KWANGWOON UNIVERSITY

Network Programming
2019 Seminar
01. introduction

01. Introduction

- 1.1 Introduction
- 1.2 A Simple Daytime Client
- 1.3 Protocol Independence
- 1.4 Error Handling : Wrapper functions
- 1.5 A Simple Daytime Server
- 1.7 OSI Model
- 1.8 BSD Networking History
- 1.9 Test Networks and Hosts
- 1.10 Unix Standard
- 1.11 64-Bit Architectures

1.1. Introduction

- **Client / Server**

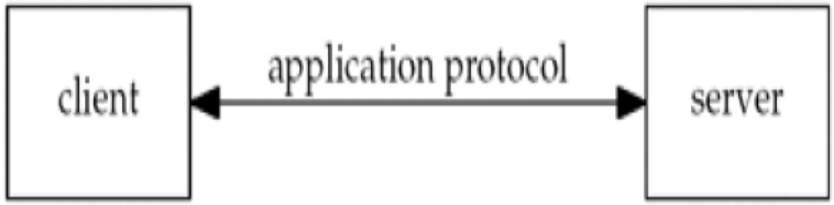


Figure 1.1 Network application : client and server

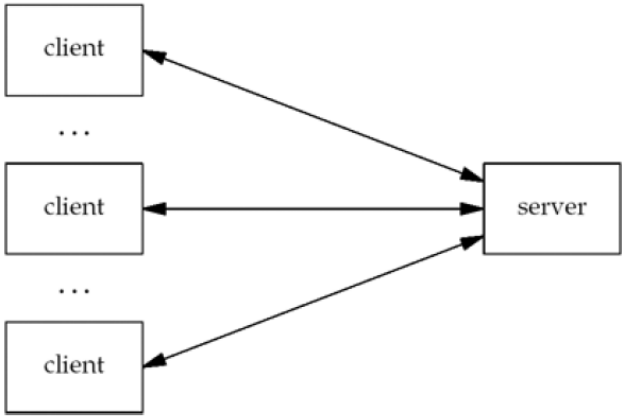


Figure 1.2 Server handling multiple clients at the same time

- Client가 Server에 접속하여 Server는 그 접속을 연결시켜 준다.
- Example : Telnet, ftp Client, http Server
- 주로 TCP/IP Protocol을 다룬다.

1.1. Introduction

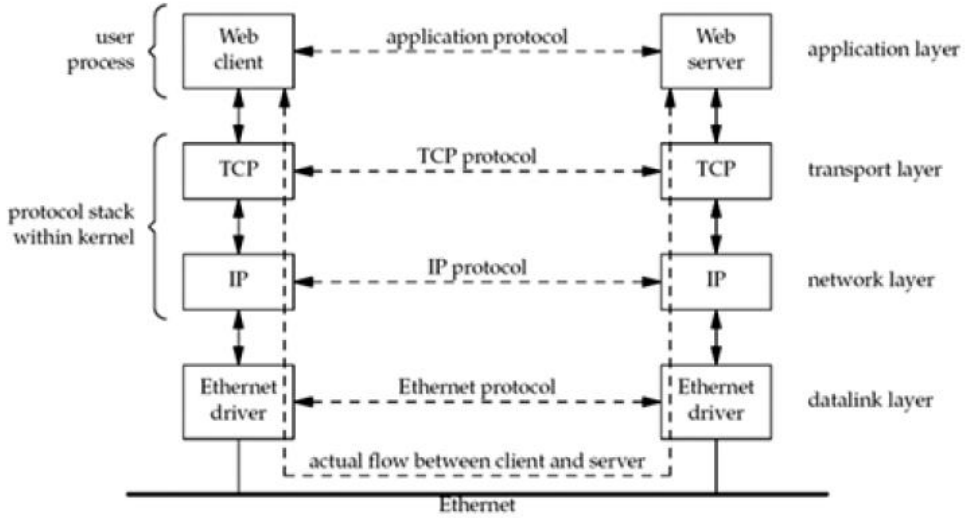


Figure 1.3 Client and server on the same Ethernet communicating using TCP

- Actual flow of information between client and server
- The client and server are typically user processes
- IP : IPv4(1980s), IPv6(1990s)

1.1. Introduction

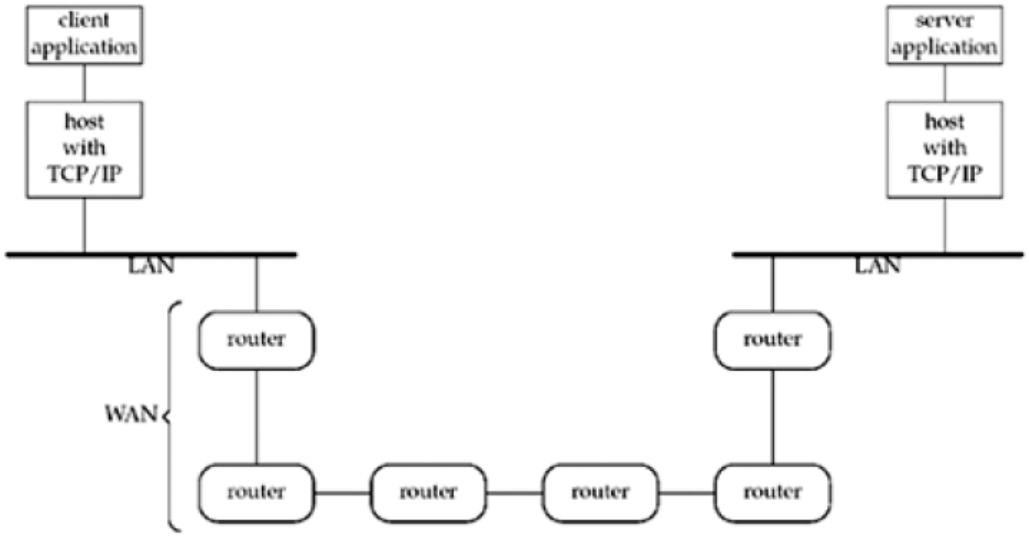


Figure 1.4 Client and server on different LANS connected through a WAN

- Routers are the building blocks of WANS
- The largest WAN's Example : Internet

1.2. A Simple Daytime Client

- **TCP Daytime Client**

```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);
}
```

```
jsh0116@jsh0116-VirtualBox:~/unpv13e/intro$ ./daytimetcpcli 203.253.70.30
30 JUN 2019 18:58:00 KST
```

- 1: Include our own header
- 2~3: command-line arguments
- 10~11 : Create TCP socket
- 12~16 Specify server's IP address and port
- 17~18 : Establish connection with server
- 19~25 Read and display server's reply
- 26 : Terminate Program

1.3. Protocol Independence

- **TCP daytime client for IPv6**

```c
#include        "unp.h"

int
main(int argc, char **argv)
{
        int                                     sockfd, n;
        struct sockaddr_in6     servaddr;
        char                                    recvline[MAXLINE + 1];

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

        if ( (sockfd = socket(AF_INET6, SOCK_STREAM, 0)) < 0)
                err_sys("socket error");

        bzero(&servaddr, sizeof(servaddr));
        servaddr.sin6_family = AF_INET6;
        servaddr.sin6_port   = htons(13);       /* daytime server */
        if (inet_pton(AF_INET6, argv[1], &servaddr.sin6_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);
}
```

- It is dependent on IPv6
- It is better to make a program protocol-independent.

1.4 Error Handling : Wrapper Functions

- **Our Wrapper function for the socket function**

```
jsh0116@jsh0116-VirtualBox:~/unpv13e/lib$ vim wrapsock.c
```

```
/* include Socket */
int
Socket(int family, int type, int protocol)
{
        int             n;

        if ( (n = socket(family, type, protocol)) < 0)
                err_sys("socket error");
        return(n);
}
/* end Socket */
```

- 오류 처리를 위해 Wrapper Function을 사용하여 프로그램을 좀 더 짧게 만들 수 있다.
- 예를 들어 Socket함수를 Wrapping하는 Socket함수로 다음과 같이 사용한다.

```
sockfd = Socket (AF_INET, SOCK_STREAM, 0);
```

- 대문자로 시작하는 함수들은 대부분 Wrapper함수를 의미한다.

1.4 Error Handling : Wrapper Functions

- **Our Wrapper function for pthread_mutex_lock**

```
jsh0116@jsh0116-VirtualBox:~/unpv13e/lib$ vim wrappthread.c
```

```
/* include Pthread_mutex_lock */
void
Pthread_mutex_lock(pthread_mutex_t *mptr)
{
        int             n;

        if ( (n = pthread_mutex_lock(mptr)) == 0)
                return;
        errno = n;
        err_sys("pthread_mutex_lock error");
}
/* end Pthread_mutex_lock */
```

- Thread함수를 사용할 때 반드시 한 변수를 할당하여야 한다.
- Err_sys()은 Errno 전역 변수의 값에 따른 메시지를 보여준다.
- Errno와 Err_sys() 호출을 하나로 결합

1.5 A Simple Daytime Server

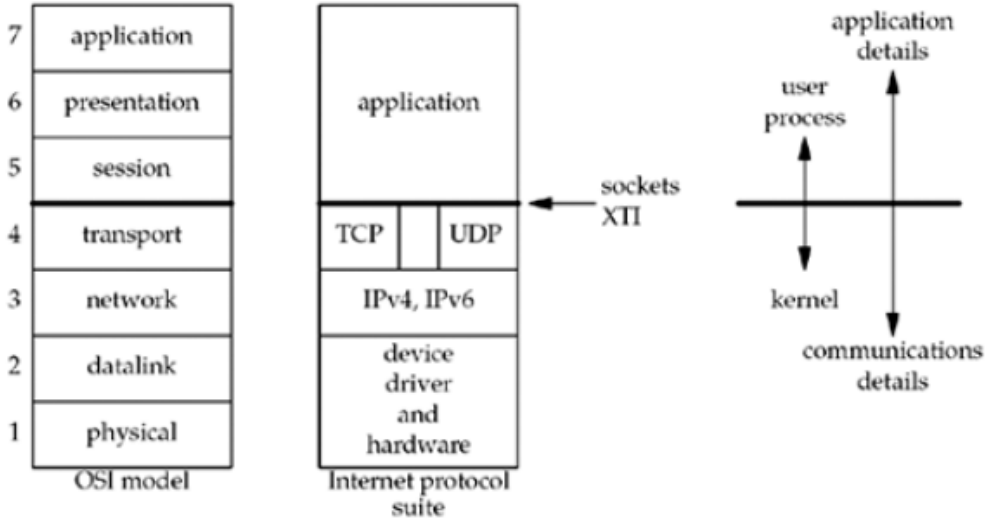
- **TCP Daytime Server : Daytime Client와 함께 구동**

```
#include        "unp.h"
#include        <time.h>

int
main(int argc, char **argv)
{
        int                                     listenfd, connfd;
        struct sockaddr_in      servaddr;
        char                            buff[MAXLINE];
        time_t                          ticks;

        listenfd = Socket(AF_INET, SOCK_STREAM, 0);

        bzero(&servaddr, sizeof(servaddr));
        servaddr.sin_family      = AF_INET;
        servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
        servaddr.sin_port        = htons(13);   /* daytime server */

        Bind(listenfd, (SA *) &servaddr, sizeof(servaddr));

        Listen(listenfd, LISTENQ);

        for ( ; ; ) {
                connfd = Accept(listenfd, (SA *) NULL, NULL);

        ticks = time(NULL);
        snprintf(buff, sizeof(buff), "%.24s\r\n", ctime(&ticks));
        Write(connfd, buff, strlen(buff));

                Close(connfd);
        }
}
```

- 크게 Socket, Bind, Listen 3단계
- 10 : TCP Socket 생성
- 11~15 : Bind()를 호출하여 Socket Address 구조체를 채우고 Server의 Well-known port에 연결한다.
- 16: Listen()를 호출하여 Listen Socket으로 변경.
- 17~21 : Client의 접속 accept와 reply

1.7 OSI Model

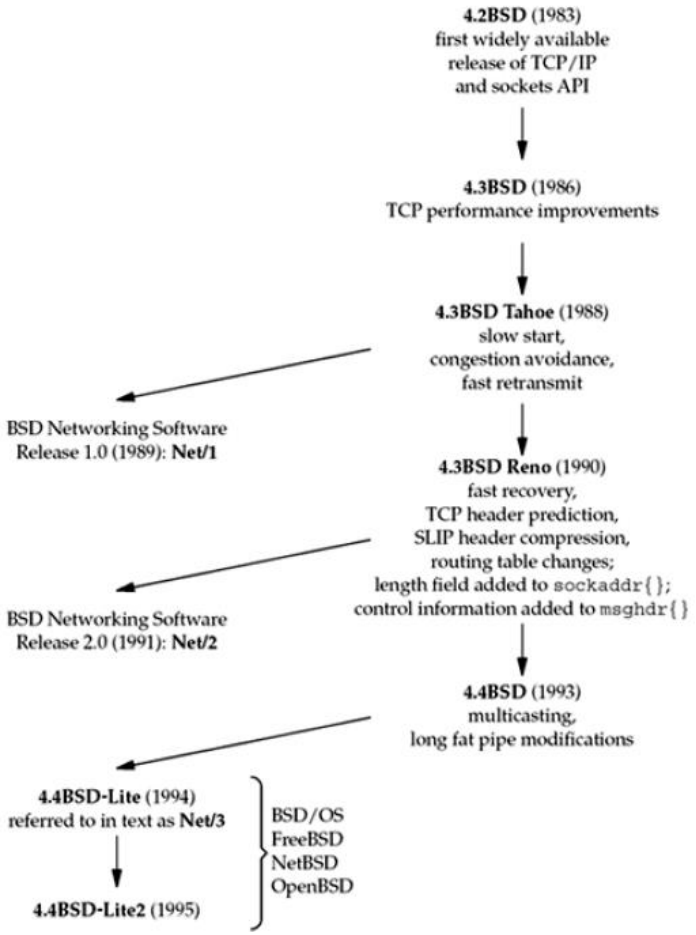
- **Layers in OSI model and Internet protocol suite**



- First, the upper three layers handle all the details of the application and the lower four layers handle all the communication details.
- Second, the upper three layer often form what is called a user process while the lower four layers are normally provided as part of the OS kernel.

1.8 BSD Networking History

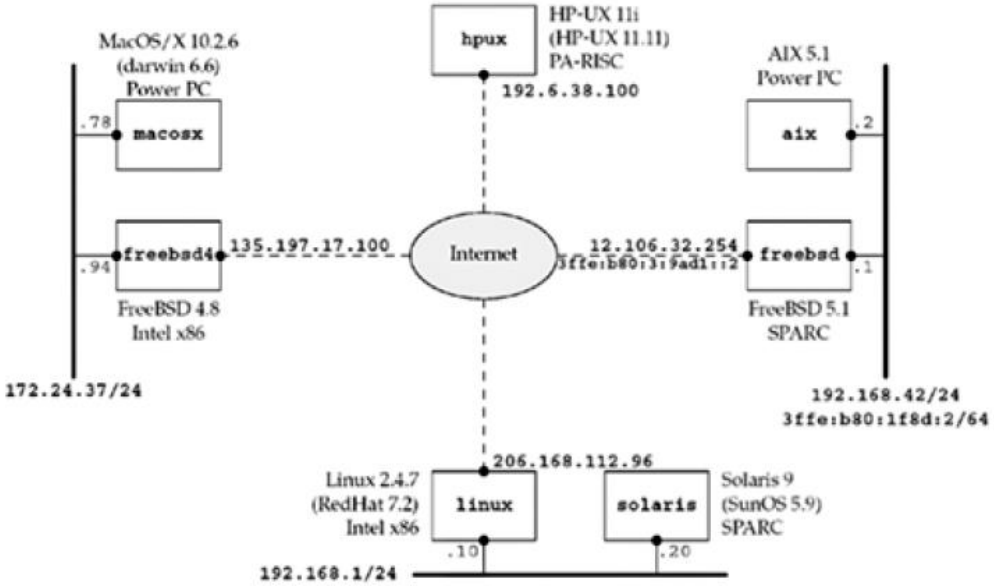
- **History of various BSD releases**



- Berkeley Software Distribution
- Berkeley CSRG에서 개발한 Unix OS
- 4.3BSD(1986), 4.4BSD(1990)
- 4.4BSD_Lite (AT&T 소스 미포함)
- 4.4BSD_Lite2
- 주요 BSD 파생 OS

1.9 Test Networks and Hosts

- **Networks and hosts used for most examples in the text**



- 각각의 Host에 대해 OS 및 Hardware type을 표시.
- 상자의 이름은 text에 나타나는 Host 이름.
- "/24" Notation : Network와 Subnet을 확인하는데 사용되는 Address의 leftmost bit부터 시작하는 연속 bit 수

1.9 Test Networks and Hosts

- **Discovering Network Topology**
  - netstat

  1) Netstat –I : provide information on the interfaces.
  2) Netstat –ni : provide information representing Number on the interfaces.
  3) Netstat –r : Routing Table
  4) Netstat –nr : provide information representing Number on the interfaces

     ex) defaul가 0.0.0.0으로 change

  - ifconfig

  1) ifconfig eth() : provide information of the eth() interfaces.
  2) Ifconfig –a : all of interface information provide

  - ping :

  1) Ping –b <Broadcasting Address>을 사용하여 Network 내의 IP Address를 알 수 있다.

1.10 Unix Standards

- **POSIX**
  - Portable Operating System Interface
  - A family of standards being developed by IEEE
- **The Open Group**
  - an international consortium of vendors and end-user customers from industry, government, and academia.
- **IETF (Internet Engineering Task Force)**
  - a large open international community of network deoperators, vendors, and researchers concerned with evolution of the Internet architecture and the smooth operation of the internet.

1.11 64-Bit Architecture

- **Comparison of number of bits to hold various datatypes for the ILP32 and LP54 models**

Datatype	ILP32 model	LP64 model
char	8	8
short	16	16
int	32	32
long	32	64
pointer	32	64

- ILP32 : 기존 32-Bit Unix System을 위한 Programming Language Model
- LP64 : 64-Bit Unix System Model,
  long과 Pointer 자료형의 데이터 크기가 각각 64Bit이다.

참고문헌

https://notes.shichao.io/unp/ch1/
https://eunguru.tistory.com/95?category=667830
https://www.unpbook.com/srcl/html **(unp.h Download link)**
https://blog.naver.com/grram/140007059985
https://slideplayer.com/slide/6812566/

Thank You
Q&A