컴네

ARPAnet(아파넷)

- 미 국방부 산하 ARPA에서 제작한 최초의 패킷 스위칭(Packet Switching)네트워크.
  - 패킷이란 송수신하는 정보를 잘게 쪼개어 헤더(Header)를 붙인 정보의 패키지다.
  - 레너드 클라인록(Leonard Kleinlock) 교수가 아르파넷 개발에 큰 역할을 했다.
    - 레너드 교수는 패킷 교환망의 이론적 배경과 큐잉 이론으로 잘 알려져있다.
  - 훗날 인터넷의 기반이 된다.
- 아파넷에서 통신을 위해 사용한 프로토콜이 AHHP와 ICP였으며, 이를 제어하는 프로그램이 NCP였다.
  - 이는 훗날 TCP/IP의 기반이 된다.

TCP/IP 프로토콜

- 인터넷 네트워크 내부에서 컴퓨터들 사이의 정보교환에 쓰이는 프로토콜 중 하나이다.
  - TCP는 **전송 제어 프로토콜(Transmission Control Protocol)** 을 의미한다.
  - IP는 **인터넷 프로토콜(Internet Protocol)** 을 의미한다.
  - 수 많은 프로토콜의 모음을 인터넷 혹은 TCP/IP 프로토콜 슈트(Protocol Suite)라고 칭한다.
- 인터넷의 아버지라 불리는 로버트 칸(Robert Kahn)과 빈트 서프(Vinton cerf)가 TCP/IP를 발명했다.

컴퓨터 네트워크란?

- 디바이스(호스트, 종단 시스템)들이 통신망을 통해 서로 연결된 체계를 말한다.

> 컴퓨터 네트워크 = (에지와 코어 = 호스트와 라우터) 그리고 프로토콜

- James Kurose가 제창한 Nuts and bolts 관점으로 인터넷을 살펴보자.

> 네트워크 엣지: Hosts, Access network, Physical media

    - 네트워크에선 모든 디바이스(컴퓨터, 서버)들을 호스트 혹은 종단 시스템이라고 칭한다.
        - 클라이언트와 서버 정도로 구분한다.
    - 이 디바이스들을 1차적으로 엣지 라우터에 연결시키는 액세스 네트워크(Access network)와 물리적 매체(Physical media)들이 있다.
        - 엣지 라우터는 호스트라고 볼 순 있으나, 종단 시스템이나 종단 호스트는 아니다.
        - 각 매체에 따라 전송 속도(Transmission rate)는 달라진다.
        - 이때 전송률은 대역폭을 따른다.
    - 디바이스들과 액세스 네트워크를 포함하여 네트워크 에지라고 칭한다.

> 네트워크 코어: Packet/circuit switcing, Internet structure

    - 네트워크의 코어에는 서로 연결된 라우터와 스위치들이 존재하며, 이는 ISP(인터넷 서비스 제공자)들이 관리하게 된다.
    - 이 라우터들이 그물처럼 엮여있고, 네트워크 에지들을 연결하게 된다.
        - 즉 네트워크의 네트워크(Network of Networks)이다.
- 네트워크의 모든 상호작용에는 통신규약인 프로토콜(Protocol)이 관여하게 된다.
    - 따라서 컴퓨터 네트워크를 요약하자면, 에지(호스트와 액세스 네트워크)와 이를 연결하는 코어(라우터) 그리고 그 사이의 통신규약인 프로토콜을 합친 것이라고 볼 수 있다.
- 후에 더 자세하게 다루게 된다.

OSI 7 Layers(OSI 7 계층)

- OSI 7 계층은 네트워크에서 통신이 일어나는 과정을 7 단계로 나눈 것.

> OSI(Open Systems Interconnection)

    - 왜 나눴느냐? = 시인성이 좋아지고, 이해하기 쉬우며, 특정 계층에서 문제가 발생했을 때 다른 단계를 건들이지 않고도 이상이 생긴 단계만 고칠 수 있음.

1. Physical layer
    - 물리 계층, 단순히 비트 단위의 데이터를 전기적 신호로 전달한다.

    - 이 계층에 속한 장비는 케이블, 리피터, 허브 등이 있다.
    - 데이터에 오류가 있는지, 어떤 데이터인지는 관심이 없다.
2. DataLink layer
    - 데이터 링크 계층, 프레임 단위 정보의 안전한 전달을 수행하는 계층이다.
        - 이 계층에 속한 장비는 네트워크 브릿지나 스위치 등이 있으며, 두 지점간 전송을 담당한다.
            - 이더넷, 알로하, ATM 등이 이 계층에 속한다.
        - 통신에 오류가 있는지 검사하고, 재전송도 하게 된다.
        - MAC 프로토콜 주소를 사용하여 통신하게 된다.
3. Network layer
    - 네트워크 계층, 데이터를 목적지까지 가장 안전하고 빠르게 송신하는 라우팅을 담당한다.
        - 대표적인 장비가 바로 라우터이다.
        - 논리적인 주소(IP)를 가지고 여러 길이의 데이터를 네트워크에 전달하고, 전송 계층이 요구하는 QoS을 제공한다.
    - TCP/IP 상에서 IP 계층이 바로 3계층을 의미한다.
        - 패킷을 목적지까지 전달하고, 그에 수반되는 일을 하는 역할.
        - 즉, 3계층 아래에 어떤 하드웨어가 있고, 어떤 특성을 가졌는지는 관계 없이 독립적인 역할을 수행한다.
4. Transport layer
    - 전송 계층, 통신을 활성화하기 위한 계층이며 주로 TCP나 UDP 프로토콜을 이용한다.
        - 1 ~ 3 계층을 거친 데이터를 하나로 합쳐 상위 계층에 올려주는 역할을 한다.
        - 패킷의 전송이 유효한지 확인, 실패한 패킷은 재전송한다.
            - 즉 데이터를 전송했을 때, 도착을 보증하는 역할을 한다.
        - 패킷들의 헤더를 뜯어내어 Assembly 역할을 수행한다.
    - TCP 프로토콜은 OSI 관점의 4 계층에 해당하며 신뢰성있는 전송을 보장한다.
    - UDP 프로토콜은 TCP와 대비되게 신뢰성이 낮은 프로토콜이다.
5. Session layer

- 세션 계층, 데이터 통신을 위한 논리적인 연결. 실제 연결은 4 계층에서 맺고 끊으므로 어플리케이션의 관점에서 해석한다.
    - TCP/IP 세션을 만들고 없애는 작업을 한다.
    - 양 끝단의 프로세스가 통신을 관리하는 방법을 제공한다.
        - 동시 송수신(Duplex), 반이중(half-duplex), 전이중(full duplex) 방식이 있다.
    - 이런 작업은 주로 운영체제가 담당한다.

6. Presentation layer
    - 표현 계층, 데이터 표현이 다른 프로세스 사이의 독립성을 제공, 암호화 한다.
        - 문서의 인코딩, 확장자의 구분 등이 이 계층의 역할이다.
        - 사용자의 명령어를 완성, 결과를 표현한다.
7. Application layer
    - 응용 계층, OSI의 최종 목적지로 HTTP, FTP, SMTP, POP3, IMAP 등의 수 많은 프로토콜이 있다.
        - 모든 통신의 양 끝단은 프로토콜이다
    - 응용 계층은 응용 프로세스와 함께 서비스를 제공한다.
    - 과거에는 smtp 프로토콜(email)을 주로 사용했으나, 현재에는 http 1.1 프로토콜을 사용한다.

TCP/IP

- TCP는 4 계층인 전송 계층에 위치한 프로토콜이며, IP는 3 계층인 네트워크 계층에 위치하고 있다.
    - 따라서 TCP와 IP는 자신의 아래에 어떤 계층이 오든 간에 사용할 수 있다.
    - 우리가 익히 알고있는 통신 기술(ex.이더넷, 와이파이, 4g, lte...)들은 1 ~ 2 계층에 위치하고 있다.
- OSI나 TCP/IP 모두 표준으로 정해져 있다.
    - ISO(국제 표준화 기구)에서 OSI의 표준을 지정했다.
    - IETF(Internet Engineering Task Force)에서 RFC 표준을 지정한다. (주로 3계층 이상을 지정한다.)
        - UDP = RFC768

        - IPv4 = RFC791
        - TCP = RFC793
        - IPv6 = RFC2460
        - RFC는 변하지 않는 표준이며, 누구나 제출할 수 있는 Draft에서 토론을 거쳐 지정된다.
            - 드래프트는 유효 기간을 가지고 있다.
    - 1 ~ 2 계층은 ITU-R이나 IEEE등에서 표준을 지정한다.
        - WIFI도 IEEE 802.11 표준을 따라 만들어진 기술이다.(1 ~ 2 계층)

이더넷(Ethernet)

- 제록스에서 개발한 IEEE 802.3으로 표준화 된 1 ~ 2 계층에서의 통신 기술이다.
    - MAC 주소를 사용하여 통신한다.
    - 무작위 액세스 프로토콜인 CSMA/CD를 통해 이더넷에 연결된 여러 컴퓨터들이 하나의 전송 매체를 공유할 수 있게 한다.

WLAN(Wireless Local Area Network)

- IEEE 802.11으로 표준된 유선 이더넷과 호환되는 무선 로컬 네트워크.
    - WIFI는 WLAN의 브랜드 명이다.

액세스 프로토콜

- 멀티플렉싱(MUX)를 네트워크에서는 다음과 같은 의미로 사용한다.

    > 2 계층에서 전파, 데이터를 수신하는 순서를 정한다.

    - OS의 스케줄링과 유사하다.
- 채널화 멀티 액세스 프로토콜
    - LAN(Local)이 아닌 WAN(Wide Area)에서 쓰인다.
    - 접근하려는 모든 채널이 분할된 여러 대역폭을 사용해서 동시에 접속할 수 있게 하는 기술.
        - CDMA = 대역폭을 code에 따라 나누겠다. (3g에서 사용)
            - 3gpp에서 표준을 제정했다.

        - FDMA = 대역폭을 Frequency에 따라 나누겠다.
        - TDMA = 대역폭을 Time에 따라 나누겠다. (2g에서 사용)
        - OFDM...
- 무작위 액세스 프로토콜
    - 이더넷과 WLAN = LAN에서 사용한다.
    - 무작위 = 확률, 수학과 연관되어있다.
        - CSMA/CD,CA...
        - ALOHA가 이를 사용했다.

ALOHA

- 아파넷이 1969년에 등장한 후, 1970년도에 등장한 무선 패킷 스위칭 통신 기술
    - WAN의 조상격이 된다.
- 하와이 주립대학에서 만들었으며, 아파넷과 연결했다.
- 대표적인 무작위 액세스 프로토콜을 사용했다.
    - FIFO 처럼 선입된 패킷이 선출된다.
    - 만약 패킷이 보내지는 중에 다른 패킷이 들어오면 충돌이 일어난다.
    - 충돌이 발생하면 패킷이 손실(loss)되므로 뒤로 빠져 랜덤한 시간동안 대기 후, 다시 송신한다.
    - 패킷이 모두 전송되어야 다음 패킷을 보낼 수 있으므로 효율이 낮다.
        - 최대 효율이 18.4% 밖에 되지 않는다.
- 현재는 RFID같은 초소형 칩에 사용한다.
    - 간단한 프로토콜이라 초소형 칩에 사용가능하다.
    - 패킷이 랜덤하게 들어오지 않고, 특정한 시간 배분(슬롯)에만 들어오게끔 알고리즘을 개선했다.
        - Slotted ALOHA = 36.8%의 효율을 자랑한다.

ATM(Asychronous Tranfer Mode)

- 패킷 스위칭을 통한 이더넷의 100배의 속도를 자랑했던 기술.
    - QoS(Quality of Service)도 상당부분 지원했다.

- Gigabit networking 책이 등장할 정도로 신기술이였다.
    - 하지만 이더넷이 빠른 시간내에 ATM의 속도를 따라잡으면서 구관이 승리를 거머쥐었다.

QoS(Quality of Service)

- 인터넷에서의 QoS는 일반적인 의미와 약간 다르다.

> 대역폭(Bandwidth), 손실(Loss), 순서(Order), 타이밍(Timing) 등을 보장하는 지에 대한 개념이다.

    - 인터넷은 best effort 모델을 가진다.
        - 최선을 다하지만, 그 어떤 항목도 보장하지 않는다.
    - ATM은 여러 서비스 모델이 있으며, 각 서비스는 다양한 QoS를 보장한다.

네트워크의 엣지

- 호스트를 포함한 액세스 네트워크(RAN)
    - 이들을 잇는 링크는 두 가지로 나눈다.
        - Guided media = 물리적 매체
            - TP(Twisted Pair) = 랜선이나 광케이블 등
        - Unguided media = 물리적 선이 없는 무선 통신
            - Wireless Radio

네트워크의 코어

- 3 계층에 위치한 라우터들이 그물처럼 엮인 구조
    - 라우터 간 통신은 패킷 스위칭을 이용한다.
        - 과거에는 서킷 스위칭이 있었으나, 패킷에 밀려 사라지게 되었다.
        - 가상 서킷(Virtual Circuit)을 이용하는 ATM 같은 기술들도 등장했으나 사라졌다.
            - 서킷은 출발지와 목적지 사이의 콜에 종단간 자원을 할당한다.
- 네트워크의 코어에서 가장 중요한 두 개념은 **포워딩** 과 **라우팅** 이다.

> 포워딩이란 라우터의 Local한 기능이며, 패킷이 목적지로 가기 위한 경로를 설정하는 것이다.

- 입력된 링크를 적절한 출력 링크에 넣어 주는 것이며, 단순히 받은 패킷을 전달하는 것으로 끝이다.

> 라우팅이란 라우터들 사이의 Global한 기능이며, Source와 Destination 사이에서 최적의 Path를 찾는 것이다.

- 출발지와 목적지 사이의 수 많은 경로 중에서 가장 빠른 경로를 찾아내는 것이다.
  - 이는 라우팅 알고리즘(ex.다익스트라 알고리즘)등의 최적 경로 알고리즘을 통해 계산한다.

지연(Delay)

- 패킷은 라우터를 통해 포워딩(Forwarding)되며 나아간다.
- 이 과정에서 처리 지연(Processing delay), 전송 지연(Transmission delay), 전파 지연(Propagation delay), 큐잉 지연(Queuing delay) 등의 지연이 발생한다.

> 처리 지연이란 패킷의 헤더를 읽고, 패킷을 어느 출력 링크로 보낼지 결정하는데 걸리는 지연이다.

- 특정 놀이기구를 탑승하기 위해 어디로 갈지 결정하는 시간이다.

> 전송 지연이란 패킷이 전송을 시작하기 전, 패킷을 링크에 올리는데 걸리는 지연이다.

- 놀이기구를 탑승하는데 걸리는 시간이라고 이해하면 된다.
- L 비트를 R bps의 전송률로 보내면 L/R로 지연을 표기할 수 있다.

> 전파 지연이란 링크의 처음 지점에서 목적 라우터까지 전파할 때 발생하는 지연이다.

- 놀이기구가 출발해서 한 코스를 도는데 걸리는 시간이다.
- 출발지와 목적지의 거리가 d고, 링크의 전파 속도(광속에 가깝다)가 s면 전파 지연은 d/s로 표기한다.

        - 패킷의 전송률과는 상관 없다.
    - 이는 거의 광속에 근접하므로 우리는 무시한다.

> 큐잉 지연이란 라우터에 저장 및 진행을 거칠 때 발생하는 지연이다.

    - 놀이기구 탑승을 기다리는 줄이라고 이해하면 된다.
    - 패킷은 라우터의 버퍼(큐)에 저장되어있다가 링크를 타고 전달된다.
        - 모든 패킷이 라우터에 도착 해야 다음 링크로 보내질 수 있다.
    - 큐잉 딜레이는 큐에 앞서 도착한 패킷의 수에 따라 0에서 무한대의 지연을 가질 수 있다.
        - 라우터에 패킷이 입력, 출력되는데 지연이 생기며,
        - 패킷이 큐에 꽉 차 있으면 다른 패킷은 손실된다.
- 큐잉 지연은 없다고 가정하고, 출발지와 목적지 호스트 사이에 N - 1개의 라우터가 있다고 가정하면

> 종단간 지연 = $ N (L/R)$

Network Core

- 네트워크의 코어는 상호 연결된 라우터로 구성되어 있다.
    - 이 라우터들은 패킷 스위칭을 통해 데이터를 주고 받는다.
    - 모든 패킷은 링크를 통해서 Source에서 Destination까지 이동한다. (Forwarding)
    - 링크가 꽉 찼을 때 패킷이 전송된다. (Store and Forward)
    - 이때 패킷의 전송시간(Packet Transmission Delay)는 L / R (패킷의 길이 / 시간당 전송 가능한 비트 수)로 계산할 수 있다.
    - 패킷이 도착하는 속도가 전송하는 속도보다 빠른 경우
        - 패킷이 Queue에 쌓여 전송되기를 기다린다.
        - 만약 버퍼가 꽉 차면, 일부 패킷이 loss될 가능성도 있다.
- 네트워크의 코어는 크게 두 개의 Function을 가진다.
    - Forwarding

        - local한 작업이며, 라우터의 입력 링크에서 적절한 출력 링크로 보내기만 하는 작업이다.
    - Routing
        - Global한 작업이며, Source-Destination 간에 최적의 경로를 찾아 패킷을 전송한다.
        - 이때 다익스트라나 벨만-포드 알고리즘 등 라우팅 알고리즘을 사용한다.

Circuit Switching

- 서킷 스위칭은 일종의 연탄이다.
    - 안쓰는 것은 아니지만, 대세에는 밀린 기술이다.
- 회선 하나를 전용으로 사용한다.
    - 종단간 자원을 할당하고, 출발지와 도착지가 서로 Call을 약속하고 대기한다.
- 자원을 공유하지 않는다.
    - 회로와 비슷하며, 일정한 성능을 보장한다.
    - circuit segment가 사용되지 않을때는 가동하지 않는다.(공유 X)
- 이 단일 회선(정해진 대역폭)에서 여러 신호를 보내기 위해 다중화(Multiflexing)을 사용한다.
    - FDM(Frequency Division Mux)

      > FDM은 대역폭을 여러 주파수로 나누고, 각 주파수에 대응하는 신호를 보내게 만드는 기술이다.

    - TDM(Time Division Mux)

      > TDM은 대역폭을 시간에 따라 분할하여 할당받은 Time Slot에 신호를 전송하게 하는 기술이다.
      > 2g 기술에 사용되었다.

    - Frequency hopping(FDM + TDM)

      > 특정 Time slot에 무작위 주파수에 할당된 신호를 전송한다.

    - CDMA(Code Division Mutiple Access)

> 각 사용자가 다른 코드를 통해 신호를 전송하게 만든다.
> 일부의 2g와 3g 기술에 사용되었다.

- OFDM(Orthogonal Frequency Division Mux)

> FDM 처럼 주파수를 활용하는 기술이다.

> FDM은 주파수의 중복을 방지하는 guard band가 추가적인 오버헤드를 발생시켰으나, OFDM은 이를 해결한 기술이다.

> 4g 기술에 사용하고 있다.

Circuit VS Packet

- 그렇다면 서킷 스위칭은 항상 패킷 스위칭에 밀리는가? 즉, 패킷 스위칭이 Slam dunk winner인가?
- 서킷의 장단점을 살펴보자
  - Service Quality를 보장하기 쉽다.
    - 자원을 남과 공유하지 않으므로, 내가 할당받은 양을 대역폭을 항상 사용할 수 있다.
  - Call 중에 자원이 낭비된다.
    - 전체 네트워크를 고려하면, 남는 대역폭은 공유하는게 더 낫다.
  - 데이터가 loss될 일이 없다.(이미 reserved 되어있으므로.)
  - 비효율적이나 장시간 연속적인 Traffic에 적합하다(목소리 등).
  - 큐잉 지연을 완화할 수 있다.
- 패킷의 장단을 살펴보자.
  - 한정적인 자원에서 최고의 성능을 보장하긴 힘들다.
    - Best effort라고 볼 수 있다.
  - 데이터가 짧은 시간동안 집중적으로 발생되는 Busty traffic에 적합하다.
  - 필요할 때만, 필요에 의해 네트워크를 사용하므로 효율적이다.
  - 패킷이 손실될 가능성이 있다.
- 예시를 들어 1기가 대역폭의 링크가 있다고 가정하자.

- 사용중인 상태일 때 100메가를 사용하고, 전체 시간의 10%동안 사용한다고 가정하자.
  - 서킷으로는 총 10명의 사용자만 사용할 수 있다.
  - 패킷으로는 대략 35명 정도가 동시에 사용할 수 있다.
    - 어느 시점에 한 명이 사용 중일 확률은 0.1이고, 35명 중에서 10명 초과 사용자가 발생할 확률은 0.0004 미만이기 때문이다.
- 만약 패킷으로도 처리할 수 없는 트래픽 폭주가 발생할 경우, 일종의 병목 현상이 발생한다.
  - 4계층의 TCP가 Conjestion Control을 진행한다.
- 인터넷은 현재 패킷 스위칭을 통해 돌아간다.
  - 호스트는 ISP를 통해 인터넷에 연결된다.
  - ISP는 다른 ISP와 상호 연결되어, 두 호스트가 패킷을 주고받을 수 있게 한다.
  - 따라서 Network of networks 즉 인터넷은 엄청나게 복잡하게 구성되어있다.

손실과 지연

- 패킷들은 라우터의 버퍼에서 큐를 만들게 된다.
  - 만약 출력 링크의 용량보다도 패킷의 도착비율이 높다면, 큐가 꽉 차버려 나머지 패킷은 손실된다.
- 노드 간 지연은 총 4개의 지연의 합이다.
  - Process Delay
    - 패킷이 정상인지 체크하고, 어느 링크로 나갈지 결정하는데 쓴다.
    - 일반적으로 1 msec보다도 작다.
  - Queuing Delay
    - 링크에서 전송되기 전에 기다리는 시간이다.
    - 라우터의 밀집도에 따라 0 ~ 무한대의 지연이 걸린다.
  - Transmission Delay
    - 큐에서 링크에 실제로 패킷들을 얹는데 걸리는 시간이다.
    - 패킷 길이 / 링크 전송률(L / R)로 표기한다.
  - Propagation Delay
    - 다음 노드에 실제로 전파되는데 걸리는 시간이다.

        - 링크의 길이 / 전파 속도 (d / s)로 표기한다.

Protocol layers and Reference models

- 네트워크들은 굉장히 복잡하며, 많은 부분으로 구성되어있다.
- 따라서 복잡한 시스템을 다룰 때, 여러 계층으로 구분 짓는다.
    - 명시적인 구조는 복잡한 시스템을 식별하고, 관계를 허용한다.
    - 모듈화를 통한 시스템의 유지보수, 업데이트가 용이해진다.
    - 한 계층에서 구현한 서비스를 변경해도, 다른 시스템에는 영향이 없다.
- 각 계층에서는 계층에 맞는 서비스를 구현해서 사용한다.
    - 각 서비스는 자기 아래 계층의 서비스에 의존한다.

Internet protocol stack

- 총 5 계층으로 구성된 인터넷의 구조이다.
- OSI 7 계층에서 두 계층이 Application 계층에 통합된 모습이다.
    - Application, 응용 계층
        - 네트워크 응용프로그램을 지원한다.
        - IMAP, SMTP, HTTP등이 여기 속해있다.
    - Transport, 통신 계층
        - 프로세스간, end to end 간 데이터 전송을 담당한다.
        - TCP, UDP가 여기 속해있다.
    - Network, 네트워크 계층
        - Source to Destination의 데이터 라우팅을 담당한다.
        - IP, Routing Protocol이 여기 속해있다.
    - Link, 링크 계층
        - 이웃한 네트워크 원소 간 데이터 전송을 담당한다.
        - Ethernet, 802.11(WiFi), PPP등이 속해있다.
    - Physical, 물리 계층
        - 실제 회선을 통한 비트의 전송이 이뤄진다.

- 이 인터넷 프로토콜 스택을 위에서 아래로 내려오며 헤더를 붙여나간다.
  - 이를 캡슐화(Encapsulation)이라고 칭하며, 보내는 데이터는 payload라고 칭한다.
    - 아래 계층 입장에서는 위 계층의 헤더 + 데이터 모두가 payload가 된다.
  - 각 계층에서 수행한 정보가 담겨있다.
- 목적지나 라우터에서는 이 헤더를 아래에서 역으로 읽으며 데이터를 식별한다.
  - 이를 역 캡슐화라고 칭한다.
- OSI 7 reference model의 두 계층은 사라졌다.
  - Presentation, 표현 계층은 데이터를 해석하고, 그 의미를 표현하는 역할을 맡고 있었다.
  - Session, 세션 계층은 데이터를 동기화, 저장, 복구하는 역할을 맡고 있었다.
  - 이 계층들은 필요해질 경우 Application 계층에서 구현해야한다.

인터넷 역사

- 1961, Kleinrock이 패킷 스위칭의 큐잉 이론 제시
- 1969, ARPAnet 등장
- 1970, ALOHA
- 1974, Cerf and Kahn
  - Architecture for interconnecting networks에서 TCP/IP에 대해 씀.
  - 4가지 원칙.
    1. Minimalism, autonomy.
       - 무언가 변화하여 부작용이 발생하는 것을 막기 위해 미니멀한 구조로 만들자.
       - 네트워크들을 서로 연결하는데에 있어 내부적인 변화는 필요 없다.
    1. Best-effort service model
       - QoS와는 반대로 최선의 노력을 다하나, 어느것도 보장하지는 않는다.
    1. Stateless routing
       - 패킷 스위칭에선 패킷이 여러 경로를 지날 수 있으므로 라우팅이 필수적이다.
       - 이 라우팅을 상태가 없는 프로토콜이 담당해야한다(IP)

        - IP가 라우팅을 하면 패킷이 어디서 왔는지 알 필요가 없고, 이를 저장할 필요도 없다.
        - TCP는 반대로 상태를 가지는 프로토콜이다.
    1. Decentralized control
        - 중앙 집중이 아닌, 흩어져서 통제권을 가져라.
- 1974, Donald Knuth가 the Art of Computer Programming으로 튜링상 수상.
- 1976, 제록스 PARC에서 이더넷 개발
- 1970년대 후반, ATM 등장.
- 1979, 아파넷의 서버가 200개 달성.
- 1983, Dennis Ritchie가 UNIX와 C 언어를 개발해 튜링상을 수상.
- 1990, 아파넷이 해체되고 인터넷, Web이 등장했다.
- 1994, Mosaic과 Netscape의 등장
- 2000년대 초, 모자이크와 넷스케이프가 망하고 어플리케이션이 대거 등장했다.
- 2023년 올해에는 Avi Wigderson 교수가 튜링상 수상.

Application layer

- OSI에서는 7계층, IP에서는 5계층에 Application layer가 존재한다.
    - Presentation layer, Session layer가 합쳐져있다.
- 이번 장에서는 Principle of network applications를 중점적으로 살펴보게 된다.

Principle of network applications

- 네트워크 app은 단순히 네트워크를 사용하는 프로그램이다.
    - 가장 오래된 email부터 웹 브라우저, sns, 게임, 유튜브, Skype 등이 모두 속한다.
- Internet Protocol Stack를 되짚어 보자.
    - Application layer는 프로그램이 실제 구현할 수 있는 프로토콜을 제공하여 프로그램들을 지원한다.
        - HTTP, SMTP, IMAP, FTP 등이 여기에 속한다.
    - Transport layer는 실제로 다른 호스트 내의 프로세스 간 통신, 종단 시스템 간의 통신을 담당한다.

        - TCP, UDP가 여기에 속한다.
    - Network layer는 데이터그램을 출발지 -> 목적지로 라우팅하고, 전송, Addressing을 한다.
        - IP, Routing protocol이 속한다.
    - Link layer는 프레임을 실제로 유, 무선을 통해 연결된 이웃 네트워크 사이에 전송한다.
        - Ethernet과 WiFi(802.11)이 여기에 속한다.
    - Physical layer는 선으로 비트를 전송한다.
        - Optical fiber, Twisted Pair 등이 이를 사용한다.
- IP를 거쳐내려오며 전송할 메시지에 헤더를 붙이는 과정을 캡슐화(Encapsulation)이라고 칭한다.
    - 각 단계의 메시지는 Payload라고 칭한다.
    - Application layer → Message
    - Transport layer → H1 + Message = segment
    - Network layer → H2 + Segment → Datagram
    - Link layer → H3 + Datagram ⇒ Frame
- 응용 프로그램은 크게 두 가지 구조를 가진다.

> 실제 통신은 네트워크 코어가 아닌, 엣지들에서 이뤄진다.
> 네트워크 코어는 프로그램이 실행되는게 아니라, 패킷을 전달하는 게 전부이다.

    - Client-Server 구조
        - 서버
            - 항상 호스트, 항상 연결되어있는 상태이다.
            - 영구적인 IP 주소를 가진다.
            - 처리할 데이터의 규모에 따라 바뀌는 데이터 센터
        - 클라이언트
            - 서버를 통해서만 간헐적으로 소통한다
            - 유동적인 IP 주소를 가진다.

            - 클라이언트끼리는 절대 연결하지 않는다.
        - Peer-to-Peer(P2P) 구조
            - 항상 켜져있을 필요가 없다.
            - 소통시 무작위로 연결된다.
            - 자가 확장성 - 피어가 늘어날 수록 성능이 좋아진다.
            - 동적인 IP를 가지고, 변경될 수 있다.
- 프로세스 통신
    - 프로세스는 호스트 내부에서 실행되는 프로그램이다.
        - Client 프로세스는 서버에 무언가를 요청하는, 통신을 시작하는 프로세스이다.
        - Server 프로세스는 요청을 기다리는, 통신을 기다리고 있는 프로세스이다.
    - 프로세스는 다른 프로세스와 통신한다.
        - 같은 호스트 내부에서는 Inter Process Communication을 통해 통신한다.
        - 서로 다른 호스트의 프로세스는 Message를 주고받으며 통신한다.
            - 이때 서로 다른 프로세스는 Socket을 통해 메시지를 주고받게 된다.
- 소켓(Socket)
    - 프로세스가 메시지를 주고받을 때 사용하는 도구이다.
        - 프로세스 하나당 하나씩 적어도 두 개의 소켓이 필요하다.
    - Transport layer와 Application layer 사이에 위치하여, 개발자가 통제하는 문이다.
        - 그 아래 계층은 OS가 통제한다.
    - 보내는 프로세스가 소켓에 데이터를 넣으면, 받는 프로세스가 소켓에 있는 데이터를 건져간다.
- 주소(Addressing)
    - 메시지를 주고받기 위해서는 프로세스가 각자의 식별자(Identifier)를 가져야한다.
    - 호스트 디바이스를 구분하는 식별자 = IP
        - IP 주소는 32비트(IPv4)로 이루어져 있다.
        - 집 주소와 비슷한 기능을 한다.
    - 데이터를 받을 프로세스를 구분하는 식별자 = 포트 번호(Port number)

        - 포트 번호는 정수이며, 오래된 기술일 수록 낮은 번호를 가진다.
        - HTTP 서버는 80, 메일 서버는 25를 가진다.
        - 집 내부의 방 번호와 비슷한 기능을 한다.
- Application layer protocol
    - 프로토콜은 여러 정보를 담고 있어야한다.
        - 주고 받을 메시지의 타입
            - request? response?
        - 메시지 문법(Syntax)
            - Header, 메시지 내부에 어떤 필드가 있는지
        - 메시지 시멘틱(Semantic)
            - 메시지의 의미, 출발지와 도착지 ip, 포트 번호 등
        - 규칙(rules)
            - 언제 어떻게 메시지를 보내고 받을지.
    - 프로토콜은 크게 두 종류가 있다.
        - Open protocol
            - TCP/IP 등 RFC 표준으로 지정된 프로토콜
        - Proprietary protocol
            - Skype 등 특정 기업이 만든 프로토콜
- Application layer가 필요로 하는(기대하는) Transport layer의 서비스(역할)
    - 서로 붙어있는 두 계층은 상호작용하기 때문에, 상부상조해야한다.
    - Data Integrity(데이터 보전)
        - 데이터를 손실 없이 전송해줬으면 좋겠다(보안과는 전혀 관련 없다).
        - Reliable data transfer = 내가 보낸 데이터가 100% 도착해야 함.
        - 그러나 이는 프로그램의 종류에 따라 변화할 수 있음.
            - 메일이나 웹 통신은 100% 보장되어야하지만, 오디오 등은 조금의 손실이 있어도 괜찮다.
    - Timing

        - Delay와 반대되는 말로, Transport layer가 적은 딜레이로 데이터를 주고받기를 원한다.
    - Throughput(처리량)
        - 사실상 최종 목적이며, 높을 수록 좋겠다.
        - less delay = more throughput
            - 일반적으로 패킷 손실 = delay가 되는 경우가 많다.
        - 하지만 이 처리량도 elastic(탄력적)일 수 있다.
            - 실시간 멀티미디어는 최소치가 정해져 있으나, 문서 전송, 메일 등의 경우에는 조금 천천히 와도 받기만 하면 된다.
    - Security(보안)
    - 이 4가지는 아주 중요하나, 모든 프로그램에서 이 4가지를 완전히 만족시키지 못하므로 적절히 타협이 필요하다.

TCP와 UDP

- 4 계층, Transport 프로토콜에는 TCP와 UDP의 두 종류가 있다.
- TCP
    - Transmission control protocol이며, 신뢰적인 전송을 추구한다.
    - 송수신자 간의 신뢰적인 전송을 보장한다.
    - 흐름 제어 = 송, 수신측의 데이터 처리 속도 차이를 해결한다.
        - 송신자의 데이터 처리 속도를 수신자보다 낮춘다.
    - 혼잡 제어 = 송신측의 데이터 전송과 네트워크의 데이터 처리 속도 차이 해결(Bottleneck)
        - 라우터의 버퍼를 늘리거나, 버퍼가 넘치지 않게 송신자의 속도를 낮춘다.
    - 데이터 보전을 제공, 다른 건 전혀 신경쓰지 않는다.
    - 서버-클라이언트 간 연결 설정 단계를 거친다.
- UDP
    - User datagram protocol이며, 단순한 전송을 추구한다.
    - 비신뢰적인 전송 방식이다.

    - 그 어떤 것도 제공해주지는 않지만, 데이터를 빠르게 전송한다.(real-time streaming = fast!)
- TCP가 대부분 UDP를 대체했으나, 아직 UDP를 쓰는 프로그램도 존재한다.
    - 멀티미디어 스트리밍, 인터넷 전화 등은 loss-tolerant하므로 UDP를 사용하기도 한다.

Socket Interface

- 1980년도 ARPA에 의해 투자, UC 버클리에서 개발되었다.
    - 목적 : TCP/IP 소프트웨어를 UNIX로 전송하자.
- 1981년도 UNIX 사이의 통신을 위한 generic interface로 소개되었다.

Socket

- 그래서 소켓이 뭐냐?

> 네트워크를 통해 서로에게 연결(plug into)될 수 있는 인터페이스.

- 많은 프로토콜(TCP, UDP)등의 일반적인 인터페이스가 된다.
    - Application layer와 Transport layer 사이에 위치한다.
    - 프로세스와 end-end-transport protocol 사이의 문의 역할을 한다.
- 프로세스간 식별을 위해 두 가지의 식별자를 가진다.
    - IP: 32비트로, 호스트의 식별자가 된다.
    - Port: 16비트로, 궁극적인 목적지를 식별한다.