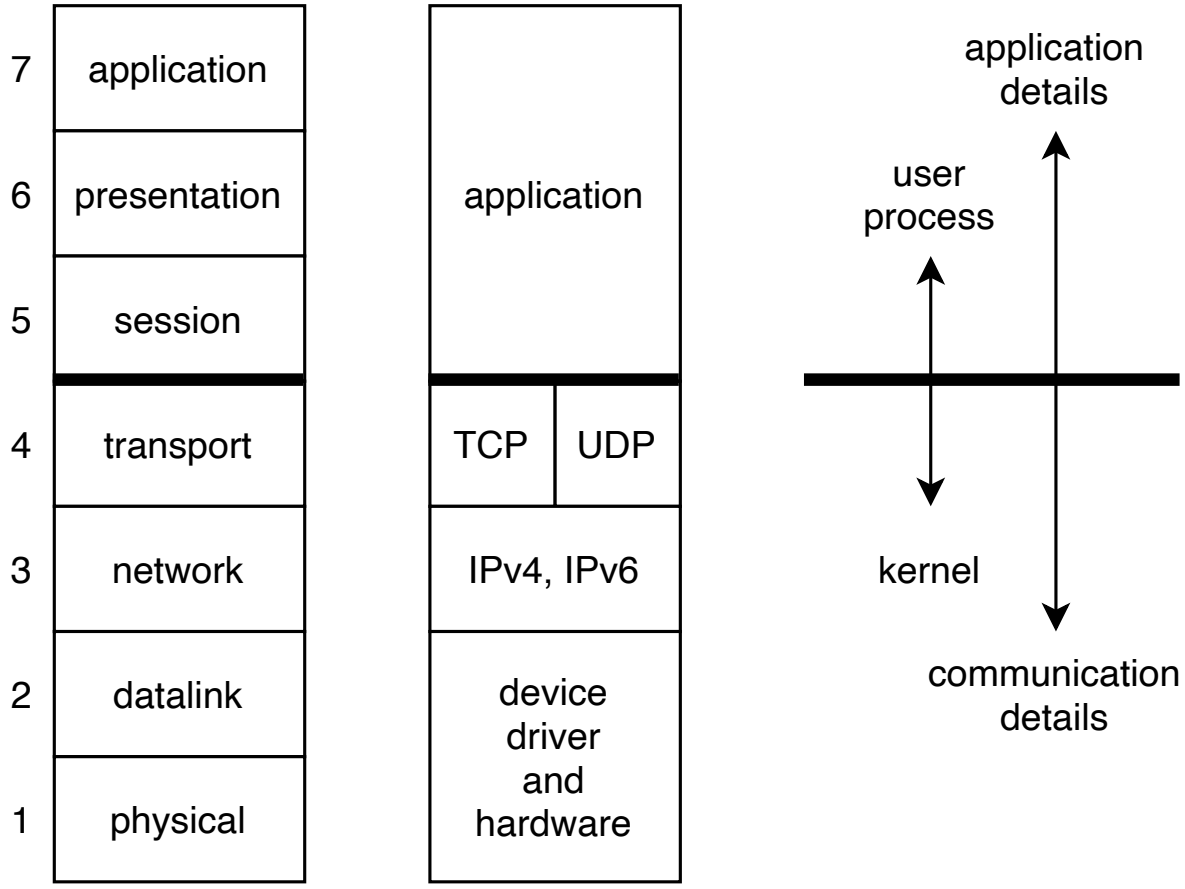
Variants of daytimecli/srv – Topics in the Textbook

- IPv4 / IPv6
- TCP / UDP
- Single process connection multiplexing
  - select / pool
  - epoll, kqueue
- Non-blocking I/O

Example: IPv4 vs IPv6



OSI Model


7
application
6
presentation
5
session
4
transport
3
network
2
datalink
1
physical
application
TCP
UDP
IPv4, IPv6
device
driver
and
hardware
application
details
user
process
kernel
communication
details

Network Relevant Commands and Tools

- netstat -ni | -nr | -na
- ifconfig -a | if-name
- (linux-specific) ip (link | addr) show
- ping
- traceroute
- nc
- telnet (in inetutils package)

Network Relevant Commands and Tools (Cont'd)

- nmap
- tcpdump / (gui) wireshark
- sniffit
- mitm-proxy

Q & A