<컴퓨터 네트워크>

1. 네트워크 내부 구조
   - 계층적 구조
2. 네트워크의 역사(통신의 역사)
   - 통신이 가능해야 네트워크가 가능. (통신은 네트워크의 필요조건)
   - **우편** -> **봉화**(최초의 전자기 신호(빛)를 이용한 디지털(이산적인 신호) 통신 시스템)

     -> **전보**(글자를 모스부호로 전달, Data Network, 적은 데이터, 디지털 네트워크) ->

     전화(Voice Network, 많은 데이터, 아날로그 네트워크, 사용자의 폭발적 증가, 서브 네트워크에서 다른 서브 네트워크로의 통신 가능한 자동 교환 방식) -> **컴퓨터**

   - 데이터 네트워크/ 디지털 네트워크가 아날로그 네트워크보다 가공하기 편하고, 효율이 좋다.
   - 컴퓨터 통신
     1) 네트워크 엣지(단말기 – PC, 서버) + 네트워크 코어
        - 네트워크 코어 : 네트워크를 전송해주는 부분.
        - 네트워크 엣지 : 네트워크를 사용하는 사용자.
     2) ↓↓↓네트워크 엣지를 네트워크 망에 연결하는 마지막 1단계 통신 방법 ↓↓↓
     3) 전화선(음성정보 전달)을 이용한 통신 : Dial-up Modem (Binary데이터를 가청 주파수 20hz ~ 20,000hz 로 전환하여 데이터 전달), 전화를 걸어 통신, 음성 정보와 데이터 정보를 동일한 방식으로 처리(고주파 데이터는 전화망에서 잘려 나감.), 네트워크 코어=전화망 자체에서 데이터 처리 ,데이터 전송률 = 1,200bps ~ 9,600bps/14,400bps , Dedicated Channel
     4) DSL(Digital Subscriber Line)을 이용한 통신 : Last Mile(마지막 1단계만 전화선을 통신 선로로 이용), 네트워크 코어는 따로 존재(음성 정보와 데이터가 분리되어 처리됨 – 더 높은 주파수의 데이터 처리 가능.) , 전화기와 컴퓨터가 연결된 전화선 사이에 Splitter가 존재, 데이터 전송률 = 1~8Mbps
     5) 케이블 모뎀 : 케이블 TV 선로(동축선)를 통한 통신, 데이터 전송률 = 30Mbps, 하나의 케이블 선로를 여러 집에서 공유하기 때문에 데이터 전송률은 전화선보다 낮을 수도 있다. Shared Channel

6) FTTH(Fiber To The Home) : 집 앞까지 광 케이블(기존의 구리선을 이용하여 전기 신호를 전송하는 대신 광섬유를 이용하여 가시광선과 같은 전자기파 신호를 전송)을 끌어오는 방식, 데이터 전송률이 높은 대신 설치 비용이 비싸다는 단점이 존재. 구리선에 비해 잡음이 적다. 국내에서는 활발한 통신 방법

   * 파장이 짧고, 주파수가 높을수록 데이터 전송률이 높다.

7) 기타 Last Mile 통신 기법 : 와이파이, 이더넷(랜선을 이용한 네트워크), 3G, LTE 등

3. 통신 선로의 종류

   1) 구리선 (예: 전화선 – 일반 pair 선/두 개의 선을 일렬로 , 이더넷 – Twisted pair 선/두 개의 선을 꼬아서 , 케이블 TV 선 – 동축선/-선은 원통으로 만들고 원통 정 가운데에 +신호선을 위치시킴)

   2) 광섬유 : 단위 길이 당 전송률의 가성비가 더 높다, 잡음에 강하다, 반영구적

   3) 무선 (공기) : 공유채널, 잡음이 많음 -> 실제 전송률은 떨어질 가능성이 있다.

4. 음성 정보 vs 데이터 정보 (= circuit switching vs packet switching)

   1) 음성 정보 – circuit switching 시스템에 적합, 전송 량의 예측 수준이 높다. (시간 당 전달될 수 있는 음성의 양의 한계가 정해져 있기 때문.)

   2) 데이터 정보 – packet switching 시스템에 적합, 전송 량의 예측 수준이 낮다.

   3) ↓ ↓ ↓Circuit switching과 packet switching 시스템의 차이↓ ↓ ↓

   4) Circuit switching : 시작과 끝이 명시적으로 구분. 꾸준히 정보가 전송됨. 시작부터 끝까지 선로 자원을 점유. 선로를 점유하는 방식으로 시분할 방식과 주파수 분할 방식이 있다. 초기화 비용이 높게 든다. 과금이 시간 단위로 이루어진다. 부하가 낮다. (한번 선로가 점유되면 처음에만 송신자와 수신자 정보를 보내면 계속해서 동일한 선로를 이용하기 때문에 다음부터는 송/수신자 정보를 보낼 필요가 없다.), => 통신의 시작에서 끝까지 통신을 위한 경로와 자원을 사전 할당하고 독점하는 방식. 할당된 경로는 ID를 부여하여 경로를 구분한다.

   5) Packet switching : 시작과 끝이 모호함. (예. 브라우저를 닫았다고 해서 사이트에 로그아웃이 되지 않을 수도 있음.) 짧은 요청과 반응으로 주로 이루어졌기 때문에 선로를 점유하는 방식의 통신이 낭비. 우편 시스템과 유사.(우편을 대상에게 보내면 요청이 끝나는 시스템) 패킷에 송신자와 수신자 정보를 담아 보냄. 패킷 당 부하가 높다. (모든 데이터 패킷에 송/수신자 정보 필요) 초기화 비용이 낮음. 패킷 당 과금/ 데이터 량 과금 방식. 자원을 독점하는 형태가 아니기 때문에 자원 사용률이 높다. => 패킷마다 목적지 주소를 적어서 보내는 방식. 패킷마다 할당되는

경로와 자원이 다룰 수 있음.

** packet : 네트워크에서 데이터 전송의 단위.

5. 프로토콜의 의미

1) 통신 규약을 의미한다.

- 계층적인 구조(복잡한 문제를 해결할 수 있는 가장 효율적인 소프트웨어 제작 패턴의 방법.)를 띈다.

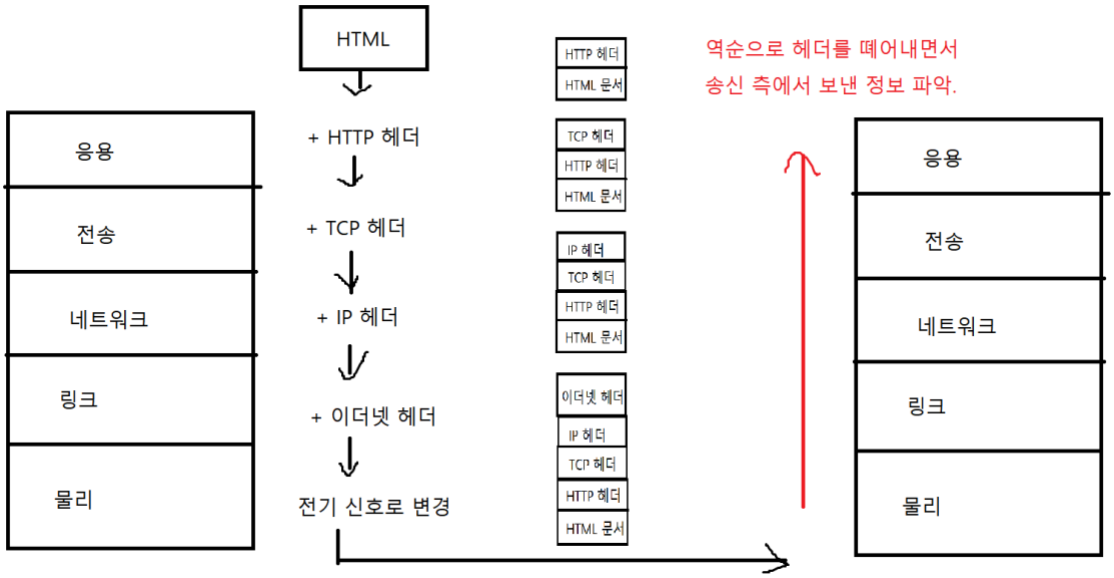
2) ISO(규약을 정의하는 국제기구)의 네트워크 7 계층 참고 모델 : 네트워크 프로토콜이 통신하는 구조를 7계층으로 분리하여 각 계층 간 상호 작동 방식을 정해 놓은 것.

- 응용 계층 : 사람이 사용하는 네트워크 응용 시스템 (예. 카톡, 유튜브 등.), 분산된 시스템을 하나의 통합된 응용 시스템으로 묶어주는 계층.
- 표현 계층 : 프레젠테이션 계층, 응용 계층의 표현 방식. (예. HTML, OpenGL, DirectX 등) 분산된 응용의 표현 방법에 대한 규약. UI 표현 방법.
- 세션 계층 : 응용/ 표현 계층의 객체를 전달하기 위한 계층 (예. HTTP – 통신의 표준), 통신에 대한 직접적인 계층. 응용의 접속/객체 통신(객체 정보를 전달)에 관련된 방법에 대한 규약.
- 전송 계층 : 트랜스포트 계층, 양 끝 단의 전송 품질 보장. 양 끝 단의 전송 품질 차이를 줄이는 역할. 신뢰성 제공.
- 네트워크 계층 : 단말기에서 단말기까지의 경로 설정. 전송 계층에서 신뢰성을 보장해주기 때문에 중간에 패킷이 소실될 위험이 존재. (네트워크에서는 패킷 재전송이 가능하기 때문에 전송 계층에서의 신뢰성을 위해 전송 계층 아래의 모든 계층이 신뢰성을 가질 필요가 없다.)
- 데이터(링크) 계층 : 각 경로에서 링크의 규약을 정의. 다음 단계까지의 전달을 책임. 네트워크 계층에서 설정된 경로에 따라 노드에서 다음 노드까지 갈 때의 전송 방법을 다룸. (예. WIFI, LTE,등.. 하나의 기기에서 기지국까지의 전송 책임..)
- 물리 계층 : 전송에 사용되는 하드웨어/장비.

3) 인터넷 5 계층 : 응용계층(응용 + 표현 + 세션), 전송계층, 네트워크 계층, 물리계층 -> 전송계층 이후로는 패킷 전송에 관한 부분만 다룬다. 응용계층에서는 라이브러리를 제공, 제공된 라이브러리를 활용하여 개발하지만 전송계층 이후로는 커널에서 제공되는 서비스.

** 클라우드 컴퓨팅 개념 : 유저가 서버에 데이터를 저장/처리하는 데, 저장/처리하는 곳의 정확한 공간을 모름. 클라우드처럼 여러 컴퓨터에 데이터를 저장/처리하도록 명령하기 때문.

** 커널 : 하드웨어적으로 보호되는 프로그램. 운영체제의 핵심 서비스. 외부의 해킹 공격으로부터 시스템을 보호할 수 있도록 운영체제에 탑재되어 있는 보안 기능.



- 각 해당 Layer의 헤더를 제외한 나머지가 본문이 된다.
- 이더넷 헤더는 링크가 변경될 때마다 바뀌지만 나머지는 변경되지 않는다.

6. 응용계층 (응용 + 표현 + 세션)

- 응용이란 ? 게임, 브라우저를 이용한 응용(검색, 웹툰, 뉴스, 블로그, 카페, 금융, 쇼핑 등..), 메일, SNS, 구글 드라이브와 같은 파일 서비스, 유튜브와 같은 미디어 서비스, 구글 지도, skype 등. 네트워크로 연결된 컴퓨터들이 협력하여 제공하는 서비스. 여러가지의 프로세스로 구성되어 있다.
- 네트워크 응용의 구조

  1) 클라이언트 – 서버 구조 : 응용을 서비스해주는 컴퓨터를 따로 가진 구조. 서버가 계속 운영되고 있기 때문에 원하는 시간에 원하는 정보를 얻을 수 있게 함. 웹 검색이 대표적인 예시. 정보를 전달할 때 서버가 관여하기 때문에 서버 부하가 높아질 수 있다. 안정된 서비스 제공 가능.

  2) Peer To Peer(P2P) 구조 : 원하는 정보를 제공하는 사람이 없으면 정보를 얻을 수 없음. 서버라는 응용 없이 직접 데이터를 주고 받음. 토렌트가 대표적인 예시. 정보 제공자의 신뢰를 보장 불가.

     - 단점 : 제공되는 서비스가 보장되지 않는다, Peer들은 IP를 바꾸어 옮겨 다닐

수 있다. 제공하는 Peer가 없을 수 있다. 신뢰성 보장 어려움 -> 해시코드를 활용하여 해결할 수 있다. 이기적인 시스템(받기만 하는 Peer가 존재할 수 있음.) -> 파일 공유를 많이 하는 Peer에게 점수 부여하여 파일 전송 속도를 조절. 패킷 조각들의 쏠림 현상. (모두 동일한 조각들만 소유하고 있는 경우 존재.)

3) 하이브리드 구조 : 필요할 때는 서버가 관여하지만 서버의 관여가 필요 없는 정보의 경우 클라이언트끼리 전달할 수 있도록 함. SKYPE가 대표적인 예시. 접속에 관련된 정보(메타 정보)는 서버가 관여해서 기록하지만 통화 중에는 서버가 관여하지 않음. 서버의 부하를 줄일 수 있다.

- P2P 구조의 단점을 보완하기 위한 구조.

** 부하 : 시스템에서 원하는 효과를 얻기 위해 취하는 행동에 필요한 동작이나 자원.

- 프로세스 간 통신 기법

** 프로세스 : 하나의 컴퓨터에서 독립된 메모리 공간을 가지고 수행되고 있는 프로그램의 단위.

1) 프로세스의 종류 : 서버 프로세스/클라이언트 프로세스, 피어 프로세스...

- 프로세스의 주소 = IP + 포트 번호

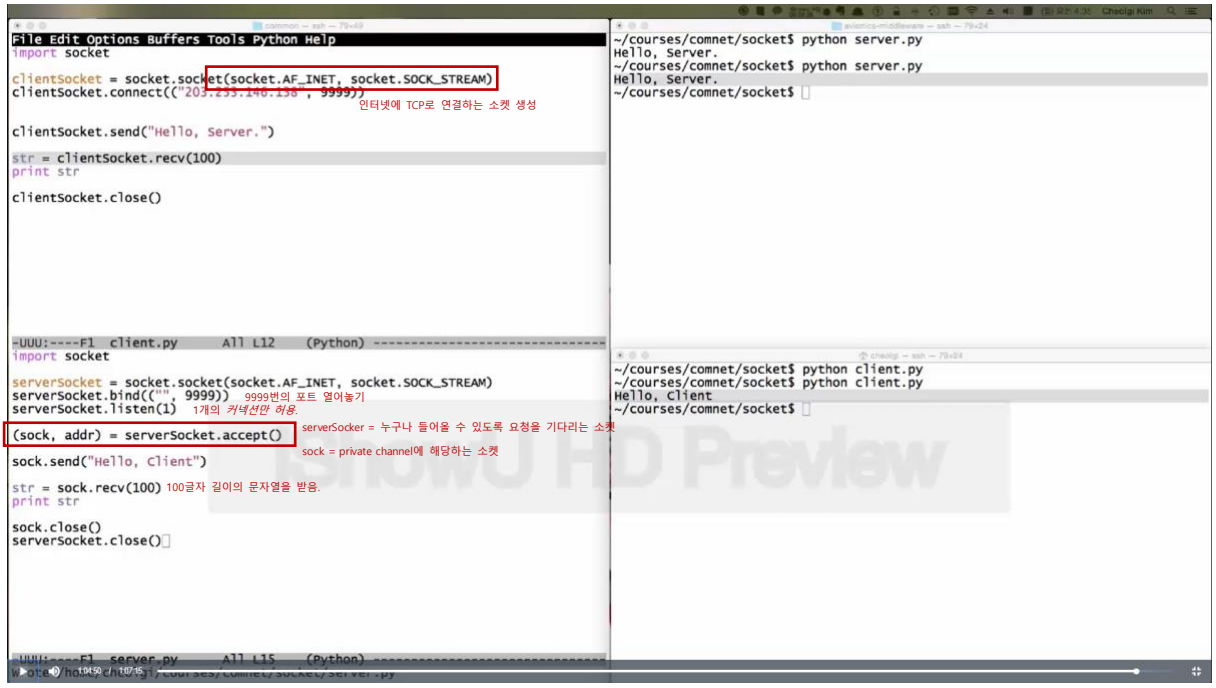
1) 기계의 주소 : IP (xxx.xxx.xxx.xxx – xxx : 0~255사이의 정수 값) – 약 40억개, 하나의 IP가 여러 기기를 가리킬 수도 있다

2) 포트 번호 : 패킷을 전달받을 프로세스를 구분하는 번호, TCP, UDP 각각 2^16개의 포트 번호가 열려 있다. 웹 서버의 경우에는 정해진 포트 번호가 있지만, 웹 브라우저 같은 경우 지정하지 않고 비어 있는 임의의 포트번호를 할당 받는다.

- 소켓 통신

1) 소켓이란 ? 클라이언트에서 서버까지의 private channel을 형성하는 통신 API, 패킷을 주고 받을 때마다 IP와 포트 번호를 계속해서 전달해주지 않아도 통신이 가능하게 한다.

2) 소켓 – 전송 계층과 세션 계층을 연결해주는 인터페이스.



- 세션 계층

1) 세션 계층 : 컴퓨터끼리 대화하는 방식의 내용.

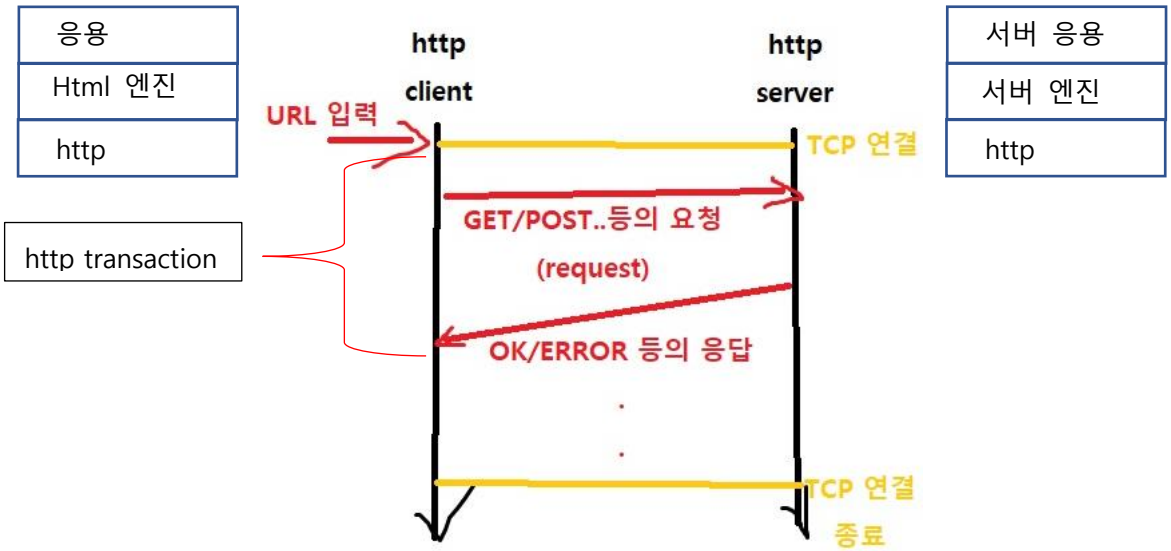
2) FTP (File Transfer Protocol)

- 목적 : 원격 호스트에 있는 파일을 가져 오거나, 원격 호스트의 파일을 보내기 위한 프로토콜.
- 가정 : 클라이언트 사용자는 서버의 파일을 읽을 수 있는 권한의 계정이 존재.
- 가정 : 계정이 없는 경우도 접근 가능하도록 설정 가능. – 권한이 있는 계정이 존재할 때보다 제한된 기능을 수행 가능.
- 특징 : 세션 로그인과 종료가 존재(HTTP와 가장 큰 차이점), 로그인을 통해 권한을 획득, 로그인 정보를 서버가 관리, Stateful Protocol – 사용자의 현재 상태 정보(사용자의 현재 작업 정보- 위치, 상태, ID 등)를 서버가 트래킹 하여 적절한 서비스 제공.

3) HTTP

- www(월드 와이드 웹)을 위한 세션 프로토콜
- HTML을 기본 표현 계층으로 사용.
- 목적 : 전세계 인터넷에 있는 정보를 탐색.
- 발명 : Tim Berners-Lee가 CERN에서 1980년대 말 ~ 1990년대 초

- 동작 : 웹 자료를 가져와서(GET) 보여주기, 웹 자료를 포스팅하기(POST)
- URL 입력 -> 클라이언트가 서버에 요청(GET) -> 서버가 요청한 자료 전송.
- TCP 기반 네트워크인 경우, TCP 연결과 TCP 연결 해제를 수행해준다.
- 특징 : Stateless Protocol(이전 요청에 대한 상태를 기억하지 않는다. -> 로그인 정보를 저장하지 않음.), 단순한 구조(요청에 대한 응답을 완료하면 서버의 역할이 완료.)로 되어 있기 때문에 다양한 응용들을 얹을 수 있게 된다.



- GET 요청 : 원하는 자원을 가져오기 위한 목적을 갖는 요청.

  * GET + 서버 상의 자원의 위치 + 버전

    + 매개변수 1 : ... (ex) host : formal.kau.ac.kr

    + 매개변수 2 : ... (ex) user-agent : Mozila/5.0

  http 요청 헤더

  한 줄 띄기 : 헤더의 끝

- GET에 대한 서버의 응답

  * HTTP/1.1 200 OK

  + Last-modified : date..

  + content-length : 487..

  헤더

  한 줄 띄기

  +실제 자원 (파일)

  몸통 – 한번에 한 개의 파일만 전송 가능.

- POST 요청

  - 자원을 서버에 게시하고 싶을 때.(예 : 게시판에 글 올리기, 인스타그램에 사진 올리기, 웹 브라우저를 이용한 이메일 보내기, 로그인 등)

  - GET 명령어로도 로그인 기능을 수행할 수 있지만, url에 로그인 정보가 붙기 때문에 보안에 취약하다. 또한 게시할 내용이 짧은 경우에만 가능한 방법이다.

  (사진과 같은 긴 내용의 경우에는 GET을 통해 전달이 불가하다.)

  -> GET /comnet?id=kau&pw=kau&action=login HTTP/1.1

  -> URI : http://formal.kau.ac.kr::8080/comnet?id=kau&pw=kau&action=login..

  URI : 인자에 따라 동적으로 생성되는 자원

  ** URL은 정적으로 할당된 자원 <- ?로 붙은 인자가 없다.

  - GET 명령어에서는 받아올 목록을 하나씩 받아오는 반면, POST 요청에서는 서버에 올릴 정보를 한꺼번에 첨부할 수 있다.

  * POST /comnet/ HTTP/1.1

    Host : formal.kau.ar.kr

    User-name : Mozilla/5.0

    올라갈 정보 첨부

  헤더

  한 줄 띄기

  몸통

- URL (Universal Resource Locator)

  * http://formal.kau.ac.kr/comnet/

  -> http://formal.kau.ac.kr/comnet/index.html

  -> http://formal.kau.ac.kr:8080/

  포트번호

  1) http: - 세션 프로토콜 이름

  2) //formal.kau.ac.kr – URL의 시작을 의미하는 // + 호스트 이름(작은 개념 ->큰 개념)

  3) /comnet/ - 원하는 자원(리소스)의 위치. (/로 끝나는 경우 디렉토리임을 의미.) -> http에서는 일반적으로 디렉토리를 자원으로 지정하면, 디렉토리가 지정하는 대표적

인 웹페이지를 보여준다.

4) 포트번호를 생략할 수 있는 이유 : 많은 사람들이 자주 사용하는 시스템의 경우에는 포트번호가 지정되어 있기 때문.

5) 도메인을 포함한 경로. (프로토콜 + 도메인)

- cookie

1) HTTP의 stateless 특성을 보완. 서버 응용이 클라이언트의 이전 작업 정보(state)를 파악하기 위한 도구.

2) cookie의 프로토콜

* HTTP Response 헤더(서버->클라이언트)에 Set-cookie라는 필드 존재.

* HTTP Request 헤더(클라이언트 -> 서버)에 cookie 라는 필드 존재.

3) set-cookie 필드를 통해 클라이언트에 cookie 정보가 저장되어 있다.

4) 클라이언트 응용계층의 응용에 쿠키 저장소가 있다. 세션 계층은 stateless 형태.

5) 클라이언트에게 너무 많은 권한/정보를 오픈하는 문제점이 존재. -> 해결책 : 세션

6) 세션 : 클라이언트에게 세션 id 정보만 쿠키로 전달하고 세션들의 특성은 서버가 관리하는 방식.(서버가 세션id를 통해 미리 저장해둔 클라이언트에 대한 정보를 파악.)

- HTTP 성능 지표

** 성능 산출을 위해 고려해야 할 점

= 지연시간 + 전송률 + 전송 받고자 하는 정보의 크기 + HTTP 헤더의 크기 + TCP 연결 및 종결 시간

1) HTTP Transaction의 성능 향상

* Persistent HTTP : 하나의 트랜잭션이 끝나고 바로 TCP 연결을 종결하지 않고 모든 요청이 끝난 뒤 TCP 연결을 종결 시킴.

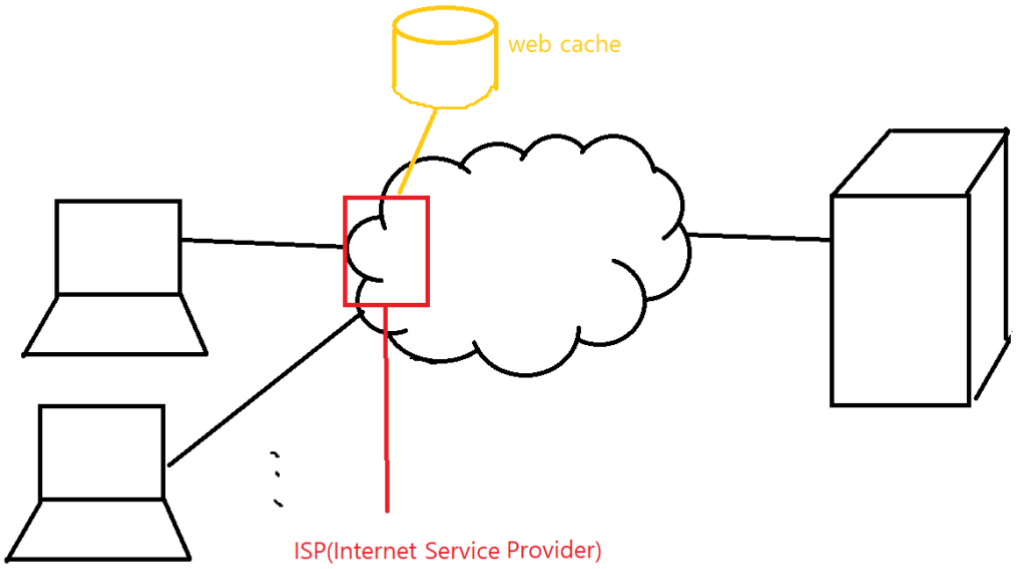
* Pipeline HTTP : 여러 요청을 하나의 패킷에 보냄.

2) 네트워크 구조 상의 성능 향상

* web cache (proxy server) : ISP에서 비용절감을 위하여 이전에 가져온 적이 있는 문서를 DB에 임시 저장해 놓았다가 동일 문서가 다시 요청될 때, 재사용. 클라이언트 입장에서는 서버 역할(정보 제공)을, 서버 입장에서는 클라이언트 역할(정보 요청)을 한다.

-> web cache가 주는 이득 : 요청 응답 시간을 절약, 제공하는 교통량 증가(절약된 교통량만큼 더 많은 사용자의 교통량을 수용할 수 있다.), 더 적은 서버 용량으로 더 많은 사용자를 지원 가능.

-> web cache가 서버의 응답을 받았을 때, 문서가 수정된 시간도 받게 되는데 클라이언트가 정보를 요청했을 때, web cache는 서버에 문서가 저장된 수정 시간 이후에 변경된 부분이 있는 지 확인 후 클라이언트에게 최신 정보를 전달한다.



- DNS(Domain Name Service/Server)

  1) Internet에서의 ID

     - IPv4 : 32bit(8bit*4) -> 현재 주로 사용 중인 IP : IP 주소의 개수가 부족하기 때문에 존재하는 IP 주소를 여러 기기가 공유하는 경우가 많다. 웹에서는 html 정보와 이미지 정보를 받는 IP주소가 다르더라도 웹 브라우저에 저장된 쿠키 정보로 클라이언트를 구분하기 때문에 동일한 기기로 정보 전달이 가능하다.
     - IPv6 : 128bit(16bit*8) -> 개발되어 있지만 IPv6를 위한 네트워크 기기(라우터 등)로 변경하는 데 발생하는 비용과 대부분의 ID가 v4로 되어 있다는 문제 때문에 거의 사용하지 않는다.

  2) 순수하게 인간을 위한 시스템 -> IP 주소를 이름으로 기억.

  3) 응용계층(인터넷계층)에서만 관여/사용.

4) 도메인 이름을 IP 주소로 변경해 주는 서비스

5) Aliasing(같은 곳에 대해 여러 주소를 사용 가능하게 함.) 관리

Ex) 같은 기기에 여러 도메인을 할당.

6) 전우주적으로 중앙 관리된다. (중앙에서 도메인-IP 매핑 수행.)

7) 계층적 분산 관리를 통해 중앙의 부하를 줄임.

- DNS의 계층 (Root NS – Top Level DNS – Authorized DNS – Local DNS)

1) 최상위 계층 : Root Name Server

- Top-Level Domain Name Server(예. .kr, .com, .net, .org, .edu..) 관리
- 탑 레벨 도메인 서버에 IP를 유효기간과 함께 반환. -> 유효 기간동안 동일한 탑 레벨 도메인 서버에 대한 IP를 요청하지 않게 된다.

- DNS 계층에 접근하는 방법

1) Recursive

- Local -> Root -> Top Level -> Authorized -> Local...
- 현재는 사용되지 않음.

2) Iterative

- Local 에서 Root로 가면 Root는 Top Level의 위치를 알려줌.
- Local 에서 Root에서 받은 Top Level로 가서 Authorized의 위치 받아 옴.
- Local 이 Top Level에서 받은 Authorized의 위치에서 해당 DNS의 ip주소를 받아옴.

7. 전송 계층

- 신뢰할 수 있는 네트워크(단말 간에 데이터를 전달할 때, 모든 데이터가 세는 것 없이 전달됨.)를 end to end로 전달할 것인지에 관한 계층.
- 포트번호 담당.
- 네트워크 현실에서의 문제점

1) 데이터를 끊임없이 제공할 수 없음. 한 번에 전달할 수 있는 데이터 양이 정해져 있음. 무한한 흐름이 아님. -> 데이터를 패킷 단위로 전달.

2) 패킷이 유실될 수 있다. -> 지정된 시간 내에 ACK 패킷(전달한 패킷을 받았다고

응답하는 패킷)이 전달되지 않으면 패킷을 재전송.

- A에서 B로 전달할 패킷이 유실되는 경우
- A에서 B로 패킷이 잘 전달되었으나, B의 ACK 패킷이 유실되는 경우
- A에서 ACK 패킷이 전달되기 전에 타이머가 만료되는 경우

3) 패킷의 순서가 바뀔 수 있다.

4) 패킷 자체가 변조될 수 있다.

5) 2), 3)문제 해결방안 : 패킷에 번호를 붙여 전달하는 패킷의 종류를 구분한다. 재전송된 패킷을 구분함으로써 중복된 패킷을 구분. -> 성능이 낮아지는 문제점 존재(ACK 패킷을 전달받기까지의 지연시간이 존재.) -> 성능향상을 위해 파이프라인 기법을 사용.

- 성능향상 기법

1) 파이프라인 기법 : 연속된 대량의 작업이 순차성을 갖고 있으나, 앞의 일이 종료하지 않고도 다음 일을 시작할 수 있는 병렬성을 가진 경우의 성능향상 기법.

- Go-back-N : 최대 N개의 패킷을 병렬적으로 처리, 송신 측에서는 N개의 패킷을 버퍼링(=아직 수신이 확실하지 않은 패킷에 대하여 재전송을 위해 버퍼에 보관.) 수행, 수신 측에서는 순차적으로 잘 수신된 패킷에 대해 ACK을 송신하고 패킷의 payload(=패킷 내의 실제 전송하고자 하는 내용)를 응용계층으로 올려 보냄, 송신 측에서는 ACK을 받은 후 버퍼에 여유가 생기면 추가로 패킷을 버퍼에 올린다.(파이프 라이닝을 수행한다.)

  ➔ 수신 측에서 순차적으로 ACK 패킷을 보냈다고 여기기 때문에 버퍼 내에 존재하는 중간 패킷에 대한 ACK 패킷을 받지 못해도 그 다음 ACK 패킷을 받으면 중간 패킷에 대한 ACK 패킷도 받았다고 처리한다.

  ➔ 수신 측에서 순서에 맞지 않는 패킷이 온 경우의 반응 (2가지)

  + 조용히 있는다.

  + 잘 받은 마지막 패킷에 대한 ACK를 전송. (그 다음 패킷에 대한 ACK는 전송하지 않는다.)

  ➔ Go-back-N에서의 재전송 정책

  + 각 패킷 전송 시에 패킷을 위한 타이머 설정.

  + ACK을 받으면 ACK 해당 패킷과 앞쪽 패킷에 대한 타이머 소멸.

+ 타이머 이벤트 발생 시, 해당 패킷부터 재전송.

+ k번째 ACK 패킷이 반복적일 경우, k+1번째 패킷의 유실을 의미. -> 3번 정도의 k번째 패킷에 대한 ACK가 오면 타이머와 무관하게 k+1번째 패킷을 재전송.

➔ 장점 : 단순(특히 수신 측 – 수신 측에 대한 버퍼 존재 X), 간결하게 시스템의 상태가 추상화됨.

➔ 단점 : 패킷 유실에 대한 복구 비용이 높다.(하나의 패킷 유실에 대해 그 다음 패킷에 대한 재전송도 수행해야 하므로..)

- Selective Repeat : Go-back-N의 단점을 보완한 파이프라인 기법. Go-back-N과 달리 수신 측에도 버퍼 존재. 송신 되지 않은 패킷 이후의 패킷이 잘 전송된 경우, 응용계층에 가기 전 버퍼에 저장. -> 빠진 패킷이 추후 도착하면 버퍼에 저장한 이후 패킷들까지 순차적으로 응용계층에 전달.

  ➔ 버퍼는 연속적인 패킷에 대한 개수를 지정하기 때문에 2번 패킷이 제대로 송신되지 않고 3,4번 패킷이 제대로 송신 되었다고 하더라도 버퍼에서 3,4번 패킷을 제거하지 않는다.

  ➔ 장점 : 실패한 패킷만 재전송(성능이 향상)

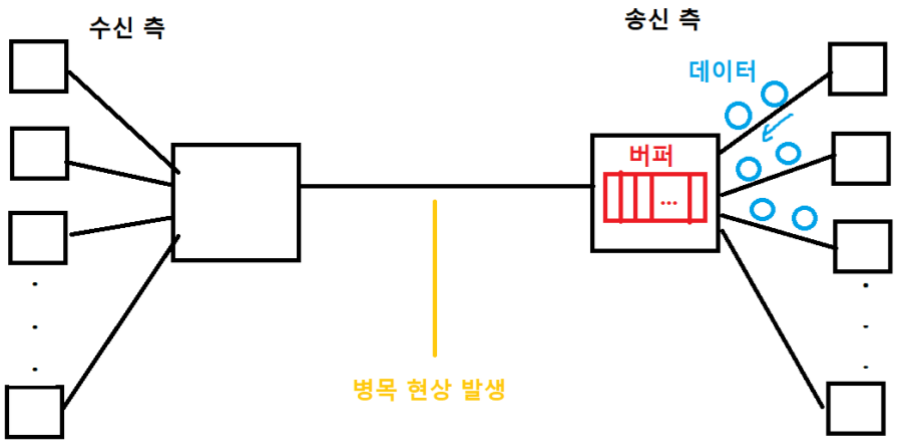
  ➔ 단점 : 시스템의 상태 추상화가 복잡(간단하게 설명 불가.), 수신 측에도 버퍼가 필요함.

- 전송 계층 서비스

  1) TCP

  - 연결 기반의 전송계층 : 프로세스가 가상의 연결을 통해 정보 전달.
  - 연결에 대한 관리 수행 – 데이터 유실 복구 기능을 수행해줌, 데이터에 대한 순서를 보장해줌, 혼잡에 민감한 패킷 전송률을 조정함(네트워크가 혼잡할 경우, 패킷의 전송 속도를 늦춤.)
  - 패킷의 유실이 없도록 보장해주는 전송계층. -> 잘 전송된 내용 중간에 빠진 내용이 없다. (=내용의 이빠짐 현상이 없다.)
  - 패킷 전송 순서의 신뢰성 보장. -> 시퀀스 넘버, 타이머, ACK 를 이용.
  - 고의적 지연(내용의 이빠짐 현상이 발생할 경우, 응용계층으로 연속적으로 전달된 패킷만 전달한다.)이 존재한다. UDP에 비해 속도가 느림.
  - 고의적 지연의 이유 : 패킷 수신 순서, 혼잡제어(네트워크 혼잡 시 송신할 패

킷을 보내지 않을 수 있다.)



<네트워크 혼잡>

병목 현상이 발생함에 따라 송신 측 버퍼에 데이터를 쌓아 놓음.

버퍼도 크기가 제한되어 있어, 버퍼가 찬 경우 데이터 송신을 제재할 수 있다.

- 전송단위 : 바이트 단위로 데이터 전송. -> 내용 유실이 없기 때문에 특정 크기의 데이터 단위 만큼 기억해서 내용 유실을 파악할 이유가 없다. 따라서 send() 함수와 recv() 함수에서 주고받은 내용의 전송 단위 별 일치관계를 따지지 않음.
- 내용 변조(악의적인 변조 외의 기계적인 문제에 의한 변조)방지 -> 체크섬 비트 활용, 해시코드(역함수가 존재하지 않아 역함수로 암호를 추적할 수 없게 만드는 코드) 활용.
- 흐름 제어 : 수신 측에서 패킷을 받아들일 공간이 없을 때, 송신 측에서 인지하고 패킷 전송을 중단.
- 포트(사서함의 개념과 유사)개념 지원. -> 응용 프로그램을 구분 시킴.

  ➔ 소켓 vs 포트

  소켓 여러 개가 하나의 포트번호에 연결될 수 있다. (다대일 매핑 관계)

2) UDP

- 무연결 전송계층 : 각각의 프로세스는 소켓을 열고, 열린 소켓을 통해 정보 전달. TCP와 달리 연결되어 있지 않기 때문에 패킷에 계속해서 송수신 주소

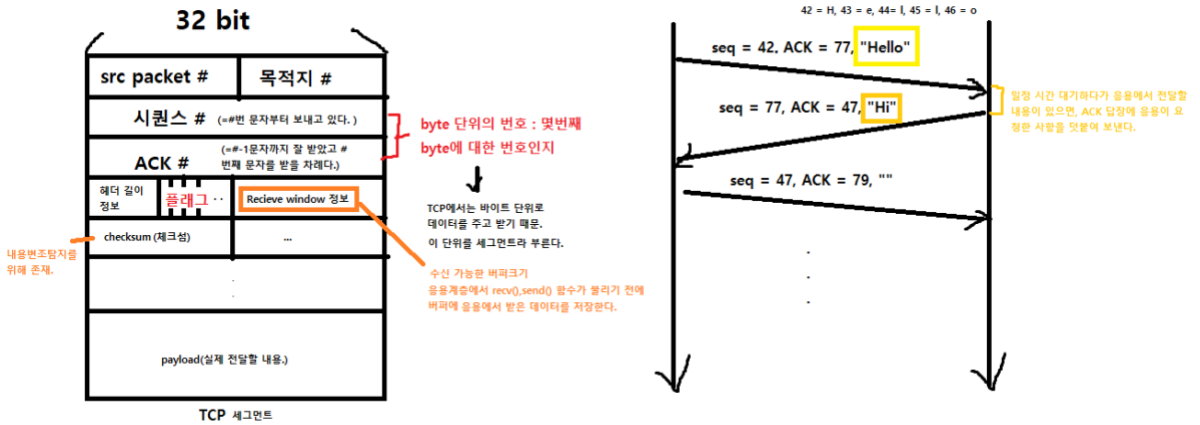
값을 입력해 주어야 한다.

- 데이터 유실 가능성 존재, 데이터 순서가 역전될 가능성 존재, 최대한 성능으로 패킷을 전송함. (전송 속도가 중요한 경우, UDP를 사용.)
- UDP에서의 소켓 통신 : 컴퓨터 상에서 하나의 포트를 점유.
- 내용 유실이 있을 수 있다.(내용의 이빠짐 현상이 발생할 경우에도 응용계층으로 전달한다.)
- 데이터 전송 순서를 보장하지 않음.
- 고의적인 지연이 존재하지 않는다. 속도가 빠름.
- 전송 단위 : 패킷(데이터그램) -> send()함수 호출 단위. send() 함수와 recv()함수에서 주고 받은 데이터를 전송 단위별로 대응 관계를 파악하여 패킷 유실을 확인한다. 따라서 각 단말에서 임의로 패킷을 다른 데이터 단위로 나눌 수 없다.

- **TCP** 헤더에 포함되는 내용

1) HTTP 헤더와 달리 문자열로 되어 있지 않고 바이너리 코드로 구현되어 있음.

2) 32bit로 끊어서 헤더를 분석. -> 맨 처음 16bit = 소스 포트 번호(송신 측의 포트 번호), 16bit = 목적지 포트 번호 -> 이를 이용하여 동일한 포트로 들어오는 소켓을 구분한다.



3) 체크섬의 기능 : 체크섬 영역을 제외한, 헤더와 바디(payload)영역을 체크섬 덧셈 수행한 결과에 체크섬 덧셈 결과를 반전한 비트(체크섬 비트)를 더해주어 모든 비트를 1로 만들어준다. -> 모든 비트가 1인 경우, 내용 변조가 일어나지 않은 것임.

- 체크섬 기능은 IPv4에서만 사용한다.

** 체크섬 계산 예

ⓐ 다음과 같은 4비트의 데이터가 있다. : 0x25, 0x62, 0x3f, 0x52

ⓑ 모든 바이트를 덧셈 : 0x118

ⓒ ⓑ의 결과값에서 캐리 니블(= 최상위 자릿수를 넘는 값)을 버림 : 0x18

ⓓ 0x18의 2의 보수 : 0xE8 = 체크섬 바이트

ⓔ ⓑ값에 체크섬 바이트를 모두 더한다. : 0x200

ⓕ 결과값의 캐리 니블을 버린다. -> 0x00 = 오류가 없음을 의미.

- 네트워크 혼잡 제어

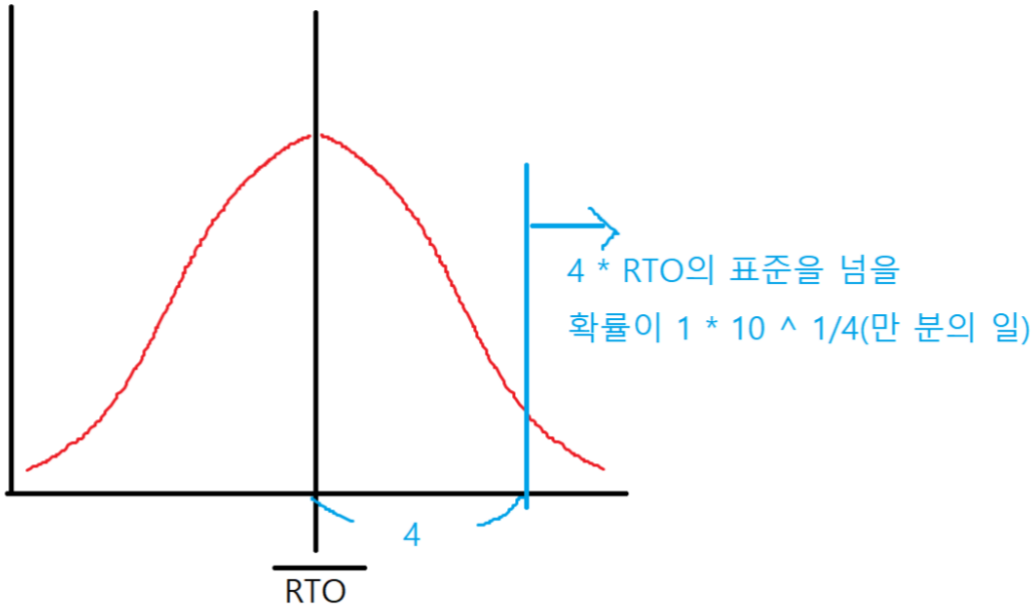
1) 혼잡 인지

- TCP에서 혼잡 인지 방법 : 패킷 유실이 발생한 경우(=ACK 신호가 타임 아웃 내에 오지 않은 경우), 혼잡 상황으로 인지.
- ACK 신호를 수신 받기 위한 타이머는 너무 길거나 짧아서는 안된다. -> 너무 길 경우, 재송신 지연이 너무 길어지는 문제가 생기고, 너무 짧은 경우 ACK 신호가 타이머보다 늦게 전송되어 잘 전송되었음에도 불구하고 전송되지 않았다고 인식할 가능성이 있다.
- 혼잡이 인지될 때, 패킷 전송률을 2배 낮춘다.
- 패킷 전송이 원활할 경우, 패킷 전송률을 현재 전송률 + c(상수, 세그먼트 하나의 크기)로 높인다.

  = AIMD(Additive Increase Multiplicative Decrease)

2) 재전송 타임 아웃(Retransmission Time Out = RTO)에서 타이머 시간 설정

- 패킷을 보내고 받는 시간(Round Trip Time = RTT) <- 네트워크 장비 내에서의 큐잉 지연이 가장 큰 영향을 준다.
- RTO = max (여러 RTT 평균값 + 4 * RTT의 표준편차 , 1초)
- RTT의 표준편차에 4배를 구해주는 이유



- 여러 RTT의 평균값을 구하는 공식

  : Weighted moving average를 이용한다.

  RTT(n) : n번째 RTT 측정값

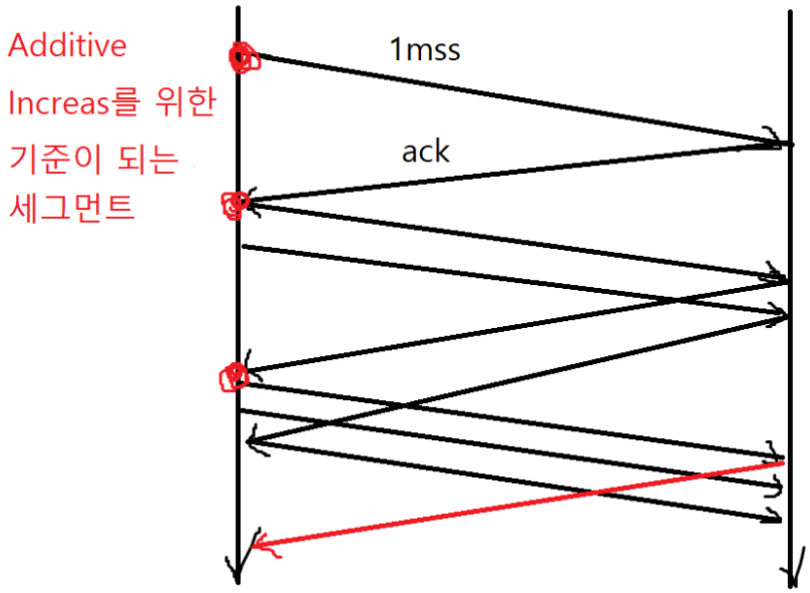
  ~~RTT~~(n) : n번째 샘플링 후 판단한 RTT 평균

  ~~RTT~~(1) = RTT(1), ~~RTT~~(n) = (1-a) * ~~RTT~~(n-1) + a * RTT(n) -> 현재에 가까울수록 더 많은 가중치를 갖게 된다.

3) 혼잡 제어

- AIMD 구현 방법

  ➔ Additive Increase : Increase의 단위는 MSS(=Maximum Segment Size) 이다. TCP로 연결된 장비 간의 MSS 크기는 달라질 수 있다(장비 간 합의 하에 더 작은 MSS를 채택). RTT에 도달하기 전에 패킷 전송이 성공했을 때, Conjection window 크기를 n * MSS에서 (n+1) * MSS로 늘린다.(=MSS 크기의 데이터를 동시에 n+1개 보낼 수 있다.)



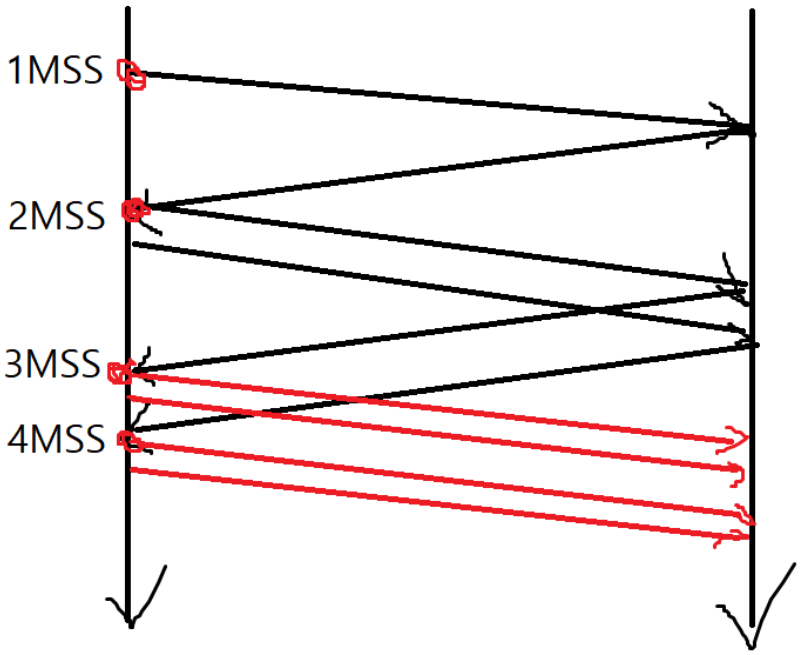
➔ 기준이 되는 세그먼트 단위가 ACK 신호를 받았을 때만 동시에 보낼 수 있는 MSS 단위를 증가시킨다.

➔ Multiplicative Decrease : 패킷 유실 시, Conjection window 크기를 절반으로 줄인다. 패킷 유실이 네트워크 이상을 반영할 때는 CW(=conjection window)를 1MSS로 줄인다.

➔ 단순 패킷의 유실과 네트워크 이상을 구별하는 근거 : 네트워크 이상 시 (예: 네트워크에서 중간에 장비가 사라지는 경우 모든 ack 신호가 송신되지 않는다.)에는 모든 전송 세그먼트가 동시에 유실된다. 단순 패킷 유실의 경우에는 유실된 세그먼트 이후에 보낸 다른 세그먼트의 ack 신호가 제대로 송신되게 된다.

➔ Slow Start 기법 : 네트워크 이상 시 CW= 1MSS로 변경될 경우, CW 증가가 느려지는 문제가 있다. (예: 기존 CW=100MSS인 경우에서 네트워크 이상이 생길 경우 1MSS에서 100MSS로 회복하기까지 많은 시간이 걸린다. 따라서, CW =50MSS가 될 때까지 multiplicative increase를 수행한다.)

slow start 모드



➔ 기준이 되는 세그먼트가 아닌 모든 세그먼트에 대한 ack 신호를 받을 경우, CW의 크기를 증가시킨다.

8. 네트워크 계층 (IP 계층)

- 주소와 라우팅(주소에 도달하기 위한 경로 설정)에 관한 계층.
- 전송 계층에서 전송이 잘 됐는 지 확인하기 때문에 IP 계층에서는 패킷 전송이 잘 됐는 지 확인하는 과정이 없다.
- **라우터(Router) : IP 패킷이 왔을 때 다음 경로를 설정하는 도구.**
- IP 주소 담당.
- DV(Distance Vector) 라우팅 방식

1) 각각의 노드가 서로의 일부 정보만 갖고 있는 상태에서 각각의 노드가 최적의 라우팅 경로 설정. -> 현재 노드와 다음 노드의 최소 비용 고려.

2) ISP 내부의 라우팅 -> 금전적인 최적화 > 기술적 최적화

3) 주변 정보만을 가짐.

4) 링크에 대한 정보가 변할 경우 최적의 경로를 설정할 수 없을 수도 있다는 단점

이 존재.

5) 장비가 다운될 경우, 점차 다운되는 장비 수가 늘어날 수 있는 단점 존재.

- LS(Link State) 라우팅 방식

1) 모든 노드가 네트워크 상의 모든 링크를 알도록 만든 뒤, 그 정보에 기반한 라우팅 경로 설정.

2) ISP 내부의 라우팅.

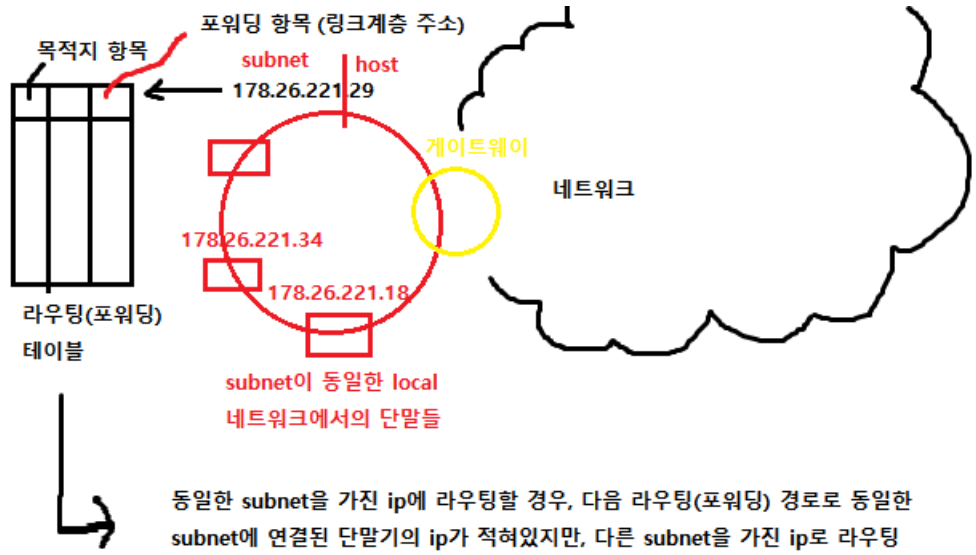
- IP 주소체계

1) IPv6 : 128 비트, 8비트를 16진수로 / IPv4 : 32 비트, 8비트를 10진수로

2) IP 체계는 하나의 IP를 가지고, 다양한 주소를 가질 수 있다는 특징 존재. 주소 비트가 한정되어 있기 때문.

3) Ip 주소 부여 방식

- 주민등록번호 체계 : 처음 기기 생성 시, ip 주소 부여, ip 개수가 부족하다는 문제가 있어서 유일하게 부여할 수 없다는 문제 존재, 라우팅 테이블에 기기의 주소를 일일이 기록해야 한다는 문제점 존재-> 개별 엔트리 수가 많기 때문에 테이블 관리 비용 증가, 연결 비용이 크다.
- 우편 주소 체계 : ISP가 변경되면 IP도 변경되어야 한다는 문제 존재, 고정 ip의 경우, 모든 컴퓨터에 설정을 변경해야 하는 문제 존재, ip 개수가 부족하다는 문제 존재.
- Classless Inter-Domain Routing(CIDR) : 계층적인 구조, 계층에 예외를 두는 것이 가능, subnet(ip주소체계(32bit)를 subnet 파트와 host 파트로 나눔, 라우팅 시, subnet이 다른 경우 subnet 단위로 라우팅하고 subnet이 같은 경우 단말 단위로 라우팅한다.)



➔ 조직(ISP, 회사, 학교 등)에서 ip를 쓸 경우 : 서브넷을 할당 받음.

Ex) 173.26.226.0/24(subnet의 비트 수) : MSB부터 24비트수를 통해 subnet 구분. Subnet 비트수가 클수록 우선순위가 높다.

** MSB (=Most Significant Bit) : 2진수에서 가장 큰 차이가 나게 만드는 비트

** LSB (=Least Significant Bit) : 2진수에서 가장 작은 차이가 나게 만드는 비트

Ex) 10110100

➔ Subnet의 조각화 : 조각화가 많이 이루어질수록 라우팅 테이블의 항목이 늘어난다.

Ex) 173.26.226.0/24 – 고양

-> 10101101 00011010 11100010 00000000

173.26.226.64/26 – 제주 : 64 ~ 127 비트까지는 제주로 향하는 ip

-> 10101101 00011010 11100010 01000000

4) IP 할당 방법

* ISP나 기관에서 ICANN(Internet Coporation for Assigned Names and Numbers)에 서브넷 영역 할당 요청.

* 서버 운영 시, static IP를 ISP나 기관에 요청하여 할당 받은 뒤, 기계에 수동으로 IP를 입력해준다.

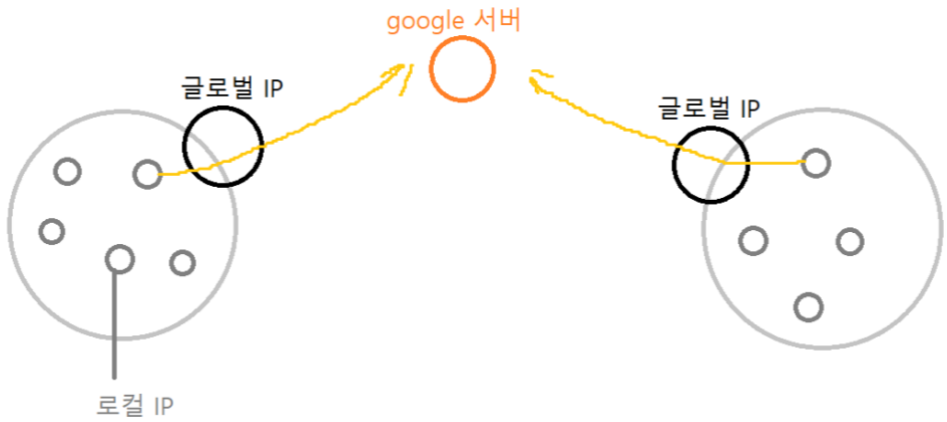
* DHCP(Dynamic Host Configuration Protocol) – 소스 address를0.0.0.0으로 할당

받아 UDP 프로토콜을 활용하여 255.255.255.255 IP(Broadcast라는 특정 목적을 위한 IP)에 IP 할당을 요청하는 패킷과 ID를 전송하면, 255.255.255.255 IP에서 브로드캐스팅 과정에서 모든 IP중 DHCP 서버에 도달하면 DHCP 서버가 남는 IP 중 하나를 골라서 255.255.255.255에 응답한다. 응답을 수신자의 ID와 함께 주기 때문에 해당 ID를 가진 수신자가 해당 응답을 받을 수 있다.

* **NAT(Network Address Transform) Router** : 부족한 IP주소를 위한 해결책 (무선 공유기= NAT Router라 생각하면 쉽다.)

-> 전송 계층과 네트워크 계층에 모두 속함.

-> 글로벌 IP내에 로컬 IP 존재하며, 글로벌 IP가 서로 다른 로컬 IP는 동일한 IP를 가질 수 있다. 로컬 IP에서 목적지로 패킷을 전달할 경우, 로컬 IP는 글로벌 IP로 변환되어 패킷에 작성된다.



-> NAT에서 로컬 IP로 사용 가능한 서브넷 : 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16

-> NAT 라우터에는 Port Mapping Table(Endpoint Independent Mapping)이 존재.
-동일한 글로벌 IP내의 다른 로컬 IP에서 동일한 포트번호로 다른 서버에 통신하기 위함. 동일한 포트번호를 가진 로컬 IP의 포트번호를 다른 포트번호로 매핑시켜준다. 하나의 로컬 IP에서의 하나의 포트번호에 다른 포트 번호를 1:1 매핑시켜 준다. ... 포트번호 개수에 제한되어 서비스할 수 있는 노드의 개수를 제한되는 한계점이 존재한다.

* Endpoint – dependent Mapping Table – Port Mapping Table의 한계점을 보완. 하나의 로컬 IP가 다른 서버로 접속하는 경우, 이전에 사용했던 동일한 포트번호를 사용할 수 있게 허용.

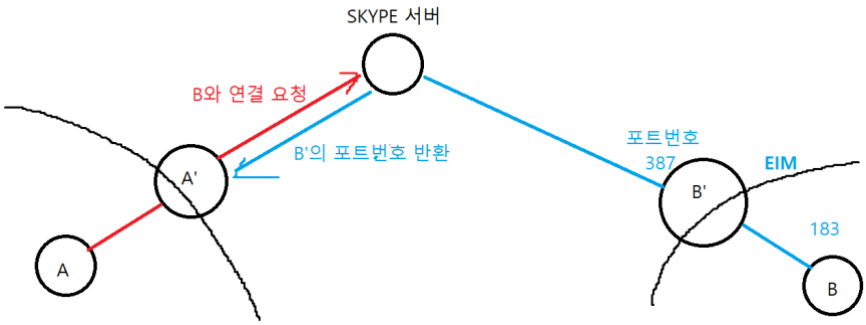
* NAT 에서의 P2P 지원

-> NAT에서느 로컬 IP끼리 직접 연결이 불가하다는 단점이 존재.

-> 방법 1 : Static mapping을 사용 – 사용자가 지식을 가지고 사전에 설정해야 한다는 단점 존재.

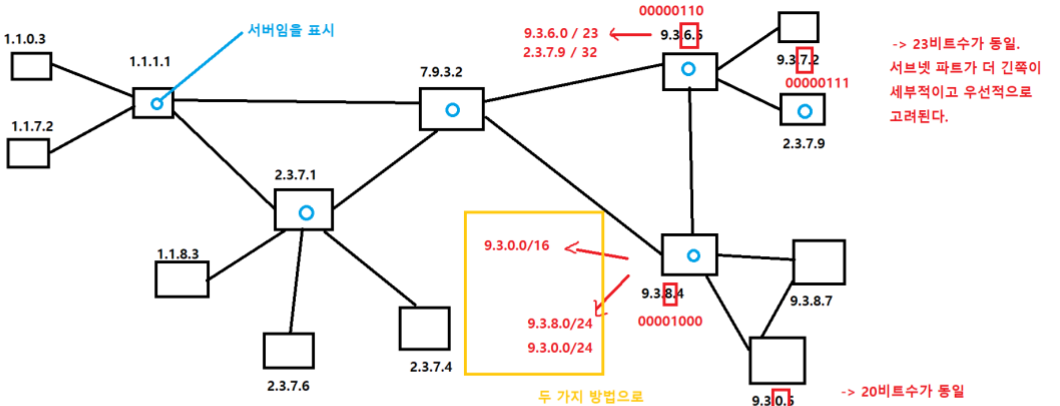
-> 방법 2 : NAT 내부의 노드가 사전에 Global IP를 갖는 Relay host를 지정 후, 서버 역할을 할 로컬 IP에 영구적으로 연결을 설정. EDM(Endpoint Dependent Mapping)과 EIM(Endpoint Independent Mapping)에 모두 사용 가능.

-> 방법 3 : Hole puncturing – 글로벌 IP를 갖는 서버와 NAT 내부의 클라이언트가 사전 연결을 수행 후, 각 NAT 라우터와 Port Mapping Table에 등록한다. 이후 노드가 연결을 시도할 때, 해당 Port번호를 알려준다. EIM에서만 사용가능.



- 응용과 링크계층은 네트워크 계층의 IP만 고려해서 프로토콜을 생성해주면 된다. -> 응용과 링크계층 각각에서 프로토콜이 변경되더라도 서로의 계층에 대한 고려 없이 네트워크 계층만 고려해주면 된다. / 네트워크 계층의 IP를 변경하는 데 어려움이 존재한다는 단점이 있다.

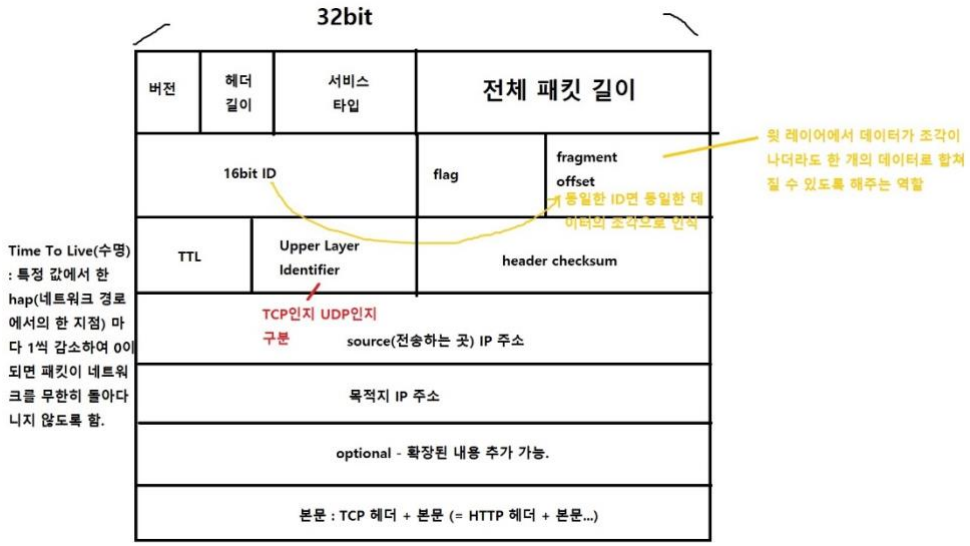
- 네트워크 경로 설정 방법 ↓



1.1.0.3의 라우팅 테이블

목적지		next
1.1.0.3 /4		-
0.0.0.0 /0		1.1.1.1

- IP헤더에 포함되는 내용



  - Fragment flag : 추가적인 fragment가 존재하는 경우 1로 변경. 예를 들어 이전 단말에서 fragment가 3번 수행된 경우, 첫번째 fragment의 fragment flag는 1, 두번째 fragment flag도 1, 마지막 fragment flag는 0으로 지정.
  - IP헤더의 기본 크기는 20B로 IPv6에서는 헤더에 optional 영역이 지정되어 있지 않고 본문에 optional에 들어갈 내용을 추가해준다.

9. 링크계층

- 네트워크에서 설정된 경로를 가장 효율적으로 도달할 수 있는 방법을 고려하는 계층, 한 합(단말에서 다른 단말까지의 연결)에서의 프로토콜(이더넷, LTE 등.) 결정.
- 

- 한 합에서의 링크이후 다음 링크로 갈 때, 프로토콜 변경이 발생 가능. (LTE -> 이더넷)
- 이더넷, LTE, WIFI, 3G.. 등이 있다.
- 라우터의 기능 : 서로 다른 링크 계층을 연결해주는 기능. 연결된 단말의 프로토콜이 서로 다를 때, 데이터 전송 단위가 다르다면 IP Fragment를 수행해준다.

- 데이터 링크 계층의 서비스

1) Error Detection, Error Correction

- Error Detection : 체크섬 + CRC으로 수행.

  ➔ 체크섬의 특징 : SW, 연산단위가 word, error detection이 CRC보다 약함.

  ➔ CRC : HW, 연산단위가 bit, 체크섬보다 성능이 좋음.

- Error Correction : Turbo Code, LDPC로 수행.

  ➔ 물리 계층으로 내려가는 경향이 있다. -> 데이터를 아날로그로 파악.

2) 채널 공유 프로토콜

- 채널 : 동시에 한 명만 사용할 수 있는 미디어의 단위. (미디어에 따라 채널의 단위는 주파수, 코드 등으로 달라질 수 있다.)

- 미리 자원을 할당 받아 놓는 방법.

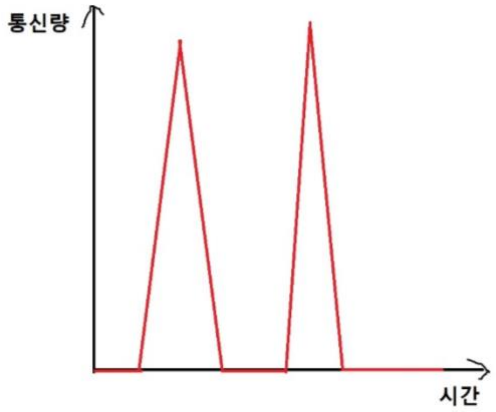
  ➔ 모든 단말에 동일한 기회 부여.

  ➔ 주파수 분할(FDMA=Frequency Division Multiple Access) – guard frequency(주파수의 간섭을 피하기 위해 일정 간격 이후에 주파수를 분할해 줌.)가 존재.

  ➔ 시분할(TDMA) – guard timer(수신 측과 송신 측의 물리적 거리 때문에 수, 송신 시간의 차이를 극복하기 위해 일정 시간 이후에 시간을 분할해 줌.)가 존재. -> 실제 분할 받으려는 시간보다 더 큰 시간을 분할해줘야 한다.

  ➔ 코드 분할(CDMA) – FDMA와 TDMA와 달리 분할하려는 것보다 더 큰 채널을 분할해 줄 필요가 없다.

  ➔ 단점 : burst 통신이 쉽지 않다.



- Carrier Sense Multiple Access(CSMA)

  ➔ Carrier를 아무도 사용하지 않을 때 통신.

  ➔ 단점 : 동시에 통신을 시도하는 경우, 파악이 어렵고 충돌 발생. -> 충돌이 잦으면 Random number Window 시간(통신을 시도하려는 단말에 랜덤한 대기 시간을 부여하여 동일하지 않은 랜덤 시간을 갖을 경우, 통신을 가능하게 하는 방식.)을 늘려준다.

- 토큰 방식

  ➔ 전달할 내용이 있을 경우, 토큰에 얹어 넘겨주고 그렇지 않은 경우 토큰만 넘겨주는 방식. 해당되는 수신자는 토큰에 얹어진 내용을 가져간 뒤, 토큰을 넘겨준다. -> 충돌을 방지하는 효과.

  ➔ 미리 자원을 할당 받아 놓는 방식과 CSMA방식의 절충안.

- Fall-Duplex **Switch**

  ➔ 스위칭 허브를 중간 지점에 놓고 패킷 수신 단말이 다를 경우 송/수신 단말을 서로 연결해주는 방법. 동일한 수신 단말을 요청할 경우, 요청된 패킷을 대기 큐에 쌓아 놓는다.

- 이더넷

  1) 규모의 경제학.

  2) 60년대 중반까지 CSMA/CD 방식으로 모든 노드를 wire로 연결. – propagation delay 문제 발생. 노드 수가 증가하면 그에 비례해 속도가 떨어진다는 문제가 존재했다.

  3) 단위 : Frame

  ● Frame 형식 = preamble : 목적지 주소(6byte) : 출발지 주소 : 상위 계층 프로토콜 타입 : 데이터 : CRC

  ● 스위치 내에서 다음 목적지(->하드웨어 주소=이더넷 주소)로 보낼 처리를 빨리 하기 위해 목적지 주소(->하드웨어 주소=이더넷 주소)가 출발지 주소보다 선행된다.

  ● preamble : 101010 ... 10101011 아무런 의미 없이 1과 0이 반복되는 것. 워밍업+타이밍 싱크 맞추는 역할.

4) HW 주소(=HW address = MAC Address)

- 6Byte(48bit) 주소
- 원칙적으로 전세계에서 유일해야 한다.
- 하드웨어 제작 시, 부여 받는 번호이다.
- 한 링크에 해당하는 영역에서 동일한 주소를 갖게 되면 문제 발생.

-> 한 링크를 벗어난 영역에서 동일한 주소를 갖는 것은 문제가 되지 않는다.

* Switch vs Router

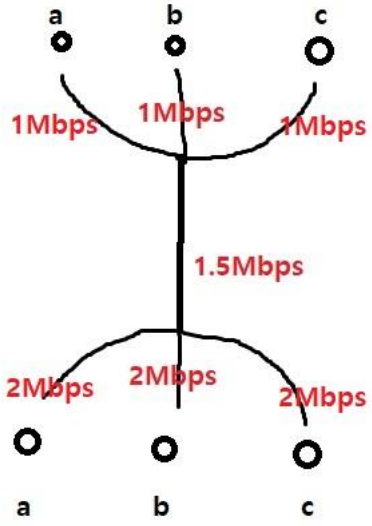
- Switch : 링크 계층, 이더넷에서만 사용, LAN 내부만 관여, 링크 계층을 하나의 네트워크로 보이게 만들어 줌, switch에는 ID가 존재하지 않는다(경로에 switch 주소 포함 X), IP주소를 이더넷 주소로 매핑 시킨 테이블 존재. (하나의 단말(하드웨어)이 IP주소로 다른 하드웨어에 접근하고자 할 때, ASP(Address Resolution Protocol = 주소결정 프로토콜)을 통해 목적지 하드웨어 주소를 FF:FF:FF:FF:FF:FF(네트워크 계층에서 255.255.255.255의 주소와 동일한 역할, broadcast역할)로 설정한 뒤, 해당 IP를 가진 하드웨어가 수신하여 자신의 하드웨어 주소를 반환, 이를 이더넷 내의 테이블에 저장해 놓는다.)

- Router : 네트워크 계층, IP 계층과 그 이상의 계층을 고려, 인터넷 전체에 관여, 포트를 구별하는 특징이 있다.(NAT 라우터의 경우, 글로벌 IP 연결단자와 로컬 IP 연결단자가 구별된다.), ID 존재(경로에 라우터 주소 포함)

10. 네트워크의 품질의 기준 (성능 지표)

- 속도

1) 대역폭(Band Width) : 주파수 넓이 – 전송률(시간 당 얼마나 많은 데이터 = Mb/s가 전송되는 지)에 비례한다. 장비의 성능과 시스템에 접근한 사람의 수가 대역폭에 영향을 줄 수 있다. End to end 서비스에서 최소의 대역폭이 나에게 주어지는 대역폭이라 할 수 있다.



Ex) 각 엣지에서 계속해서 데이터를 전송하고 있다고 가정했을 때, 중간 네트워크에서 각 엣지의 평균 대역폭은 0.5Mbps(1.5를 3으로 나눈 값)이다. 따라서 중간 네트워크를 제외한 각각의 네트워크 대역폭이 각 1Mbps, 2Mbps라 하더라도 각 엣지의 대역폭은 0.5Mbps이다.

2) 전송률 : 일정 시간 당 전송되는 데이터의 양, 대역폭에 비례하기 때문에 대역폭의 의미가 전송률이 되기도 한다.

- 신뢰도 :

1) 일정한 품질로 서비스가 제공되는 지. -> 신뢰도를 높이기 위해서는 최악의 상황에 대비할 수 있는 자원을 미리 할당해 놓아야 한다.

2) Coverage : 서비스를 제공할 수 있는 영역 범위가 넓어야 함.

3) 보안

4) Packet loss : 패킷의 누락의 원인 -> 대역폭을 줄이는 원인.

- 잡음
- 혼잡

- 지연 시간 : 패킷을 보내기 시작한 시점에서 패킷을 받기 시작한 시점까지 걸리는 전송 지연시간.

1) 지연 시간이 품질의 기준이 되는 항목 : 전화, 실시간 방송, 교통, IOT, 금융 시장(증권)

2) End to end까지 지연되는 시간들을 누적시킨 값.

3) 종류

- Processing 지연 : 컴퓨터 내에서 패킷을 처리하는 시간.
- Queuing 지연 : 동시에 동일한 경로로 패킷이 전송되는 경우, 패킷이 순차적으로 나열되는 시간. 병목 현상을 파악한 뒤, 자원을 조금만 늘리면 해결됨. 신뢰도의 문제와 연결된다.(패킷이 나열되다가 수용 범위를 넘어서면 사라지는 문제가 발생할 수 있기 때문.) -> 예측이 어렵다.
- Transmission 지연 : 미디어에 패킷이 맨 앞부분에서 끝까지 통과하는 시간. 전송률이 높아지면서 영향이 미미해졌다.
- Propagation 지연 : 물리적인 거리에 의한 지연. 전기 신호가 전달되는 데 걸리는 시간. -> 예측이 가능하다.

예) 끝 단 간 지연시간이 100ms이고 전송률이 100MB/s인 네트워크에서 1MB를 전송하여 수신 완료하는 데 걸리는 시간은?

=> 100ms + 10ms(1/100s) = 110ms