게임서버프로그래밍

2019년 2학기

한국산업기술대학교
게임공학부

정내훈

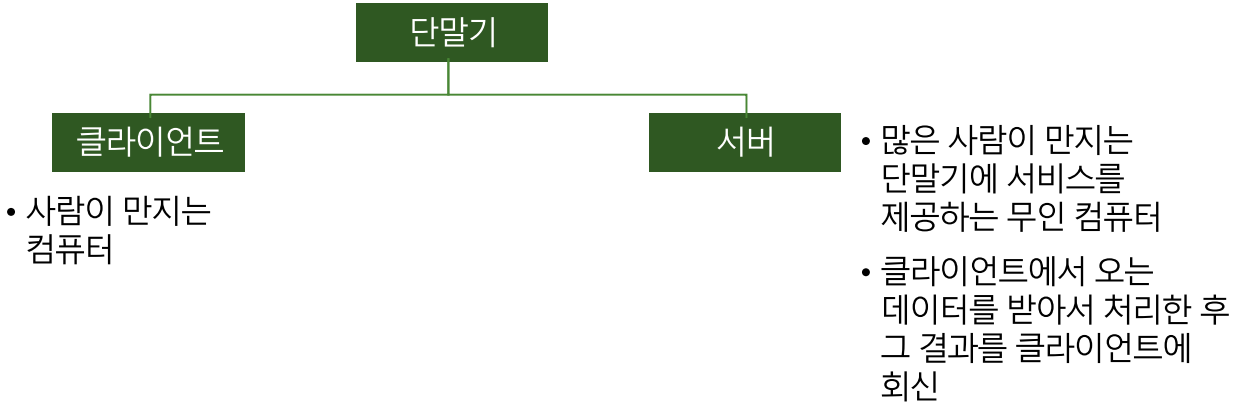
2장 컴퓨터 네트워크

2.1 | 컴퓨터 네트워크를 구성하는 기기

- 컴퓨터 네트워크의 단말기(terminal)와 네트워크 기기

단말기 : 통신을 하는 주체. 즉, 무선 혹은 유선 네트워크 연결 단자를 가진 일반적인 컴퓨터를 의미함.
데스크톱 컴퓨터, 노트북 컴퓨터, 스마트폰, 서버 컴퓨터 등이 포함.

그림 2-1 네트워크 관점에서 컴퓨터는 단말기로 지칭



2.1 | 컴퓨터 네트워크를 구성하는 기기

- 컴퓨터 네트워크 구성

그림 2- 2
컴퓨터 네트워크의
가장 기본적인 형태

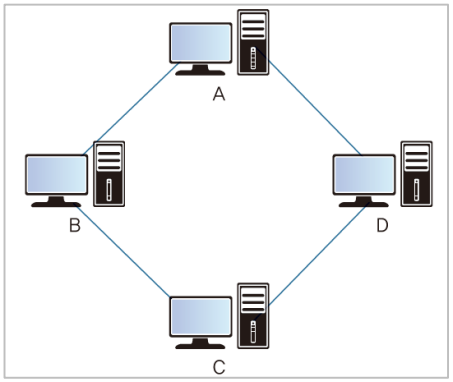


그림 2- 3
링 위상에서 통신

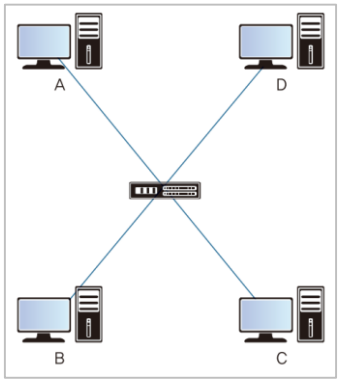


그림 2- 4
네트워크 기기를 사이에 둔 구성

OSI (Open Systems Interconnection reference) 모델
컴퓨터 네트워크 통신에 대한 국제 표준으로 이 표준만
잘 지키면 어떤 기기든, 어떤 형태의 통신 선로를
사용하든 컴퓨터 간 통신을 할 수 있다.

LAN (Local Area Network)
별 위상(star topology)

2.1 | 컴퓨터 네트워크를 구성하는 기기

- ## 2.1.1 OSI 모델

계층 1: 물리 계층
물리 계층에서는 하드웨어를 다룬다. 예를 들어 보낼 데이터를 어떤 파형의 전류로 보낼지 등을 정의.

계층 2: 데이터 링크 계층
로컬 지역 네트워크(LAN)에서 통신을 가능하게 함.

계층 3: 네트워크 계층
광역 통신망(WAN)에서 통신을 가능하게 함.

계층 4: 전송 계층
이 계층에서는 상대방에게 데이터가 반드시 도착하게 함.

계층 5: 세션 계층, 계층 6: 표현 계층, 계층 7: 응용 계층
응용 프로그램이나 운영체제 안 모듈이 다른 컴퓨터의 응용 프로그램이나 운영체제 모듈과 통신을 하는 동안 논리적 연결 단위나 기능들은 계층 5~7에서 다룬다. 그 예로 동영상 스트리밍의 통신 규약(MPEG), 통신 암호화 규약(SSL), 웹 브라우저와 웹 서버 간 통신 규약(HTTP) 등을 들 수 있음.

2.1 | 컴퓨터 네트워크를 구성하는 기기

- 2.1.2 OSI 모델의 계층 2
  - 각 단말기는 고유한 주소를 갖는다.
  - 단말기는 데이터를 프레임(frame)이라는 단위로 주고받는다.

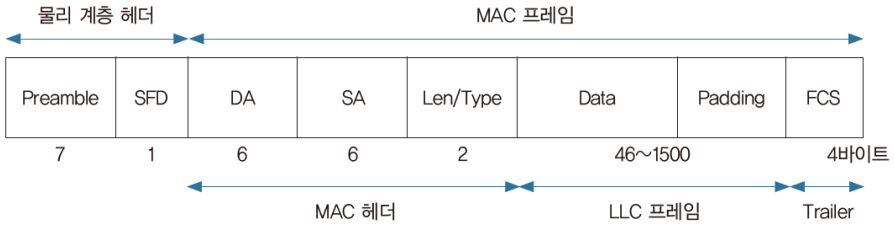


그림 2-5 프레임

- 보다 많은 컴퓨터를 연결할 때의 문제점
  - 스위치 하나가 연결할 수 있는 단말기 수가 제한되어 있다.
  - 단말기들 각각의 주소를 모두 고유하게 만들기 어렵다.

→서로 다른 네트워크 기기간을 연결하여 해결할 수 있다(해결책 중 하나)
→이렇게 서로 다른 LAN이 맞물려 연결된 것을 광역 통신망(Wide Area Network) 혹은 WAN이라고 함.

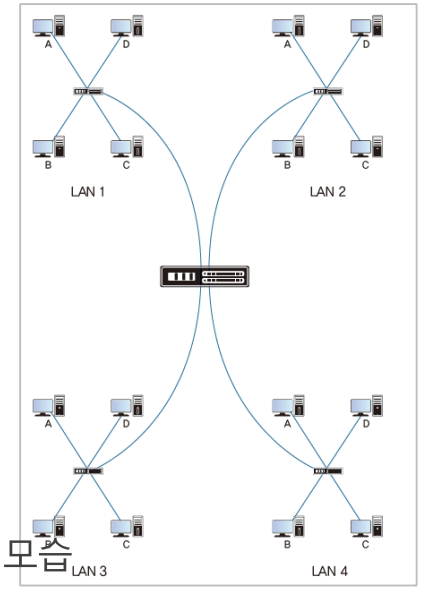


그림 2-6 LAN과 LAN을 연결한 모습

2.1 | 컴퓨터 네트워크를 구성하는 기기

- 2.1.3 OSI 모델의 계층 3
  - WAN의 모든 단말기는 OSI 모델의 계층 3에서 요구하는 형식의 주소를 가짐.
  - WAN에서 직접 데이터를 건네는 것이 아니라, 계층적으로 데이터를 건네주는 방식으로 작동, 이러한 일을 담당하는 네트워크 기기를 라우터(router)라고 함.

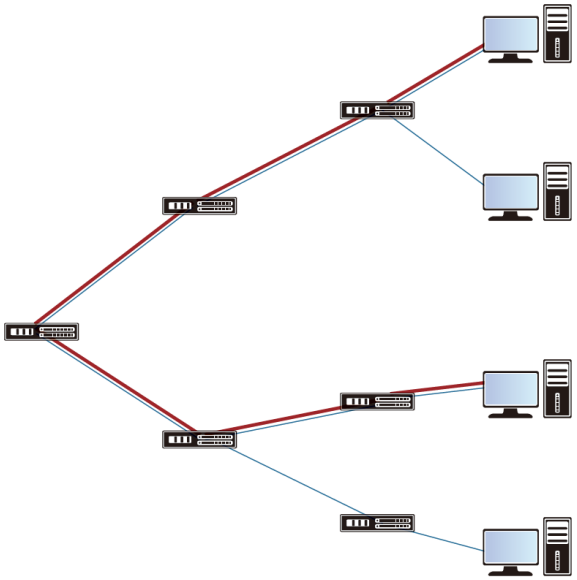


그림 2-7 라우터 계층에서 데이터 전달

그림 2-7 통신망에서 쓰는 라우터와 가정용 라우터

2.2 | 인터넷

- 인터넷
  - 서로 다른 종류의 많은 스위치와 라우터가 연결되어 지구를 뒤덮고 있다.

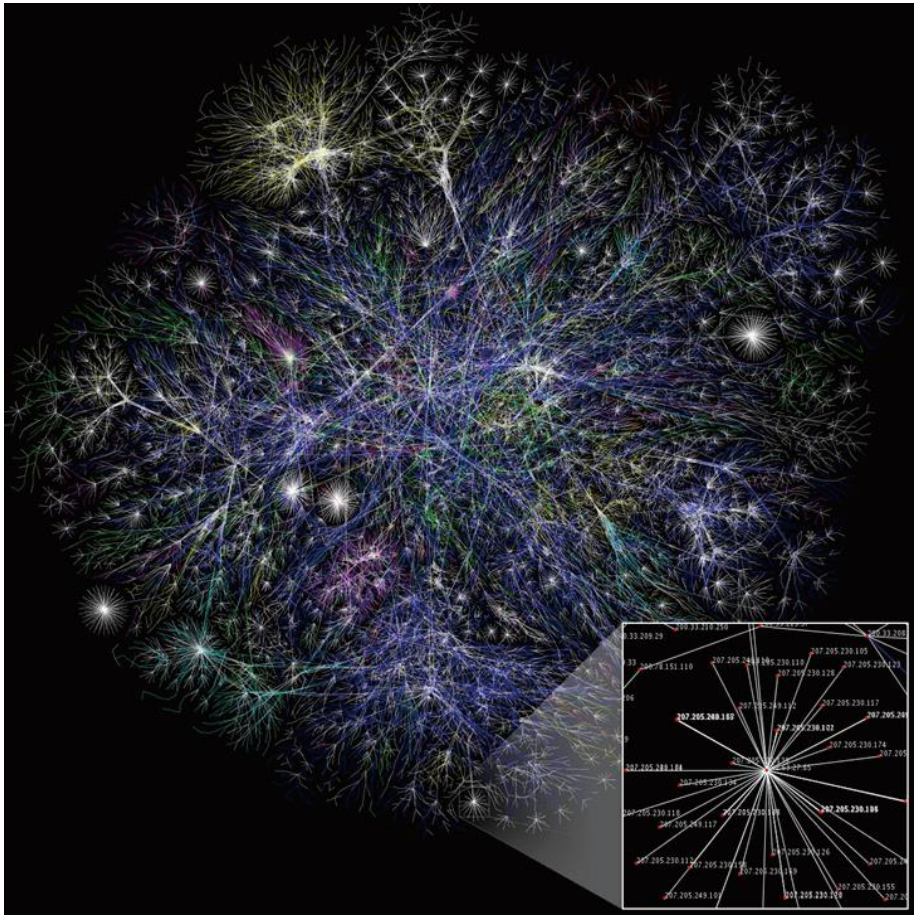


그림 2-9 지구를 연결하는 인터넷

2.3 | 컴퓨터 네트워크 데이터

- 2.3.1 스트림 형식
  - 스트림(stream) : 데이터의 흐름

```
PrintFileContents()
{
        fp = OpenFile("a.txt");
        while (!fp.IsEOF())
        {
                data = fp.ReadStream(100);
                Print(data);
        }
}
```

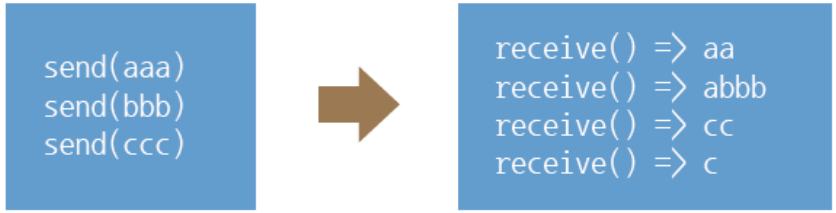


그림 2-10 스트림 송신 횟수와 수신 횟수는 불일치할 수도 있음

그림 2-11 헤더를 붙이는 방식과 구분자를 쓰는 방식

2.3 | 컴퓨터 네트워크 데이터

- 2.3.2 메시지 형식
  - 메시지는 자체적으로 데이터 시작과 끝을 구별할 수 있다.
  - 메시지 형식 : 각 데이터가 정확히 구별되는 것.
  - 데이터 시작과 끝을 구별할 필요가 없으며 메시지를 여러 필드 (field)로 나누어서 사용하는 것으로 충분함.

0~1	2~13	14~25	26~27
MessageType_FireBullet	Position(x, y, z)	Direction(x, y, z)	Bullet Type
총알이 발사되었다	총알이 발사된 위치	총알이 발사된 방향	총알의 종휴

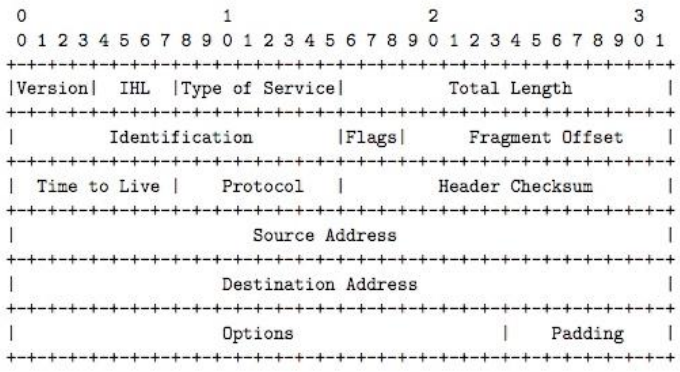


그림 2-12 IP 패킷의 구조

2.3 | 컴퓨터 네트워크 데이터

- 단편화(fragmentation)
  - 프로그램이 매우 긴 스트림을 송신할 때 운영체제는 이를 IP 패킷의 크기 제한에 맞추어 여러 조각을 냄.
  - 각 조각은 IP 패킷 하나하나가 되어 받는 쪽에 송신함.
  - 받는 쪽에서는 이 조각들을 받아 조립한 후 스트림 형태로 복원하며, 이 과정도 운영체제 안에 서 진행.
  - 조립된 스트림은 프로그램으로 넘어가 처리.

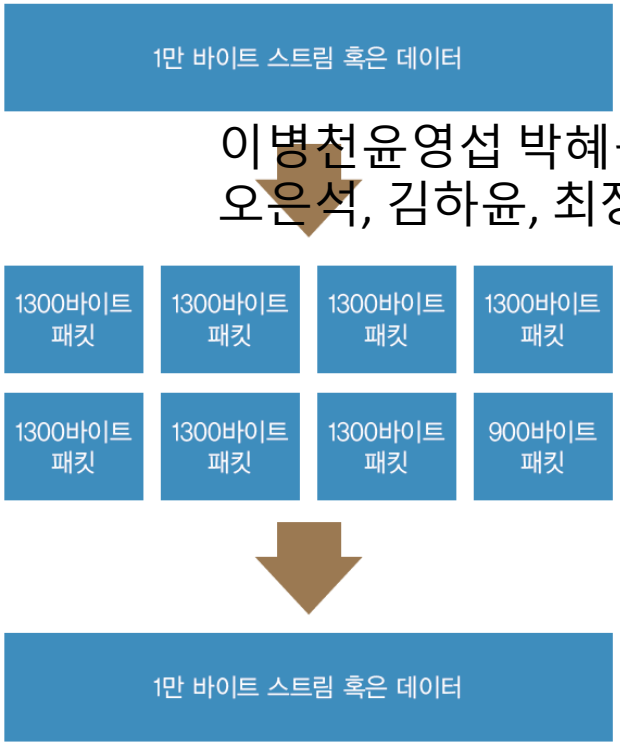


그림 2-13 스트림을 패킷으로 조각 내고, 이를 다시 스트림으로 복원

2.4 | 컴퓨터 네트워크 식별자

- 컴퓨터 네트워크 식별자(주소)는 인터넷에서 모두 고유하다.

  ipconfig 도구를 이용하면 기기에 활성화된 인터넷 주소 목록을 볼 수 있다.

```
C:\Users>ipconfig

Windows IP 구성

이더넷 어댑터 로컬 영역 연결:

   연결별 DNS 접미사. . . . :
   링크-로컬 IPv6 주소 . . . . : fe80::f999:a42:e88c:601d%2
   IPv4 주소 . . . . . . . . . : 172.16.10.107
   서브넷 마스크 . . . . . . . : 255.255.255.0
   기본 게이트웨이 . . . . . . : 172.16.10.254

이더넷 어댑터 vEthernet <vnet-1>:

   연결별 DNS 접미사. . . . :  corp.nettention.com
   링크-로컬 IPv6 주소 . . . . : fe80::50c:b666:58e6:2733x18
   IPv4 주소 . . . . . . . . . : 192.168.77.130
   서브넷 마스크 . . . . . . . : 255.255.255.0
   기본 게이트웨이 . . . . . . : 192.168.77.1
...(중략)
```

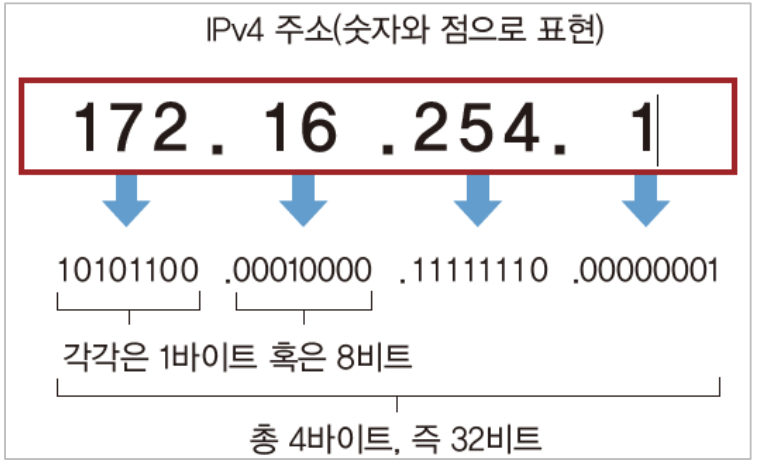


그림 2-14 IPv4 주소 형식

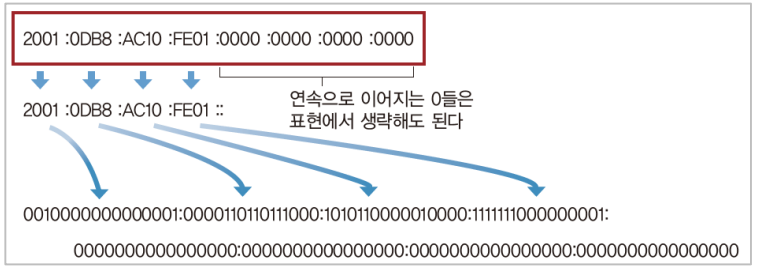
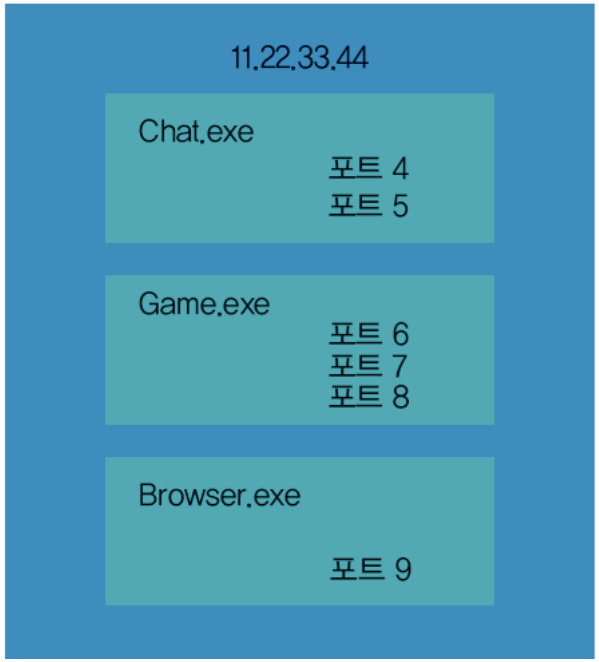


그림 2-15 IPv6 주소 형식

2.4 | 컴퓨터 네트워크 식별자

- 포트(port) : 한 IP 주소 안에서도 누가 주고받는 것인지 식별하는 역할

  2바이트 정수로 65535 이하의 값 중에서 할당하여 쓸 수 있다.




```
...(중략)
TCP    172.16.10.107:54728    172.217.24.133:443     ESTABLISHED     19136
TCP    172.16.10.107:54908    74.125.203.189:443     ESTABLISHED     19136
TCP    172.16.10.107:55044    104.244.42.200:443     ESTABLISHED     19136
TCP    172.16.10.107:55178    210.89.160.88:443      ESTABLISHED     19136
TCP    172.16.10.107:55231    172.217.25.234:443     TIME_WAIT       0
TCP    172.16.10.107:55310    210.115.150.3:80       TIME_WAIT       0
TCP    172.16.10.107:55311    52.109.44.29:443       TIME_WAIT       0
TCP    172.16.10.107:55313    172.217.161.228:443    ESTABLISHED     19136
TCP    172.16.10.107:55319    34.231.248.5:443       CLOSE_WAIT      19136
TCP    172.16.10.107:55320    34.231.248.5:443       TIME_WAIT       0
TCP    172.16.10.107:55321    172.217.25.98:443      ESTABLISHED     19136
TCP    172.16.10.107:55332    172.217.161.42:443     ESTABLISHED     19136
TCP    172.16.10.107:55342    13.107.18.11:443       ESTABLISHED     52836
TCP    172.16.10.107:55347    13.107.21.200:443      ESTABLISHED     52836
...(중략)
```

IP 주소와 포트를 같이 표현하는 방법은 <IP 주소:포트> 형식.
이렇게 IP 주소와 포트를 한데 묶어서 끝점(endpoint)이라고 함.

그림 2-16
단말기 안에 프로세스가 있고, 프로세스 안에 포트들이 있음

2.4 | 컴퓨터 네트워크 식별자

- 호스트 이름(host name)

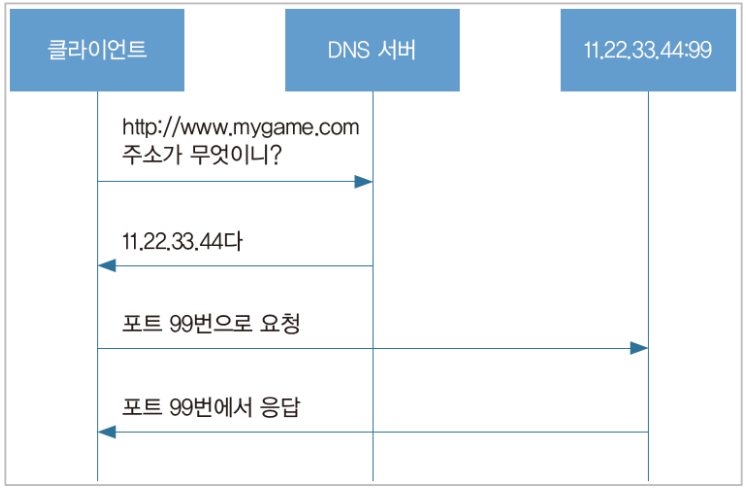


그림 2-17 호스트 이름에서 IP 주소를 얻고 통신 시작

- 우리가 호스트 이름을 사용하면 컴퓨터는 이를 IP 주소로 바꾸는 것을 확인할 수 있다

```
C:\>ping www.naver.com
ping www.naver.com.nheos.com [202.179.177.21]
```

2.5 | 컴퓨터 네트워크의 품질과 특성

- ## 2.5.1 네트워크의 품질을 저해하는 것들

스위치나 라우터에 자신이 처리할 수 있는 한계를 넘는 데이터가 수신되면

1. 자기가 처리할 수 있는 것 이상을 그냥 버린다. - 패킷 유실(packet loss)이 일어남
2. 아직 처리하지 못한 것들을 메모리에 누적한다.

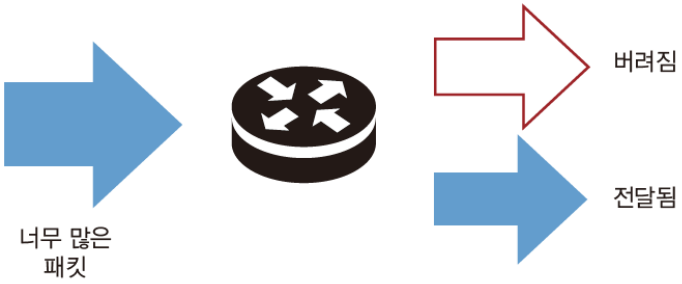


그림 2-18 감당하지 못하는 처리량은 그냥 버리는 라우터가 있음

그림 2-19 감당하지 못하는 처리량을 억지로 끌어안는 라우터도 있음

2.5 | 컴퓨터 네트워크의 품질과 특성

- ## 2.5.1 네트워크의 품질을 저해하는 것들

데이터는 유선이나 무선 회선으로 나가는 과정에서 일정 진폭이나 주파수의 전기(유선) 신호나 전자기파(무선) 신호로 변경되어 방출된다.
→ 디지털 정보를 아날로그 신호로 바꾼 것으로, 이 일은 OSI 모델의 계층 1에서 일어남.

수신자는 아날로그 신호를 받으면 진폭, 주파수 등을 파악하여 디지털 정보로 바꾸고 이를 OSI 모델의 계층 2 처리 모듈로 넘기면 이후 계층 3 모듈로 넘어감. 이 과정에서 아날로그 신호에 변화가 생길 수 있다. (오류 발생)
→ 네트워크 기기에 데이터가 가는 동안 잡음이 섞이면 프레임이나 패킷을 그냥 버려짐

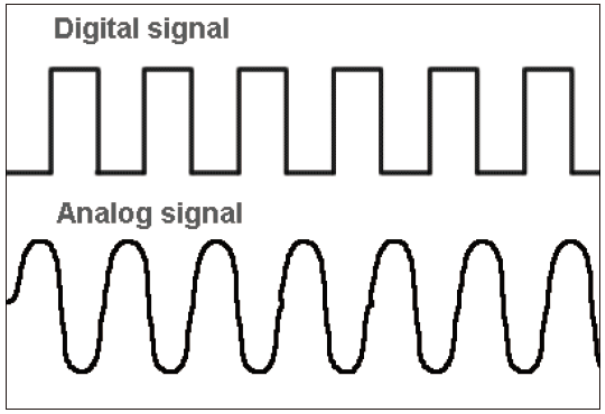


그림 2-20 하드웨어 레벨에서 데이터 형태

그림 2-21 정상적이던 데이터가 중간에 변경되어 버린 상태

네트워크 기기가 처리할 수 있는 한계를 넘어가면
패킷 유실이 발생할 수 있다.

회선 신호가 약하거나 잡음이 섞이면
패킷 유실이 발생할 수 있다.

2.5 | 컴퓨터 네트워크의 품질과 특성

- 2.5.2 전송 속도와 전송 지연 시간

전송 속도 :

두 기기 간에 초당 전송될 수 있는 최대 데이터양을 의미함. 보통 초당 비트수(bits per second, bps) 혹은 바이트 수(Bytes per second, B/s)로 표현

전송 속도에 영향을 주는 것으로는 두 기기 간의 매체인 선로의 종류와 품질, 두 기기의 소프트웨어와 하드웨어 종류를 꼽을 수 있다.

레이턴시 :

두 기기 간에 데이터를 최소량 전송할 때 걸리는 시간을 의미함. 많이 쓰는 단위는 밀리초(ms). 레이턴시에 영향을 주는 것으로는 매체의 종류와 품질, 송신자 -수신자 사이의 라우터 처리 속도가 있다.

송신자와 수신자 간 네트워크 기기 안 하드웨어와 소프트웨어의 처리 속도도 네트워크 레이턴시의 원인이 된다.

```
C:\Users>ping www5.usp.br

Ping www5.usp.br [200.144.248.54] 32바이트 데이터 사용:
200.144.248.54의 응답: 바이트=32 시간=333ms TTL=240
200.144.248.54의 응답: 바이트=32 시간=333ms TTL=240
200.144.248.54의 응답: 바이트=32 시간=333ms TTL=240
200.144.248.54의 응답: 바이트=32 시간=333ms TTL=240
```

한국에서 남미에 있는 컴퓨터로 가는 레이턴시 측정(ping 도구 이용)

2.5 | 컴퓨터 네트워크의 품질과 특성


```
C:\Users>tracert www5.usp.br

최대 30홉 이상의
www5.usp.br [200.144.248.54](으)로 가는 경로 추적:

  1    <1 ms     1 ms     1 ms  172.16.10.254
  2     *        *        *     요청 시간이 만료되었습니다.
  3     *        *        *     요청 시간이 만료되었습니다.
  4     1 ms    <1 ms     1 ms  112.188.53.197
  5     *        *        *     요청 시간이 만료되었습니다.
  6     1 ms     1 ms     2 ms  112.174.49.66
  7     1 ms     1 ms     1 ms  112.174.84.46
  8   140 ms   140 ms   140 ms  112.174.88.174
  9   236 ms   236 ms   214 ms  sl-mpe51-sea-.sprintlink.net [144.224.113.125]
 10   212 ms   217 ms   207 ms  144.232.0.108
 11   240 ms   234 ms   221 ms  ae-19.a01.sttlwa01.us.bb.gin.ntt.net [129.250.8.149]
 12   239 ms   237 ms   215 ms  ae-2.r04.sttlwa01.us.bb.gin.ntt.net [129.250.5.85]
 13   218 ms   216 ms   185 ms  ae-5.r23.sttlwa01.us.bb.gin.ntt.net [129.250.2.7]
 14   256 ms   265 ms   289 ms  ae-3.r23.snjsca04.us.bb.gin.ntt.net [129.250.3.124]
 15   275 ms   252 ms   261 ms  ae-0.r22.snjsca04.us.bb.gin.ntt.net [129.250.2.182]
 16   299 ms   298 ms   299 ms  ae-7.r23.asbnva02.us.bb.gin.ntt.net [129.250.6.238]
 17   335 ms   281 ms   317 ms  ae-1.r20.miamfl02.us.bb.gin.ntt.net [129.250.2.87]
 18   312 ms   287 ms   314 ms  ae-10.r04.miamfl02.us.bb.gin.ntt.net [129.250.3.142]
 19   319 ms   322 ms   295 ms  ae-3.a01.miamfl02.us.bb.gin.ntt.net [129.250.3.208]
 20   223 ms   223 ms   223 ms  xe-0-0-26-2.a01.miamfl02.us.ce.gin.ntt.net [129.250.202.94]
 21   363 ms   360 ms   349 ms  sp-mia2-par-pac.bkb.rnp.br [200.143.252.33]
 22     *        *        *     요청 시간이 만료되었습니다.
 23     *      332 ms   332 ms  143.107.151.62
 24     *        *        *     요청 시간이 만료되었습니다.
 25   332 ms   333 ms   332 ms  www5.usp.br [200.144.248.54]

추적을 완료했습니다.
```

한국에서 남미에 있는 컴퓨터로 가는 레이턴시 측정(tracert 도구를 이용)

두 단말기 사이의 레이턴시
= 두 단말기 사이에 있는 네트워크 기기의 레이턴시 총합

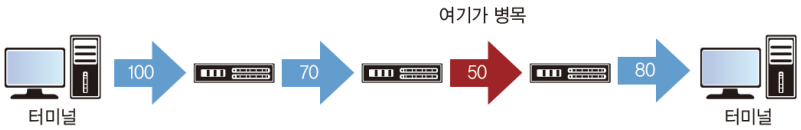


그림 2-23 각 네트워크 기기에서 최대 스루풋과 병목 스루풋

두 단말기 사이의 스루풋 =
두 단말기 사이의 네트워크 기기 중 최소 스루풋

2.5 | 컴퓨터 네트워크의 품질과 특성

- 2.5.3 네트워크 품질 기준 세 가지

전송 속도(스루풋) 높을 수록 좋다	패킷 유실률 낮을 수록 좋다	레이턴시 낮을 수록 좋다
• 전송될 수 있는 데이터의 단위 시간당 총량 • 회선의 종류가 좋을수록, 네트워크 장비의 처리 속도가 빠를수록 향상.	• 전송되는 데이터가 중간에 버려지는 비율 • 회선 품질이 좋을수록, 경로에 있는 라우터 개수가 적을수록, 라우터의 처리 성능이 좋을수록 낮다.	• 전송되는 데이터가 목적지에 도착하는데 걸리는 시간 • 회선 길이가 길수록, 경로에 라우터 개수가 많을수록 , 라우터의 처리 성능이 나쁠수록 높다.

2.5 | 컴퓨터 네트워크의 품질과 특성

- 2.5.4 무선 네트워크의 품질

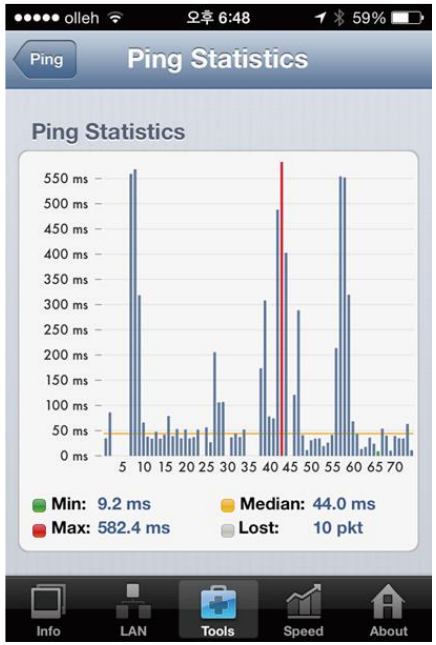


그림 2-24
무선 네트워크 기기의 ping 결과

와이파이에서 데이터를 전송할 때

1. 데이터를 전파로 변환하여 보내기 직전에 안테나를 통해 다른 전파가 감지되는지 확인.
2. 전파가 감지되는 것이 없으면 전파를 보냄.
3. 전파가 감지되면 잠시 기다렸다가 1~2 과정을 반복.
4. 신호를 보낸 후 상대방에게서 '신호를 받았음' 응답이 오는지 체크.
5. 응답이 일정 시간 동안 없으면 보냈던 신호를 다시 보냄..

CSMA(Carrier Sense Multiple Access)

2.6 | 컴퓨터 네트워크에서 데이터 보내기와 받기

- ## 2.6.1 UDP 네트워킹

User Datagram Protocol의 약어로, 사용자가 정의한 데이터그램(datagram)을 상대방에게 보낼 수 있게 하는 통신 규약(프로토콜)

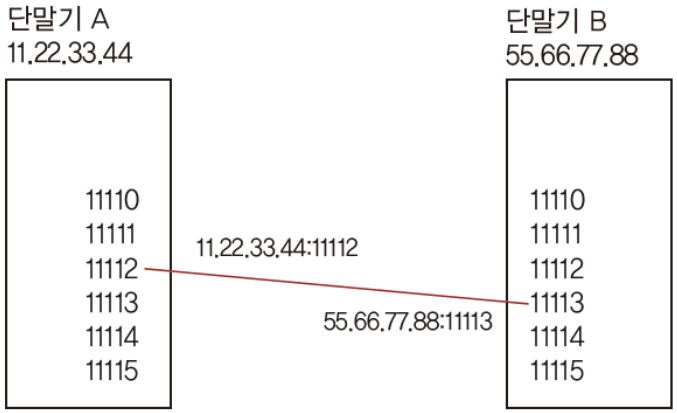


그림 2-26 UDP로 데이터 보내기

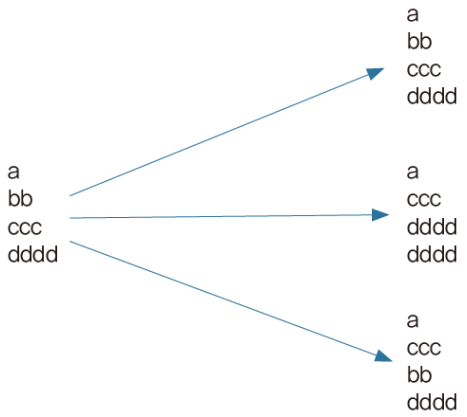


그림 2-27 UDP로 데이터 보내기

UDP를 이용하여 데이터그램을 보낼 수 있다.
데이터 그램은 메시지 성질을 가진다.
데이터 일부가 뭉치거나 쪼개지지 않는다.
패킷 유실 현상이 발생할 수 있다.

받는 쪽에서 데이터그램 유실이나 순서 뒤바뀜 혹은 중복 수신 현상이 발생해도 괜찮을 때만 UDP를 사용하는 것이 좋다.

2.6 | 컴퓨터 네트워크에서 데이터 보내기와 받기

- 단말기 A 11.22.33.44에서 송신

```
main()
{
        s = socket(UDP);                              //❶
        s.bind(any_port);                             //❷
        s.sendTo("55.66.77.88:5959", "hello");        //❸
        s.close();                                    //❹
}
```

❶ 매개변수로 UDP를 쓰겠다고 선언하면 소켓 핸들을 받는다.

❷ 데이터를 주고 받기 위해 할당하는 함수로 bind() 호출한다.

❸ sendTo()를 호출해서 상대방의 주소와 포트, 즉 끝점에 원하는 데이터를 보낸다.

❹ close()를 호출하면 소켓 핸들이 닫힌다.

2.6 | 컴퓨터 네트워크에서 데이터 보내기와 받기

- 데이터를 수신하는 쪽 코드

```
main()
{
        s = socket(UDP);
        s.bind(5959);                          //❶
        r = s.recvfrom();                          //❷
        print(r.srcAddrPort, r.data);              //❸
        s.close();
}
```

❶ 송신 측에서 의도적으로 포트 5959번 으로 데이터를 보냈기 때문에, bind()를 호출할 때 포트 번호를 명시적으로 지정함

❸ 데이터가 도착하면 비로소 함수는 리턴하고 받은 데이터와 데이터를 보냈던 송신자 주소가 변수 r에 채워집니다. r.data는 바이너리 데이터.

- UDP의 다대다 통신

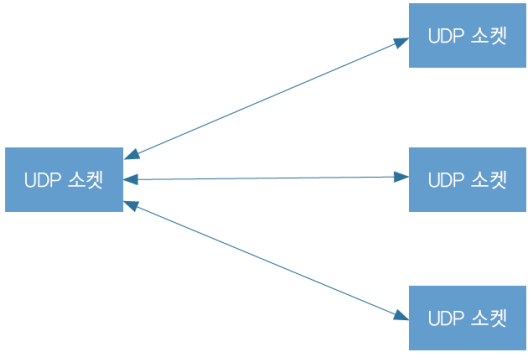


그림 2-28 UDP는 다대다 통신 가능

2.6 | 컴퓨터 네트워크에서 데이터 보내기와 받기

- ## 2.6.2 TCP 네트워킹

TCP(Transmission Control Protocol; 전송 제어 프로토콜) : 보내는 쪽 데이터가 받는 쪽에서 완전히 동일함을 보장해 주는 프로토콜
연결 지향형(connection oriented)이라고 한다.
인터넷 프로그램 대부분이 활용.
스트림 형태로 데이터를 뭉치거나 쪼갤 수 있다.

그림 2-29 TCP는 일대일만 가능

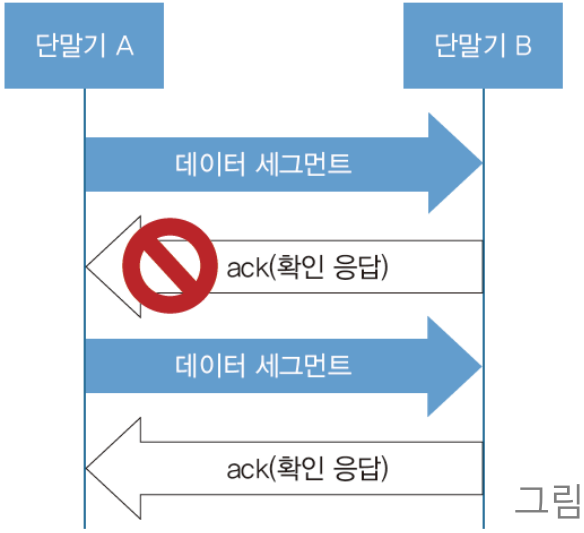


그림 2-30 세그먼트 응답이 올 때까지 데이터 보관 밑 재전송

2.6 | 컴퓨터 네트워크에서 데이터 보내기와 받기

- 송신 측 코드

```
<11.22.33.44>
main()
{
        s = socket(TCP);                          //❶
        s.bind(any_port);
        s.connect("55.66.77.88:5959");            //❷
        s.send("hello");                          //❸
        s.close();
}
```

❶ 소켓을 생성할 때 UDP가 아닌 TCP를 쓰겠다고 지정해야 한다.

❷ 상대방과 연결을 위해 connect(addr) 함수를 호출하여 addr에 상대방 측의 끝점을 넣어 준다.

❸ 데이터를 보낼 때는 sendTo()가 아닌 send(data)를 호출한다.(이미 connect() 함수로 연결 끝점을 지정했기 때문)

2.6 | 컴퓨터 네트워크에서 데이터 보내기와 받기

- 수신 측 코드

```
<55.66.77.88>
main()
{
        s = socket(TCP);              //1
        s.listen(5959);               // 2
        s2 = s.accept();              //3
        print(getpeeraddr(s2));       //4
        while (true)
        {
                r = s2.recv();        //5
                if (r.length = = 0)
                  break;              //6
                print(r);             //7
        }
        s2.close();                   //8
}
```

1 소켓을 생성한다.

2 끝점을 확보한다.

3 accept()를 실행하면 상대방에게 TCP 연결을 받을 때까지 블로킹된다.

4 TCP 연결을 맺은 socket의 상대방 측 끝점을 보고 싶다면 getpeeraddr()를 호출한다.

5 TCP로 수신할 때는 recvfrom() 대신 recv()를 호출한다.

6, 8 TCP 소켓에 대해 recv()가 0바이트 수신하면 이는 '연결이 종료되었음'을 의미, 이때는 연결이 끊어졌으므로 마감 처리를 해야 한다.

7 출력

```
11.22.33.44:51409
hello
```

2.7 | 패킷 유실 시 UDP와 TCP에서 현상

- 패킷 유실이 발생하면

  UDP에서는 데이터그램이 유실됨.

  TCP에서는 중간에 지연 시간이 발생함.

UDP의 레이턴시 = 네트워크 기기의 레이턴시

TCP의 레이턴시 = 네트워크 기기의 레이턴시 + 1(00% - 패킷 유실률) * 재전송 대기 시간

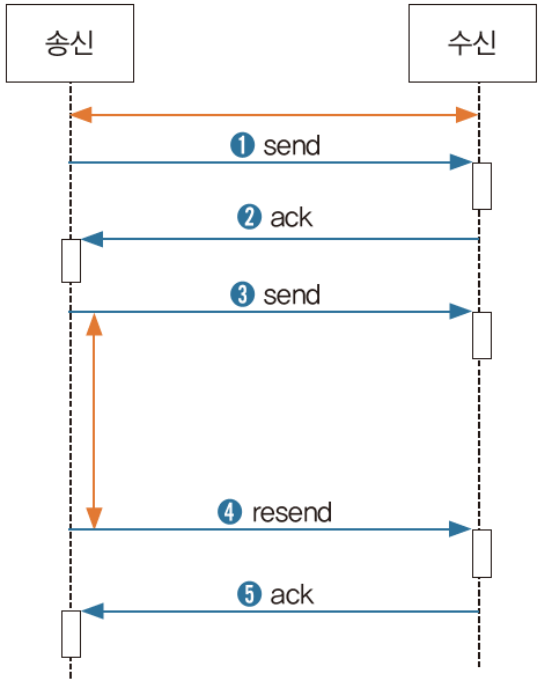


그림 2-31 수신 확인이 되지 않아 재송신을 하는 상황

- UDP는 주로 레이턴시가 민감하거나 패킷 유실이 있어도 괜찮은 곳에서 주로 사용한다. 예를 들어 캐릭터 이동이나 기관총 난사, 음성이나 화상 데이터 전송에는 UDP를 많이 사용.
- 대전 격투 게임이나 실시간 전략 시뮬레이션 게임(RTS)에서도 UDP를 사용하기도 한다.
- 그 외의 모든 경우에서는 TCP를 사용.

2.8 | 주로 사용하는 메시지 형식

- 게임 플레이에서 사용되는 메시지 형식 (**패킷** 포맷이라고 부름)

```
BuyItem<LF>
Sword<LF>
1<LF>
<0x00>
```

텍스트 형식의 메시지
: 사람이 읽을 수 있다

```
0x01 | 0x0023 | 0x0001
^ BuyItem
       ^ Sword
                ^ 1
```

바이너리 형식의 메시지
: 사람이 읽기 어렵다.

```
BuyItem<LF>
Sword<LF>
136
1<LF>
<0x00>
```

메타 데이터가 없는 형식

```
Action: BuyItem<LF>
Type: Sword<LF>
Amount: 1<LF>
<0x00>
```

메타 데이터가 있는 형식

```
0x0001 | 0x01 | 0x0005 | 0x0023 | 0x0046 | 0x0001
^ Action
        ^ BuyItem
              ^ Type
                     ^ Sword
                            ^ Amount
                                   ^ 1
```

바이너리 형식으로도 메타데이터를 포함시킬 수 있다.

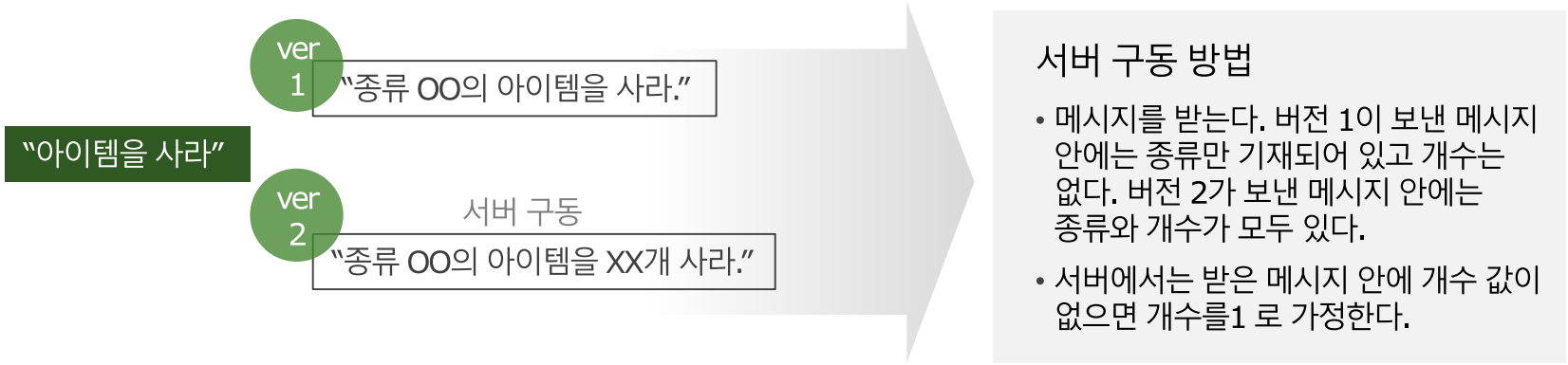
2.8 | 주로 사용하는 메시지 형식

- 메타데이터를 이용하면 하위 호환성 측면에서 유리하다

서버에서 메시지 안에 원하는 내용 일부가 있는지 없는지에 따라 작동을 다르게 만들어 준다면
버전 1과 버전 2의 클라이언트를 동시에 처리할 수 있다.

모바일앱에서는 메타데이터 방식이 유리하다. (웹서버 시스템을 그대로 사용할 수 있다. HTML/JSON등)

MMORPG는 바이너리 방식을 사용한다. (대역폭 낭비가 적고, 변환이 불필요하다.)



2.9 | 네트워크 주소 변환

- 네트워크 주소변환(Network Address Translation, NAT)

  다른 단말기로 전송되던 패킷의 송신자 주소나 수신자 주소가 다른 것으로 변환되는 과정, NAT 변환을 하는 기기를 NAT 라우터라고 한다.

  NAT 라우터가 주로 하는 일은 IP 주소 1개를 여러 기기가 공용하도록 만들어 주는 것이다.

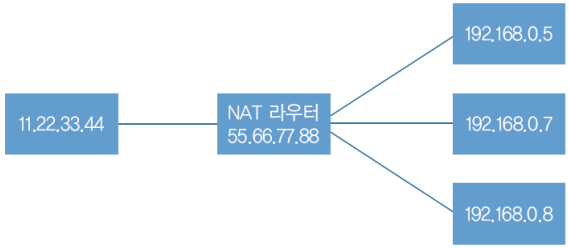


그림 2-32 NAT 라우터 역할

그림 2-33 NAT 라우터의 통신 단자

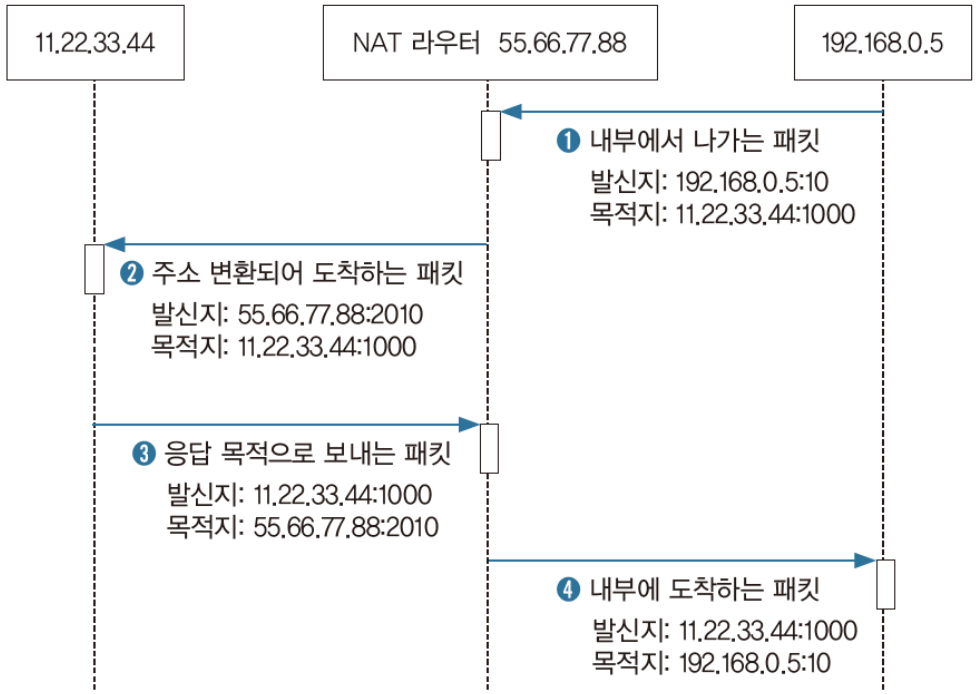


그림 2-34 NAT 라우터(공유기) 작동 예

2.10 | 요약

- 최신 컴퓨터 네트워크의 기본 유형은 LAN으로, LAN은 단말기와 네트워크 기기(스위치) 로 구성되며, 근거리 통신망임.
- LAN들은 서로 연결될 수 있으며, 그 사이에 라우터가 있다. LAN들의 연결 집합을 WAN이라고 하며, 원거리 통신망이다.
- WAN이 전 세계로 덮인 것을 인터넷이라고 한다.
- 단말기는 클라이언트와 서버로 구별된다. 클라이언트는 사람들이 쓰는 통상의 컴퓨터고,서버는 사람이 평소 만지지는 않지만 여러 사람의 네트워크 연결 및 요청을 받아 처리하는 역할을 한다.
- 통신하는 데이터 단위는 스트림과 메시지로 구별된다. 스트림은 주고받는 데이터의 시작과 끝 구별이 없고, 보내는 쪽과 받는 쪽의 주고 받는 단위가 동일하지 않을 수 있다. 메시지는 그 반대이다.
- 인터넷에서 컴퓨터가 서로 통신하려면 IP 주소나 호스트 이름이 있어야 하고, 프로세스 간 서로 통신하려면 포트 번호도 있어야 한다.
  주소와 포트 번호를 합쳐서 끝점(endpoint)이라고 한다.
- 매체, 즉 통신 선로의 품질에 따른 잡음이나 신호가 약하거나 네트워크 기기의 과부하로 통신 품질이 낮아질 수 있다.
  통신 품질에 관련된 것으로 레이턴시, 스루풋, 패킷 유실률이 있다.
- 프로세스 간 통신을 하려면 소켓을 다루어야 한다.
- 사용자 프로그램에서 다루는 대표적인 통신 규약(프로토콜)은 TCP와 UDP가 있다. TCP는 거의 모든 종류의 메시지에서 사용되고, UDP는 캐릭터 이동이나 음성, 화상 전송처럼 유실이 다소 허용되는 곳에서 사용한다. TCP는 스트림 형식이고, UDP는 메시지 형식이다.
- 사용자가 주고받는 데이터 구조는 텍스트나 이진 형식으로, 메타데이터를 부가하면 하위호환성에서 유리하지만, 통신량 같은 문제가 따른다.
- NAT 주소 변환 기술은 인터넷 공유기처럼 주소 1개로 여러 기기가 공유하는 용도로 활용한다.

2.11 | 더 읽을거리

- 라우팅(routing)과 주소 결정 프로토콜 (Address Resolution Protocol, ARP)
- Dual Stack과 NAT64/DNS64에 대한 이해
- TCP 흐름 제어
- Reliable UDP 혹은 RUDP
- 송신자와 수신자 간 메시징을 암호화
- Full cone NAT, uPNP 등
- 클라우드 서비스
- NAT64 & DNS64
- 원격 프로시저 호출(Remote Procedure Call, RPC) 혹은 원격 메서드 호출(Remote Method Invocation, RMI)

실습

• 1 대 1 소켓 통신 프로그램을 만들어 보자. (클라이언트 프로그램 샘플)

```
#include <stdio.h>
#include <WS2tcpip.h>
#pragma comment(lib, "Ws2_32.lib")
#define MAX_BUFFER        1024
#define SERVER_IP  "127.0.0.1"
#define SERVER_PORT       3500
int main()
{
   WSADATA WSAData;
   WSAStartup(MAKEWORD(2, 0), &WSAData);
   SOCKET serverSocket = WSASocket(AF_INET, SOCK_STREAM, 0, NULL, 0, 0);
   sockaddr_in serverAddr;
   memset(&serverAddr, 0, sizeof(SOCKADDR_IN));
   serverAddr.sin_family = AF_INET;
   serverAddr.sin_port = htons(SERVER_PORT);
   inet_pton(AF_INET, SERVER_IP, &serverAddr.sin_addr);
   connect(serverSocket, (struct sockaddr *)&serverAddr, sizeof(serverAddr));
   while (true) {
      char messageBuffer[MAX_BUFFER];
      printf("Enter Message -> ");
      scanf_s("%s", messageBuffer, MAX_BUFFER);
      int bufferLen = strlen(messageBuffer);
      int sendBytes = send(serverSocket, messageBuffer, bufferLen, 0);
      if (sendBytes > 0) {
         printf("TRACE - Send message : %s (%d bytes)\n", messageBuffer, sendBytes);
         int receiveBytes = recv(serverSocket, messageBuffer, MAX_BUFFER, 0);
         if (receiveBytes > 0)
            printf("TRACE - Receive message : %s (%d bytes)\n", messageBuffer, receiveBytes);
      }
   }
   closesocket(serverSocket);
   WSACleanup();
}
```

실습

- 1 대 1 소켓 통신 프로그램을 만들어 보자. (서버 프로그램 샘플)

```
#include <stdio.h>
#include <WS2tcpip.h>
#pragma comment(lib, "Ws2_32.lib")
#define MAX_BUFFER        1024
#define SERVER_PORT       3500
int main()
{
   WSADATA WSAData;
   WSAStartup(MAKEWORD(2, 0), &WSAData);
   SOCKET listenSocket = WSASocket(AF_INET, SOCK_STREAM, 0, NULL, 0, 0);
   SOCKADDR_IN serverAddr;
   memset(&serverAddr, 0, sizeof(SOCKADDR_IN));
   serverAddr.sin_family = AF_INET;
   serverAddr.sin_port = htons(SERVER_PORT);
   serverAddr.sin_addr.S_un.S_addr = htonl(INADDR_ANY);
   ::bind(listenSocket, (sockaddr *)&serverAddr, sizeof(serverAddr));
   listen(listenSocket, 5);
   SOCKADDR_IN client_addr;
   while (true) {
      int addr_size = sizeof(client_addr);
      SOCKET client_socket = accept(listenSocket, (sockaddr*)&client_addr, &addr_size);
      while (true) {
        char messageBuffer[MAX_BUFFER];
        int receiveBytes = recv(client_socket, messageBuffer, MAX_BUFFER, 0);
        if (receiveBytes > 0) printf("TRACE - Receive message : %s (%d bytes)\n", messageBuffer, receiveBytes);
        else break;
        int sendBytes = send(client_socket, messageBuffer, receiveBytes, 0);
        if (sendBytes > 0) printf("TRACE - Send message : %s (%d bytes)\n", messageBuffer, sendBytes);
      }
      closesocket(client_socket);
   }
   closesocket(listenSocket);
   WSACleanup();
}
```

숙제 (#2)

게임 서버/클라이언트 프로그램 작성

내용

숙제 (#1)의 프로그램을 Client/Server 모델로 분리
프로그램 2개 작성

Client :

키입력을 받아서 서버 프로그램에 보낸다
서버에서 보내온 좌표로 말을 이동시킨다.
시작할 때 서버의 IP주소를 입력 받는다.

Server

클라이언트에서 보내온 키입력을 보고 말의 위치 변경
변경된 위치를 클라이언트에 전송

목적

Windows 네트워크 프로그래밍 숙달

제약

Windows에서 Visual Studio로 작성 할 것
그래픽의 우수성을 보는 것이 아님

제출

10월 2일 수요일 오후 1시 30분 까지
Zip으로 소스를 묶어서 e-mail로 제출
제목에 “2019 게임서버 학번 이름 숙제 2번”