Ch14-Multicast & Broadcast

Instructor: Limei Peng
Dept. of CSE, KNU

Outlines

- What are Multicast & Broadcast?
- How to send/receive broadcast and multicast packets?

Types of IP addresses

- IP protocol supports
  - Unicast : 1 to 1
  - Broadcast : 1 to all
  - Multicast : 1 to many

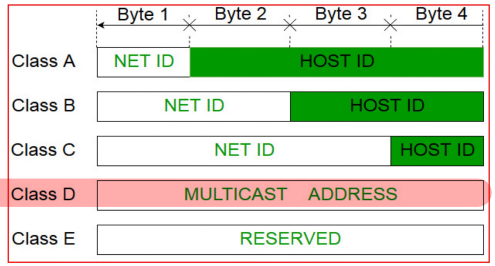
- Unicast
  - Identify one host
- Broadcast
  - Identify all hosts
- Multicast
  - Identify a set of hosts

Multicast

한번의 통신으로 목표한 여러 컴퓨터에 동시 전송

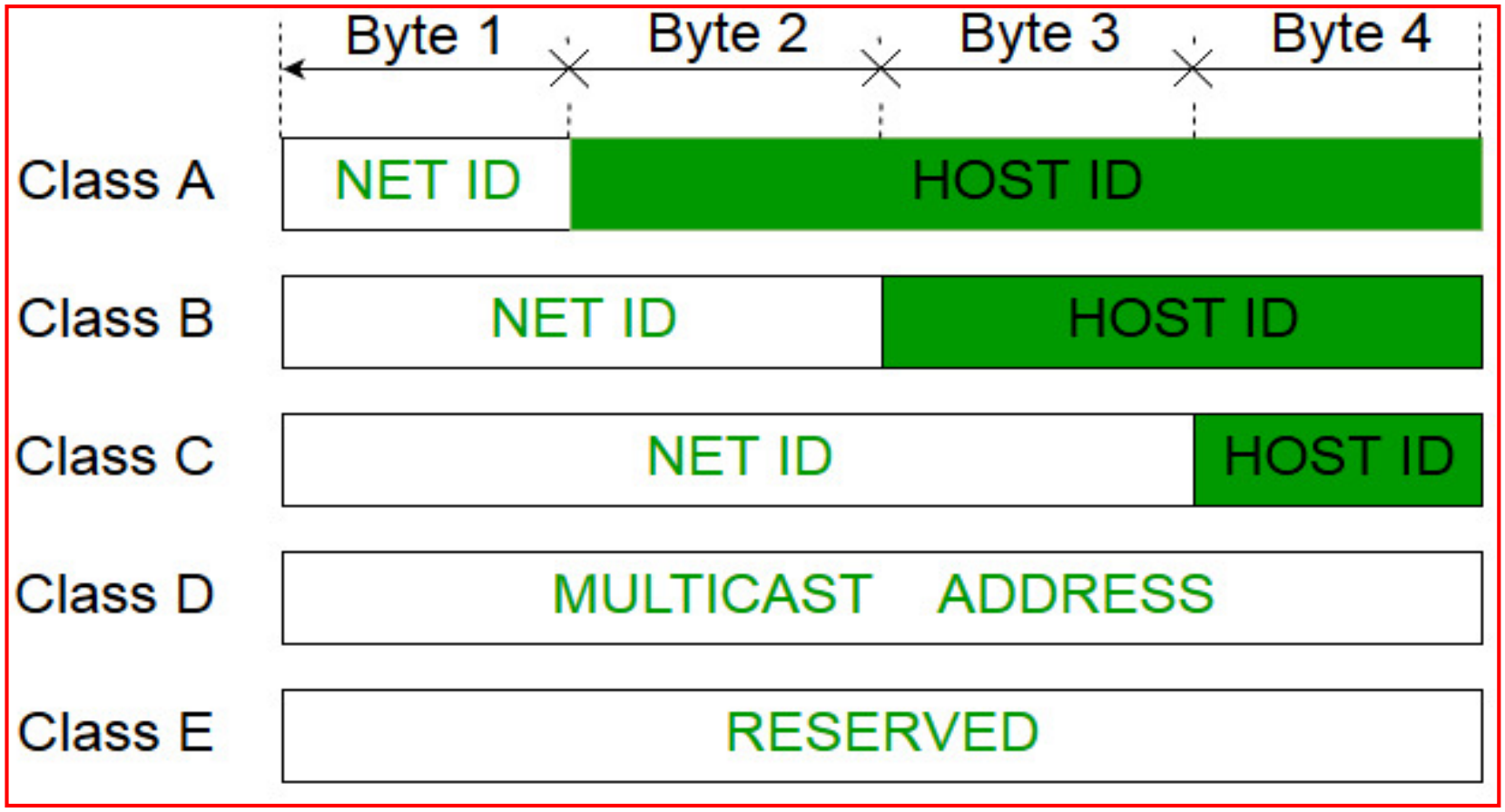
- Multicast address
  - Can be used for ***group*** communications over the Internet
  - A sender sends a message to a ***group*** of receivers
  - Multicast is always based on **UDP**
  - Every node that wants to receive multicast message must register with the Internet Group Management Protocol (***IGMP***) : multicast message를 받으려는 모든 노드는 IGMP에 등록해야함.
  - Class D addresses (224.0.0.0 to 239.255.255.255) are used for multicast gateways
    - Special purpose range: 244.0.0.0 to 244.0.0.255 (cannot be used and routers will not pass these messages on)

multicast의 예시용→



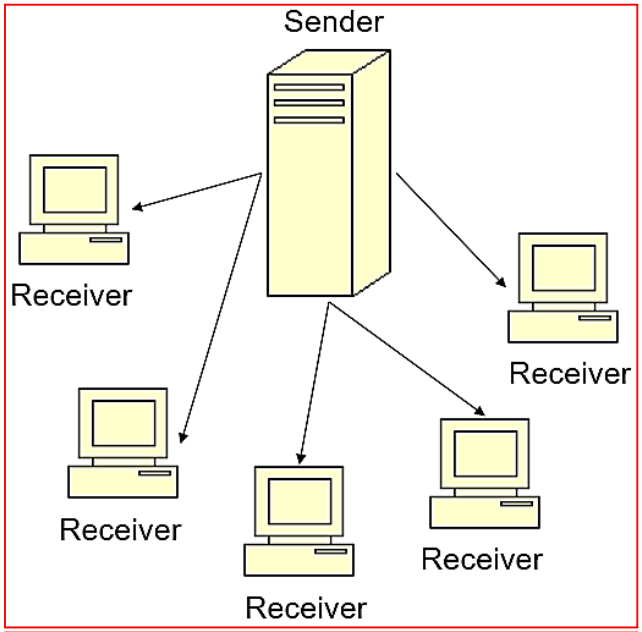
Multicast

- Multicast address: 224.0.0.0 to 239.255.255.255

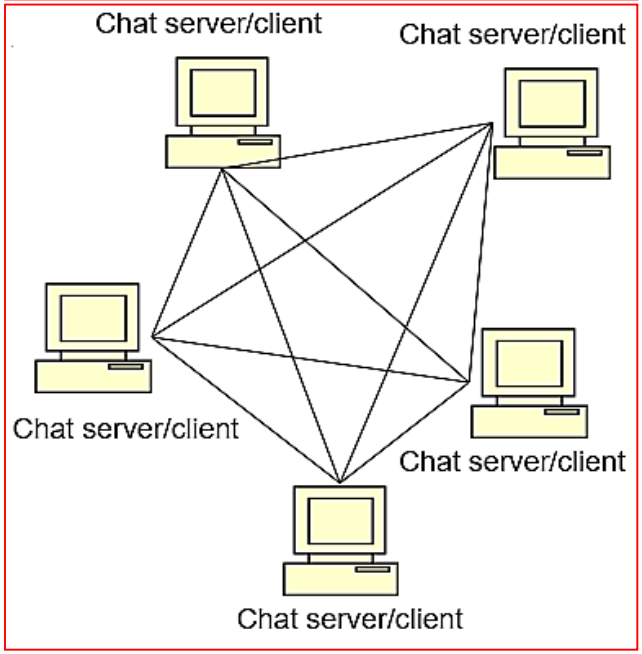


Application Models with Multicast

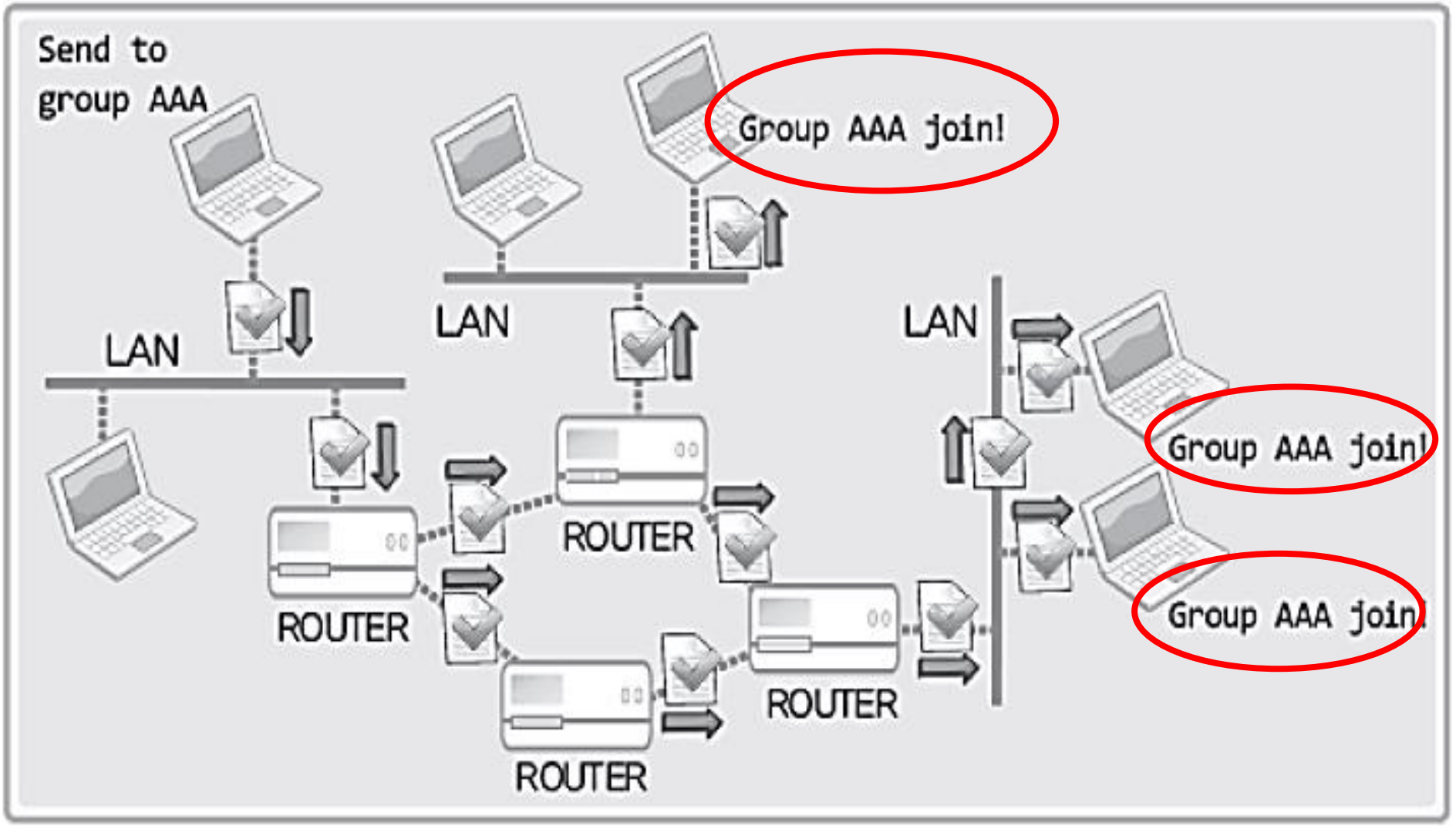
- One-to-many applications
  - One system may send a message to a group of system
  - This can be useful for video, audio or other large files/data types



- Many-to-Many applications
  - Allows every system in a group to send data to every other system in the group
  - For example: a peer-to-peer chat system (everyone in the group can see what everyone else is saying)

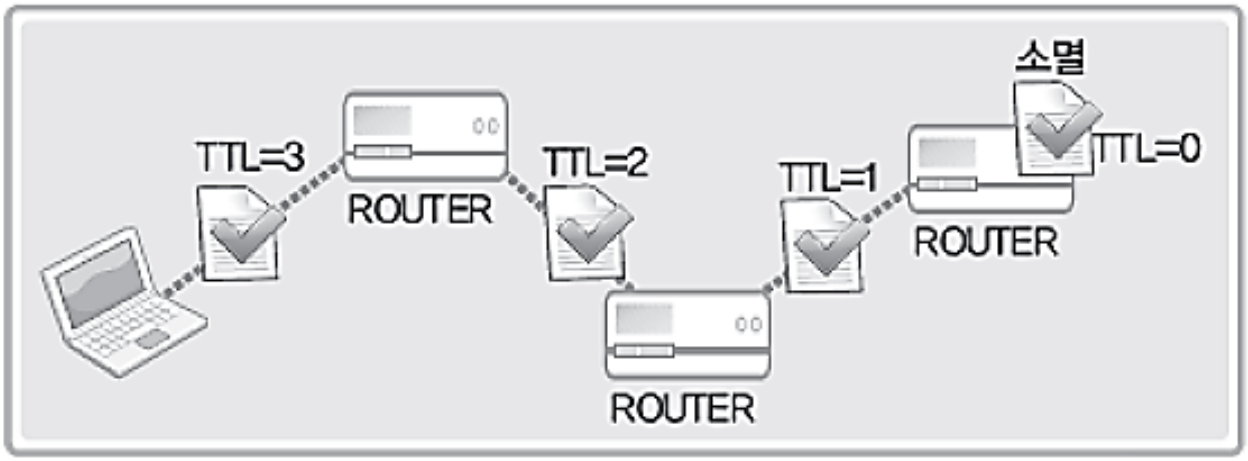


Multicast


Send to
group AAA
Group AAA join!
LAN
LAN
LAN
ROUTER
ROUTER
ROUTER
ROUTER
Group AAA join!
Group AAA join!

Multicast

- **TTL** (Time To Live) : 데이터의 유효기간을 나타내기위한 방법.
  - TTL: integer
  - It is used to imply **how far** the packet has been sent
    (패킷을 얼마나 멀리 보낼 것인가를 결정)
  - **Decreased** by **1** after bypassing each router
    (라우터를 하나 지날 때마다 1씩 감소)
    ↳ 라우터 네트워크 간 데이터 패킷을 전송하는 네트워크 장치
  - **TTL = 0**: the packet should be removed
    (TTL이 0이 되면, 해당 패킷은 소멸)



Multicast

- How to set TTL in multicast?
  - **setsockopt()**: ⇒ socket option의 변경을 위함.
    - Level associated with TTL: **IPPROTO_IP**
    - Option name: **IP_MULTICAST_TTL**

```
int send_sock;
int time_live=64;
. . . .
send_sock=socket(PF_INET, SOCK_DGRAM, 0);
setsockopt(send_sock, IPPROTO_IP, IP_MULTICAST_TTL, (void*)&time_live, sizeof(time_live));
. . . .
```

option

TTL을 64로 설정

이때 level의 socket정보를 가져올지 변경할지

SOL_SOCKET (Broadcast option) or IPPROTO_IP (multicast option)

Multicast

- How to join a multicast group?
  - **setsockopt()**
    - Protocol level: **IPPROTO_IP**
    - Option name: **IP_ADD_MEMBERSHIP**

```
struct ip_mreq
{
    struct in_addr imr_multiaddr;
    struct in_addr imr_interface;
}
```

```
int recv_sock;
struct ip_mreq join_adr;
. . . .
recv_sock=socket(PF_INET, SOCK_DGRAM, 0);
. . . .
join_adr.imr_multiaddr.s_addr=   "Multicast group address"
join_adr.imr_ interface.s_addr=  "Addresses of Hosts that will join the multicast group"

setsockopt(recv_sock, IPPROTO_IP, IP_ADD_MEMBERSHIP, (void*)&join_adr, sizeof(join_adr));
```

Multicast

```
struct ip_mreq {
 struct in_addr imr_multiaddr; (멀티캐스트 그룹정보)
 struct in_addr imr_interface; (자신의 인터페이스(IP주소))
}
```

- ***imr_multiaddr*** (중요!) `->` sender 정해진 값.
  - IP multicast address of group
- ***imr_interface*** (중요X)
  - Local IP address of interface

Example#1: multicast-sender.c

```c
/*news_sender.c*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#define TTL 64
#define BUF_SIZE 30

void error_handling(char* message);void error_handling(char* message)
{
        fputs(message, stderr);
        fputc('\n', stderr);
        exit(1);
}

int main(int argc, char* argv[])
{
        int send_sock;
        struct sockaddr_in mul_addr;
        int time_live = TTL;
        FILE* fp;
        char buf[BUF_SIZE];

        if(argc != 3)
        {
                printf("Usage : %s <Group IP> <PORT>\n", argv[0]);
                exit(1);
        }

        send_sock = socket(PF_INET, SOCK_DGRAM, 0);
        memset(&mul_addr, 0, sizeof(mul_addr));
        mul_addr.sin_family = AF_INET;
        mul_addr.sin_addr.s_addr = inet_addr(argv[1]); //multicast IP
        mul_addr.sin_port=htons(atoi(argv[2]));
        setsockopt(send_sock, IPPROTO_IP, IP_MULTICAST_TTL, (void*)&time_live, sizeof(time_live));

        if((fp = fopen("news.txt", "r")) == NULL)
                error_handling("fopen() error");


        while(!feof(fp))
        {
                fgets(buf, BUF_SIZE, fp);
                sendto(send_sock, buf, strlen(buf), 0, (struct sockaddr*)&mul_addr, sizeof(mul_addr));
                sleep(2);
        }

        close(send_sock);
        return 0;
}
```

Example#1: multicast-receiver.c

```c
/*receiver.c*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 30
void error_handling(char* message)
{
        fputs(message, stderr);
        fputc('\n', stderr);
        exit(1);
}
int main(int argc, char** argv)
{
        int recv_sock;
        int str_len;
        char buf[BUF_SIZE];
        struct sockaddr_in addr;
        struct ip_mreq join_addr;

        if(argc != 3)
           printf("Usage : %s <GroupIP> <PORT>\n", argv[0]);

        recv_sock = socket(PF_INET, SOCK_DGRAM, 0);
        memset(&addr, 0, sizeof(addr));
        addr.sin_family = AF_INET;
        addr.sin_addr.s_addr = htonl(INADDR_ANY);
        addr.sin_port = htons(atoi(argv[2]));

        if(bind(recv_sock, (struct sockaddr*)&addr, sizeof(addr)) == -1)
          error_handling("bind() error");

        join_addr.imr_multiaddr.s_addr = inet_addr(argv[1]);
        join_addr.imr_interface.s_addr = htonl(INADDR_ANY);

        setsockopt(recv_sock, IPPROTO_IP, IP_ADD_MEMBERSHIP, (void*)&join_addr, sizeof(join_addr));

        while(1)
        {
                str_len = recvfrom(recv_sock, buf, BUF_SIZE-1, 0, NULL, 0);
                if(str_len<0)
                        break;
                buf[str_len] = 0;
                fputs(buf, stdout);
        }

        close(recv_sock);
        return 0;
}
```

```
socket@ubuntu:~/Desktop/code$ cat news.txt
This is a file for testing multicasting......
socket@ubuntu:~/Desktop/code$ gcc news_sender.c -o multiS
socket@ubuntu:~/Desktop/code$ gcc news_receiver.c -o multiR
socket@ubuntu:~/Desktop/code$ ./multiS 224.0.0.8 9000
socket@ubuntu:~/Desktop/code$
```

Sender

```
socket@ubuntu:~/Desktop/code$ ./multiR 224.0.0.8 9000
This is a file for testing multicasting....
```

Receiver

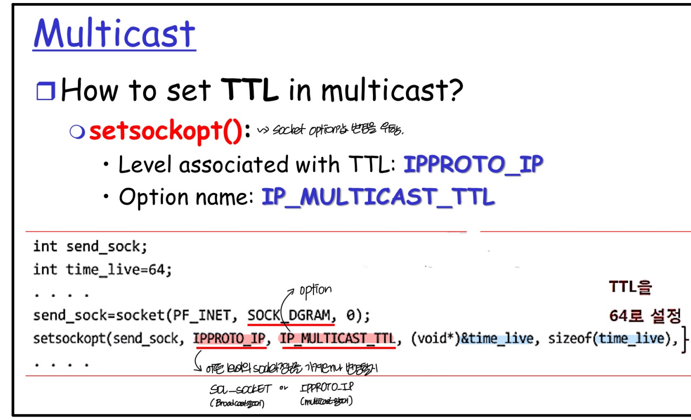
Broadcast

- Broadcast address
  - Broadcast datagrams are sent to all nodes in a subnetwork : 연결된 모든 장치로 정보통신에 사용되는 네트워크 주소
  - Broadcast is always based on *UDP*
    - Low overheads
    - No guarantee that message has been received : 메시지가 수신되었다는 보장 X
    - Broadcast don't cross subnets
  - Local broadcast
    - Local (255.255.255.255, which is reserved) means all hosts within the same network will receive the datagrams, but routers do not forward datagrams 같은 네트워크에 있는 모든 host가 datagram을 받음~. but router은 다른 network로 전달하지 X
  - Directed broadcast
    - Except the network part, the addresses for the host part are set all 1s and routers will forward the packets
    - E.g., if we have a subnet 10.10.10.0/24, we can ping from elsewhere in the network 10.10.10.255 and we would get an answer from all hosts

Broadcast

- How to set broadcast?
  - setsockopt()
    - Protocol level: **SOL_SOCKET**
    - Option name: **SO_BROADCAST**
      - set option value to 1

```
int send_sock;
int bcast=1;    // SO_BROADCAST의 옵션정보를 1로 변경하기 위한 변수 초기화
. . . .
send_sock=socket(PF_INET, SOCK_DGRAM, 0);
. . . .
setsockopt(send_sock, SOL_SOCKET, SO_BROADCAST, (void*)&bcast, sizeof(bcast));
. . . .
```



Multicast

- How to set **TTL** in multicast?
  - **setsockopt()**:
    - Level associated with TTL: **IPPROTO_IP**
    - Option name: **IP_MULTICAST_TTL**

```
int send_sock;
int time_live=64;
. . . .
send_sock=socket(PF_INET, SOCK_DGRAM, 0);
setsockopt(send_sock, IPPROTO_IP, IP_MULTICAST_TTL, (void*)&time_live, sizeof(time_live),
. . . .
```

TTL을 64로 설정

Example#2: broadcast-sender.c

```c
/*news_sender_brd.c*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<arpa/inet.h>
#include<sys/socket.h>

#define BUF_SIZE 30
void error_handling(char *message)
{
  fputs(message,stderr);
  fputc('\n',stderr);
  exit(1);
}

int main(int argc, char* argv[])
{
  int send_sock;
  struct sockaddr_in broad_adr;
  FILE *fp;
  char buf[BUF_SIZE];
  int so_brd =1;

  if(argc!=3)
  {
    printf("Usage: %s<Broadcast IP><Port>\n",argv[0]);
    exit(1);
  }

  send_sock = socket(PF_INET, SOCK_DGRAM, 0);
  memset(&broad_adr, 0, sizeof(broad_adr));
  broad_adr.sin_family = AF_INET;
  broad_adr.sin_addr.s_addr = inet_addr(argv[1]);
  broad_adr.sin_port = htons(atoi(argv[2]));

  setsockopt(send_sock, SOL_SOCKET, SO_BROADCAST,(void*)&so_brd,sizeof(so_brd));

  if((fp = fopen("news.txt","r"))==NULL)
    error_handling("fopen() error");

  while(!feof(fp))
  {
    fgets(buf, BUF_SIZE, fp);
    sendto(send_sock, buf, strlen(buf),0,(struct sockaddr*)&broad_adr,sizeof(broad_adr));
    sleep(2);
  }
  close(send_sock);
  return 0;
}
```

Example#2: broadcast-receiver.c

```c
/*news_receiver_brd.c*/
#include <stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<arpa/inet.h>
#include<sys/socket.h>

#define BUF_SIZE 30

void error_handling(char *message)
{
  fputs(message, stderr);
  fputc('\n',stderr);
  exit(1);
}

int main(int argc, char* argv[])
{
  int recv_sock;
  struct sockaddr_in adr;
  int str_len;
  char buf[BUF_SIZE];

  if(argc!=2)
  {
    printf("Usage: %s<port>\n", argv[0]);
    exit(1);
  }

  recv_sock = socket(PF_INET, SOCK_DGRAM, 0);
  memset(&adr,0,sizeof(adr));
  adr.sin_family = AF_INET;
  adr.sin_addr.s_addr = htonl(INADDR_ANY);
  adr.sin_port = htons(atoi(argv[1]));

  if(bind(recv_sock,(struct sockaddr*)&adr,sizeof(adr))==-1)
     error_handling("bind() error");

  while(1)
  {
     str_len = recvfrom(recv_sock, buf,BUF_SIZE-1,0,NULL,0);
     if(str_len < 0)
        break;
      buf[str_len] = 0;
      fputs(buf,stdout);
  }
  close(recv_sock);
  return 0;
}
```

```
socket@ubuntu:~/Desktop/code$ gcc news_receiver_brd.c -o receiverB
socket@ubuntu:~/Desktop/code$ gcc news_sender_brd.c -o senderB
socket@ubuntu:~/Desktop/code$ ./senderB 255.255.255.255 9190
socket@ubuntu:~/Desktop/code$ cat news.txt
This is a test news.
socket@ubuntu:~/Desktop/code$
```

Sender

```
socket@ubuntu:~/Desktop/code$ ./receiverB 9190
This is a test news.
```

Receiver

Auxiliary : setsockopt()

```
#include <sys/socket.h>

int setsockopt(int sock, int level, int optname, const void *optval, socklen_t optlen);
```

➔ 성공 시 0, 실패 시 −1 반환

- **sock** 옵션변경을 위한 소켓의 파일 디스크립터 전달.
- **level** 변경할 옵션의 프로토콜 레벨 전달.
- **optname** 변경할 옵션의 이름 전달.
- **optval** 변경할 옵션정보를 저장한 버퍼의 주소 값 전달.
- **optlen** 네 번째 매개변수 optval로 전달된 옵션정보의 바이트 단위 크기 전달.