

Bridge vs NAT

- Bridge Host와 동일한 네트워크 주소로 Guest B 할당하는 것.
- NAT Host에서 열어준 사설망으로 Host와는 다른 네트워크 주소로 Guest A 할당하는 것.
- 다른 호스트에서 Guest B 접근은 가능하지만 A 접근 불가
- Guest A -> B 가능, B -> A 불가 --> 포트 포워딩 필요!!

포트 포워딩

Host와 네트워크 주소가 다른 Guest에 도달하기 위해 포트 번호로 구별. -> Host에 전달되는 패킷 중 포트번호가 ****면 Guest A에게 전달하는 방식..

NAT

local

10.0.0.1 A
10.0.0.2 B
10.0.0.3 C

10.0.0.4
10.0.0.0?

138.76.29.7

R

Src: A IP, A port → R IP, 임의 port
dst: D IP, D port

→ 외부 가기 때문에 사설 IP 사용x, 공인 IP (138.76.29.7) 사용

Why?

IPv4: 2^{32} 사용가능
→ 작아..

WAN	LAN
138.76... 5001	10.0.0.1, 3345

임의 port

port번호 공인 ↔ 사설
→ 열어야 하는 것 필요

공인 IP: 전세계적으로 공유
사설 IP: 공유기에 공인 IP 할당후, 공유기에 연결된 컴퓨터에 사설 IP 할당.
공유하지 않기 때문에 외부와 통신시 공인 IP (라우터 IP) 사용.

] 공인 IP ↔ 사설 IP
→ 변환
라우터의 translation table 존재.

NAT (Network address translation) 장점

1. 하나의 공인 IP로 모든 네트워크에 사설 IP 할당 가능
2. $IPV4 = 2^{32}$ → 부족현상 해결 (다른 domain 내에서 사설 IP 잡혀도 됨)
3. 사설 IP 바우더라도 ISP (공인 Port) 에게 알릴 필요x
4. ISP를 바우더 내부까지 알릴 필요x
5. 내부 주소가 외부에 노출되지 않음.

6. 범위 확장성

LAN → WAN

ISP: Internet Service Provider

→ 인터넷 통신망 보유 & 공인 IP 할당하는 인터넷 회사
ex) KT, SK, LG U+ ..

UDP 기본

UDP - 호스트 안에서 프로세스 식별을 통한 데이터 전송

- 신뢰성 X
- 순차 전달, 혼잡 제어, 흐름 제어 X

UDP 헤더

- src port
- dst port
- length
- checksum

UDP 특징

- connect() 불필요
- 신뢰적이지 않기에 응용 프로그램 수준에서 신뢰성 있는 데이터 전송 기능 구현 필요
- 다자간 통신 쉽게 구현

UDP 서버/클라이언트 분석

UDP Socket

```
socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP / 0)
```

UDP 데이터 전송

- sendto()

▶ 실습 01: sendto.c

```
1  #include <stdio.h>
2  #include <sys/socket.h>
3  #include <sys/types.h>
4  #include <netinet/in.h>
5  #include <string.h>
6  #include <errno.h>
7  #include <stdlib.h>
8
9
10 void errorProc(const char*);
11 int main(int argc, char** argv)
12 {
13     int mySock, readLen, nSent;
14     char buff[BUFSIZ];
15     struct sockaddr_in destAddr;
16     socklen_t addrLen;
17
18     mySock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
19
20     memset(&destAddr, 0, sizeof(destAddr));
21     destAddr.sin_addr.s_addr = inet_addr(argv[1]);
22     destAddr.sin_family = AF_INET;
```

```

23 //char to int
24 destAddr.sin_port = htons(atoi(argv[2]));
25 addrLen = sizeof(destAddr);
26 while(1)
27 {
28     //사용자 입력 받아서 전송
29     fgets(buff, BUFSIZ-1, stdin);
30     readLen = strlen(buff);
31     nSent = sendto(mySock, buff, readLen, 0,
32         (struct sockaddr *) &destAddr,
33         addrLen);
34     printf("%d bytes were sent. \n",nSent);
35 }
36
37 close(mySock);
38 return 0;
39 }
40
41 void errProc(const char* str)
42 {
43     fprintf(stderr,"%s: %s \n", str, strerror(errno));
44     exit(1);
45 }
46

```

Colored by Color Scripter

- recvfrom()

▶ 실습 01: recvfrom.c

```

1  #include <stdio.h>
2  #include <sys/socket.h>
3  #include <sys/types.h>
4  #include <netinet/in.h>
5  #include <string.h>
6  #include <errno.h>
7  #include <stdlib.h>
8  void errProc(const char*);
9  int main(int argc, char** argv)
10 {
11     int mySock, readLen, nRecv, res;
12     char buff[BUFSIZ];
13     struct sockaddr_in srcAddr, destAddr;
14     socklen_t addrLen;
15
16     if(argc != 2) {
17         fprintf(stderr, "Usage: %s Port", argv[0]);
18         return 0;
19     }
20
21     //IPv4, UDP
22     mySock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
23     if(mySock == -1) errProc("socket");
24     memset(&srcAddr, 0, sizeof(srcAddr));
25     srcAddr.sin_addr.s_addr = htonl(INADDR_ANY);
26     srcAddr.sin_family = AF_INET;

```

```

27 //char to int
28 srcAddr.sin_port = htons(atoi(argv[1]));
29
30 res = bind(mySock, (struct sockaddr *) &srcAddr,
31             sizeof(srcAddr));
32 if(res == -1) errProc("bind");
33 addrLen = sizeof(destAddr);
34
35 while(1)
36 {
37     //받음
38     nRecv = recvfrom(mySock, buff, BUFSIZ-1, 0,
39                     (struct sockaddr *) &destAddr,
40                     &addrLen);
41     if(nRecv == -1) errProc("recvfrom");
42     printf("%d bytes were recv. \n", nRecv);
43 }
44
45 close(mySock);
46 return 0;
47 }
48
49 void errProc(const char* str)
50 {
51     fprintf(stderr, "%s: %s \n", str, strerror(errno));
52     exit(1);
53 }
54

```

Colored by Color Scriptor

1. 다중 수신 가능
2. recvfrom이 지켜있지 않더라고 데이터 전송 가능 -> 데이터 도달 여부 확인X
3. listen / connect를 하지 않음.

server

socket -> bind -> recvfrom -> sendto -> closesocket

client

socket -> sendto -> recvfrom -> closesocket socket -> connect -> write -> read -> closesocket

UDP 소켓의 특징

1. UDP 클라이언트 포트 번호 지정 bind() 이용하여 포트번호 지정 가능 사용하지 않으면 sendto()에 의해 임의로 포트번호 지정됨
2. 다수의 클라이언트 처리 recvfrom을 통해 송신자의 주소를 가져올 수 있음 sendto는 포트번호만 달리하여 여러 목적지로 데이터 전송 가능
3. connect()? UDP 소켓에서 sendto / recvfrom 호출하면 커널과 연결된다. -> 오버헤드 발생
UDP에 connect 사용하면 read / write 사용가능 -> 오버헤드 적어짐
하나의 UDP 소켓으로 여러 프로세스들과 통신 가능하다는 장점은 사라짐

▶ 실습 02: echoServer.c

```
1  #include <stdio.h>
2  #include <sys/socket.h>
3  #include <sys/types.h>
4  #include <netinet/in.h>
5  #include <string.h>
6  #include <errno.h>
7  #include <stdlib.h>
8  #include <arpa/inet.h>
9
10 void errProc(const char*);
11 int main(int argc, char** argv)
12 {
13     int mySock, readLen, nRecv, res;
14     char buff[BUFSIZ];
15     char * strAddr;
16     struct sockaddr_in srcAddr, destAddr;
17     socklen_t addrLen;
18
19     if(argc != 2) {
20         fprintf(stderr, "Usage: %s Port", argv[0]);
21         return 0;
22     }
23
24     //UDP Socket, 서버에 접속한 노드의 정보
25     mySock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
26     if(mySock == -1) errProc("socket");
27
28     //IP 아무거나, Port번호만 입력받음
29     memset(&srcAddr, 0, sizeof(srcAddr));
30     srcAddr.sin_addr.s_addr = htonl(INADDR_ANY);
31     srcAddr.sin_family = AF_INET;
32     srcAddr.sin_port = htons(atoi(argv[1]));
33
34     //bind를 통해 포트 번호 할당
35     res = bind(mySock, (struct sockaddr *) &srcAddr,
36               sizeof(srcAddr));
37     if(res == -1) errProc("bind");
38     addrLen = sizeof(destAddr);
39
40     while(1)
41     {
42         //접속 노드로부터 수신
43         nRecv = recvfrom(mySock, buff, BUFSIZ-1, 0,
44                          (struct sockaddr *) &destAddr,
45                          &addrLen);
46         if(nRecv == -1) errProc("recvfrom");
47         if(nRecv > 0) buff[nRecv-1] = '\0';
48         else buff[nRecv] = '\0';
49
50         //접속한 노드의 주소
51         strAddr = inet_ntoa(destAddr.sin_addr);
52         //접속한 노드의 IP: Port > data
53         printf("%s:%d>%s\n", strAddr, ntohs(destAddr.sin_port), buff);
54         nRecv = strlen(buff);
55
56         //접속한 노드에게 그대로 송신
57         sendto(mySock, buff, nRecv, 0,
```

```

58         (struct sockaddr *) &destAddr, addrLen);
59     }
60     close(mySock);
61     return 0;
62 }
63
64 void errProc(const char* str)
65 {
66     fprintf(stderr, "%s: %s \n", str, strerror(errno));
67     exit(1);
68 }
69

```

Colored by Color Scripter

▶ 실습 02: echoClient.c

```

1  #include <stdio.h>
2  #include <sys/socket.h>
3  #include <sys/types.h>
4  #include <netinet/in.h>
5  #include <string.h>
6  #include <errno.h>
7  #include <stdlib.h>
8  #include <arpa/inet.h>
9
10 void errProc(const char*);
11 int main(int argc, char** argv)
12 {
13     int mySock, readLen, nSent, nRecv;
14     char buff[BUFSIZ];
15     char strAddr;
16     struct sockaddr_in destAddr;
17     socklen_t addrLen;
18
19     mySock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
20
21     memset(&destAddr, 0, sizeof(destAddr));
22     destAddr.sin_addr.s_addr = inet_addr(argv[1]);
23     destAddr.sin_family = AF_INET;
24     destAddr.sin_port = htons(atoi(argv[2]));
25     addrLen = sizeof(destAddr);
26
27     while(1)
28     {
29         //사용자 입력
30         fgets(buff, BUFSIZ-1, stdin);
31         readLen = strlen(buff);
32
33         //서버에 보내기
34         nSent = sendto(mySock, buff, readLen, 0,
35             (struct sockaddr*) &destAddr, addrLen);
36         if(nSent == -1) errProc("write");
37
38         //서버로부터 받기
39         nRecv = recvfrom(mySock, buff, BUFSIZ-1, 0,
40             (struct sockaddr*) &destAddr, &addrLen);
41         if(nRecv == -1) errProc("read");
42         buff[nRecv] = '\0';

```

```

43     printf("Server: %s\n", buff);
44     if(!strcmp(buff, "END")) break;
45 }
46 close(mySock);
47 return 0;
48 }
49
50 void errProc(const char* str)
51 {
52     fprintf(stderr, "%s: %s\n", str, strerror(errno));
53     exit(1);
54 }
55

```

Colored by Color Scripter

echo Server & Client

- 다중 통신 가능
- 서버 안 열려도 메시지 보내짐
- srv, clnt socket 이런거 없고 mysocket으로 접속한 노드(프로세스)의 정보를 저장함

▶ 실습 01: connectedSend.c

```

1  #include <stdio.h>
2  #include <sys/socket.h>
3  #include <sys/types.h>
4  #include <netinet/in.h>
5  #include <string.h>
6  #include <errno.h>
7  #include <stdlib.h>
8  #include <arpa/inet.h>
9
10 void errProc(const char*);
11 int main(int argc, char** argv)
12 {
13     int mySock, readLen, nSent, nRecv;
14     char buff[BUFSIZ];
15     char strAddr;
16     struct sockaddr_in destAddr, destAddr2;
17     socklen_t addrLen;
18
19     //UDP Socket
20     mySock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
21
22     memset(&destAddr, 0, sizeof(destAddr));
23     destAddr.sin_addr.s_addr = inet_addr(argv[1]);
24     destAddr.sin_family = AF_INET;
25     destAddr.sin_port = htons(atoi(argv[2]));
26
27     memset(&destAddr2, 0, sizeof(destAddr2));
28     destAddr2.sin_addr.s_addr = inet_addr(argv[1]);
29     destAddr2.sin_family = AF_INET;
30     destAddr2.sin_port = htons(9050);
31     addrLen = sizeof(destAddr2);
32
33     //connect 이용 -> 다중통신 불가 / read, write 가능
34     //접속한 노드와 연결

```

```

35     connect(mySock, (struct sockaddr*) &destAddr,
36             sizeof(destAddr));
37
38     while(1)
39     {
40         // connected UDP
41         fgets(buff, BUFSIZ-1, stdin);
42         readLen = strlen(buff);
43
44         //보내고
45         nSent = write(mySock, buff, readLen);
46         if(nSent == -1) errProc("write");
47         printf("%d bytes were sent. \n",nSent);
48
49         //읽고
50         nRecv = read(mySock, buff, BUFSIZ-1);
51         if(nRecv == -1) errProc("read");
52         buff[nRecv] = '\0';
53         printf("Server: %s\n", buff);
54
55         // sendto & recvfrom 가능
56         fgets(buff, BUFSIZ-1, stdin);
57         readLen = strlen(buff);
58         nSent = sendto(mySock, buff, readLen, 0,
59                       (struct sockaddr *) &destAddr2, addrLen);
60         if(nSent == -1) errProc("write");
61         printf("%d bytes were sent. \n",nSent);
62
63         nRecv = recvfrom(mySock, buff, BUFSIZ-1, 0,
64                          (struct sockaddr *) &destAddr2, &addrLen);
65         if(nRecv == -1) errProc("read");
66         buff[nRecv] = '\0';
67         strAddr = inet_ntoa(destAddr2.sin_addr);
68
69         printf("%s:%d >%s\n",strAddr,
70               ntohs(destAddr2.sin_port), buff);
71
72     }
73     close(mySock);
74     return 0;
75 }
76
77 void errProc(const char* str)
78 {
79     fprintf(stderr,"%s: %s \n", str, strerror(errno));
80     exit(1);
81 }
82

```

Colored by Color Scripter

UDP Client에 connect 이용시, 안 열려있는 서버 접근시 Destination Unreachable

UDP 패킷 모니터링

echo server - client 통신시

5	7.367926795	127.0.0.1	127.0.0.1	UDP	45	34468 → 9050	Len=3
6	7.368003377	127.0.0.1	127.0.0.1	UDP	44	9050 → 34468	Len=2
7	14.800288287	127.0.0.1	127.0.0.1	UDP	48	34468 → 9050	Len=6
8	14.800319389	127.0.0.1	127.0.0.1	UDP	47	9050 → 34468	Len=5
9	18.752315389	127.0.0.1	127.0.0.1	UDP	46	34468 → 9050	Len=4
10	18.752346382	127.0.0.1	127.0.0.1	UDP	45	9050 → 34468	Len=3
11	20.840206350	127.0.0.1	127.0.0.1	UDP	46	34468 → 9050	Len=4
12	20.840237157	127.0.0.1	127.0.0.1	UDP	45	9050 → 34468	Len=3

▶ Frame 5: 45 bytes on wire (360 bits), 45 bytes captured (360 bits) on interface 0			
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)			
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1			
▼	User Datagram Protocol, Src Port: 34468, Dst Port: 9050		
	Source Port: 34468		
	Destination Port: 9050		
	Length: 11		
	Checksum: 0xfe1e [unverified]		
	[Checksum Status: Unverified]		
	[Stream index: 1]		
▼	Data (3 bytes)		
	Data: 48690a		
	[Length: 3]		

0000	00 00 00 00 00 00 00 00	00 00 00 00 08 00 45 00E.
0010	00 1f 27 4a 40 00 40 11	15 82 7f 00 00 01 7f 00	..J0.0.....
0020	00 01 86 a4 23 5a 00 00	fe 1e 48 69 0a	...#Z...Hi.

별다른 연결 / 해제 과정 없이 진행

Summary

- Bridge vs NAT
- UDP 기본
- UDP 서버 / 클라이언트 분석
 - sendto, recvfrom
 - 비연결 지향형
- UDP 소켓 특징
 - Unreachable packet
 - connect 지원 -> write, read 사용가능
- UDP 패킷 모니터링