03/14 - 0. Overview

어플리케이션
웹 브라우저, 웹 서버 등

라우터
메세지를 전달받아 목적지로 전달하는 장치

링크
라우터 사이의 링크는 bandwidth가 큰 케이블이 될 것이고, 어떤 링크는 모뎀 선이나 이더넷 케이블이 될 수도 있다. 물리적인 링크가 아니라 WI-FI, LTE, 5G같은 무선 링크일 수도 있다.

대역폭bandwidth
데이터의 최대 속도. 보통 초당 얼마나 큰 데이터가 전송될 수 있는지로 표현한다. 1kbps는 1초에 1kb를 전송할 수 있다는 뜻

프로토콜 Protocol
통신규약; 소통방식 또는 절차. 원활한 통신을 하기 위해 만들어진 규약. 인터넷에서의 모든 소통은 프로토콜에 의해 조정된다.

패킷
메시지를 담는 봉투와 비슷한 개념. 비트로 쪼개져 있지만 기본적으로 완전한 하나의 단위라고 보아야 한다.

Q. 임시 저장 공간이 넘치면?
그냥 패킷을 버린다. 그래서 트래픽이 몰리면 패킷 유실이 일어난다. 통신을 할 때 보통 라우터를 여러 개 거치기 때문에... 중간의 라우터 중 하나가 넘치면 패킷이 유실돼 에러가 난다.

Q. 그럼 TCP는 왜 reliable한 것인가?
그래서 서버가 패킷이 유실되었다는 것을 감지하면 패킷을 재전송한다. (라우터는 전달하는 역할만을 빠르게 수행한다, dumb core)

인터넷의 구성 요소
크게 보았을 때 네트워크의 가장자리network edge에는 어플리케이션과 호스트
네트워크의 가운데network core에는 라우터, 네트워크의 네트워크가 존재하고
이들을 이어주는 링크가 존재한다.

Network Edge
네트워크 가장자리에 있는 클라이언트와 서버가 의사소통을 할 때 인터넷 통신 서비스를 사용해 데이터를 주고받는다. 이 통신 서비스에는 두 가지 종류가 있다.

1. Connection Oriented Service
   TCP(Transmission Control Protocol)
   다음 세 가지 특징이 있다.
   reliable(유실되지 않음), in-order byte-stream data transfer(순서 보장)
   flow control: sender won't overwhelm receiver(리시버의 사양, 속도에 맞춰 전달)
   congestion control: senders "slow down sending rate" when network congested
   (네트워크 상황이 좋지 않으면 느려진다)
   ex. HTTP, FTP, SMTP(email)
   reliable해야 하면(대부분의 경우) TCP를 사용한다.
2. Connectionless service
   UDP(User Datagram Protocol)
   connectionless, unreliable, no flow ctrl, no congestion ctrl
   (받아들이는 리시버와 관계없이 보내는 사양 그대로 보낸다. 빠르지만 유실되지 않는다는 보장이 없고, 따라서 순서도 보장되지 않는다.)
   ex. 미디어 스트리밍 서비스, 화상 통화, DNS
   실시간 영상통화 등은 패킷이 몇 개 유실되어도 유저 입장에서 별로 큰 차이가 없음

Network Core
네트워크의 중심에는 라우터가 얽혀 있다. 라우터는 데이터를 다음 두 가지 방식으로 전달한다.

1. Circuit Switching
   출발지에서 목적지까지 가는 길을 미리 예약하고, 특정 사용자만 사용할 수 있도록 만든 것
   링크의 대역폭을 특정 유저를 위해 일부 할당한다.
2. Packet Switching (인터넷은 이 방식을 채택)
   사용자가 보내는 패킷을 받아 그때그때 올바른 방향으로 forward하는 것
   패킷을 Queue에 넣고, 들어온 순서대로 적당한 목적지로 보내준다.

Circuit Switching vs. Packet Switching
대역폭bandwidth이 1 Mbps인 케이블에 연결된 라우터가 있다.
100kbps의 대역폭을 가진 N명의 유저가 통신하려고 할 때,
- Circuit Switching을 사용했을 때 한 번에 최대 10명의 유저까지 통신 가능
- Packet Switching을 사용했을 때에는 제약이 없음. 일반적인 인터넷 사용 패턴에서는 통신하는 시간보다 통신을 하지 않는 시간이 더 길기 때문에, 동시에 10명까지 통신할 수 있겠지만 실질적으로는 제약이 없다고 볼 수 있는 것. 이런 점 때문에 인터넷은 Packet Switching을 택했다.

다만 Packet Switching방식의 단점도 있는데, 다음 네 개의 delay가 그것이다.

(1) Processing Delay: 패킷을 검사해 비트 오류나 패킷 목적지를 검사해야 해서(nodel processing) processing delay가 존재한다. 좋은 라우터를 쓰면 이 속도가 빨라진다.

(2) Queueing Delay: 유저 수가 많은 경우에는 router에서 빠져나가는 속도보다 들어오는 속도가 더 빠르기 때문에 넘치는 패킷이 유실되지 않도록 임시 저장 공간buffer/queue을 마련해야 한다. 이 임시 저장 공간에서 패킷이 기다리는 시간을 queueing delay라고 한다. 트래픽이 몰리면 커지는 딜레이라서 사실상 컨트롤이 불가능하다.

(3) Transmission Delay: 패킷이 라우터에서 링크로 나갈 때, 패킷의 첫 번째 비트부터 마지막 비트까지 나가는 시간. 즉 (패킷 크기)/bandwidth가 Transmission Delay가 된다. bandwidth를 늘리면(좋은 회선을 쓰면) 이 딜레이가 줄어든다.

(4) Propagation Delay: 패킷의 마지막 비트가 링크로 나간 순간부터, 목적지에 해당 패킷이 완전히 전달되기까지의 시간. 전자기파의 속도이기 때문에 이 딜레이는 빛의 속도에 따른 딜레이다.

03/14 - 1. Application Layer

네트워크 계층 (TOP-DOWN 순)
Application (HTTP)
Transport (TCP, UDP)
Network (IP)
Link (WI-FI, LTE, 3G, ETHERNET...)
Physical
하위 계층에서 상위 계층에 서비스를 제공한다.
즉 HTTP는 TCP/UDP의 기능을 사용한다.

프로세스
실행 중인 프로그램. (더 작은 실행 단위는 스레드)

Application Layer
Application을 실행하면 프로세스가 되고, 결국 Application Layer에서의 통신은 프로세스간의 통신이다. 웹 브라우저가 가장 흔한 프로세스라고 할 수 있다. Application Layer, Transport Layer은 Network Edge, 클라이언트와 서버에만 존재하는 것이다. (Router에는 Network, Link Layer만 존재한다)

Client-Server architecture
클라이언트는 브라우저, 서버는 웹 서버를 말한다.
서버는 항상 작동하여야 하며, 고정된 IP 주소를 가지고 있어야 한다. 클라이언트는 서버와 통신하며, 동적인 IP 주소를 가지고 있어도 된다. 클라이언트끼리 직접 통신하지 않는다.
두 프로세스간에는 Socket을 만들어 통신하며, 한쪽에서 write하면 다른 쪽에서 read한다. 이때 두 프로세스를 연결시켜야 하고, 그러려면 서로의 소켓의 주소를 알아야 한다. 이때 그 주소가 IP + Port이다. IP가 컴퓨터의 주소라면, Port는 특정 프로세스(에 물린 소켓)를 지칭하는 것이다. 예를 들어 https://www.naver.com/은 www.naver.com(→ DNS → IP주소), https(443번 포트)로 구성되어 있다.
Application Layer이 통신에서 필요한 것을 말해 보자면 다음과 같은 것들이 있을 수 있다.
  Data Integrity: e.g. 100% reliable data transfer
  Timing: low delay
  Throughput(처리량): 최소 1Gbps의 대역폭이 나왔으면 좋겠음
  Security: encryption, data integrity, ...
그런데 Transport에서 제공하는 것은 Data Integrity밖에 없다(TCP). 현재는 다른 것들을 해주지 않으니 Application Layer에서 보안 이슈같은 것들을 해결하는 상황.

HTTP
HyperText Transfer Protocol: 하이퍼텍스트(텍스트를 참조하는 텍스트)를 전송하기 위한 프로토콜
웹의 application layer protocol이며, client-server model을 가진다. client는 서버에 request하며, 서버는 클라이언트에 response한다. Transport layer의 TCP를 기반으로 하며, request-response 이전에 TCP Connection을 생성해야 한다.
"stateless" HTTP는 통신을 하면서 맥락이나 추가적인 정보를 전혀 기억하지 않는다.
HTTP는 TCP를 사용하는 방식에 따라 다음 두 가지로 나뉜다.
  non-persistant HTTP: request-response 이후에 연결 종료. 웹 페이지를 로드하는 (고전적인) 과정
    TCP Connection 요청 - TCP Connection 응답
    open TCP connection
      HTTP Request - HTTP Response(index.html)
    close TCP connection (server)
    close TCP connection (client)
    이후 브라우저는 index.html을 parsing하면서, ref가 걸려 있는 자료에 대해 모두 위 과정을 반복
  persistant HTTP: request-response 이후에 연결 유지, 계속해서 데이터를 주고받음. 실제로는 이 방식으로 브라우저가 통신함.
  기본적으로는 최초 통신 이후 브라우저가 몇 개의 파일을 받을지를 parsing해서, 그만큼 req와 res를 보내고 그때까지 TCP connection을 끊지 않는다. 다만 실제로는 req를 여러 번 보내고 res를 한 번에 받는 pipeline방식을 사용한다.