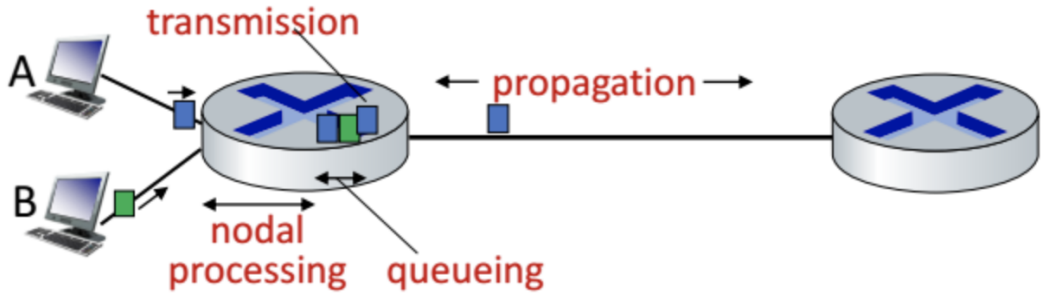
2일차

How do loss and delay occur?

- 패킷은 queue(buffer)에서 차례를 기다린다.
- queue에 있는 시간을 queueing delay 라 한다.
- queue의 용량이 초과하여 정보가 들어 온다면 정보 손실(loss) 이 일어날 수 있다.

Packet Delay 살펴보기 : 네 가지로 구분

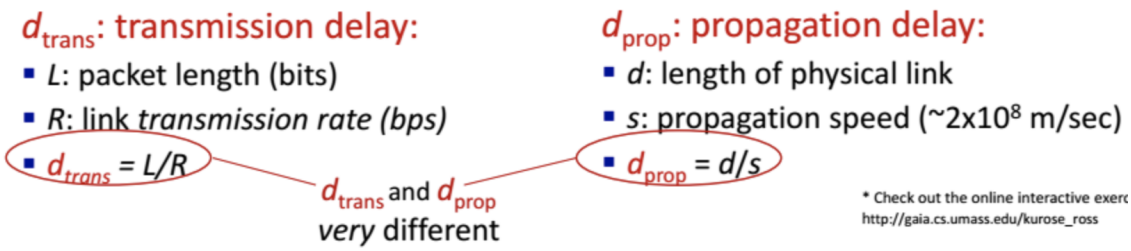


$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

nodal = processing + queue + transmission + propagation

1. processing
    a. 데이터가 에러가 났는지 bit 체크
    b. router 내부에서 처리할지 output link로 보낼지 결정
    c. 이 과정은 보통 msec 안으로 가능
2. queueing delay
    a. 보내지기 위한 순서를 기다림
    b. 일반적으로는 FIFO순으로 보내지만 다른 경우 어떤 것 부터 보낼지 select 하는 시간이 걸릴 수 있다.

3. transmission delay (물리적인 단계 router에서 physical link로 보내지기 직전까지)
4. propagation delay (물리적으로 데이터를 보내는데 걸리는 시간)



transmission delay 에서는 router에서 physical link로 보내기 직전까지 이므로 L은 패킷의 길이 / R은 링크에 transmission 되는 속도를 나타내고

propagation delay 에서는 router 와 router 사이의 physical link의 길이를 전송 속도로 나눠줘서 delay를 구할 수 있다.

자동차 톨게이트로 예를 들어 볼 수 있다.

자동차가 톨게이트를 통과하기 위해서는 돈을 지불하는 등의 시간이 소요 된다. → transmission delay

톨게이트를 통과한 자동차가 다음 톨게이트를 가기 위해서는 시간이 소요 된다. → propagation delay

Queueing Delay

- Queueing Delay 는 편차가 심하기 때문에 average packet arrival rate를 측정하여 생각해야 한다.
- La/R ~ 0 : delay 가 거의 없음
- La/R > 1 : delay 가 큼

Real Internet delays and routes

1. traceroute program : source에서 destination 까지 어떤 router를 몇초에 걸쳐 가는지 보여줌.

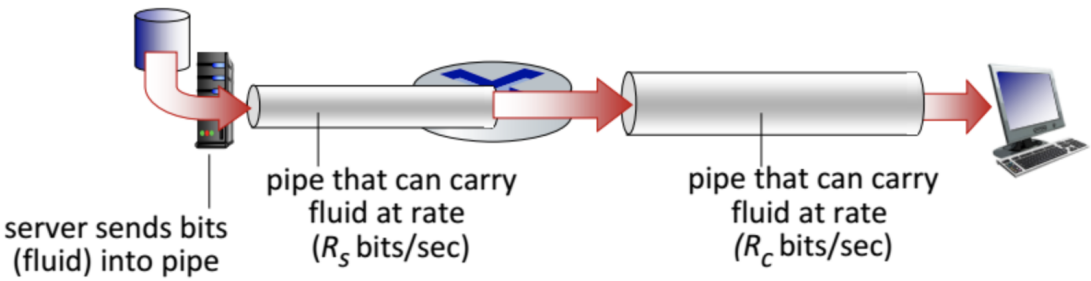
a. 명령어 예시

   i. traceroute 8.8.8.8

   ii. ping naver.com : ping 명령어는 컴퓨터 동작 여부 혹은 네트워크 상태 여부를 보여줌

Packet loss

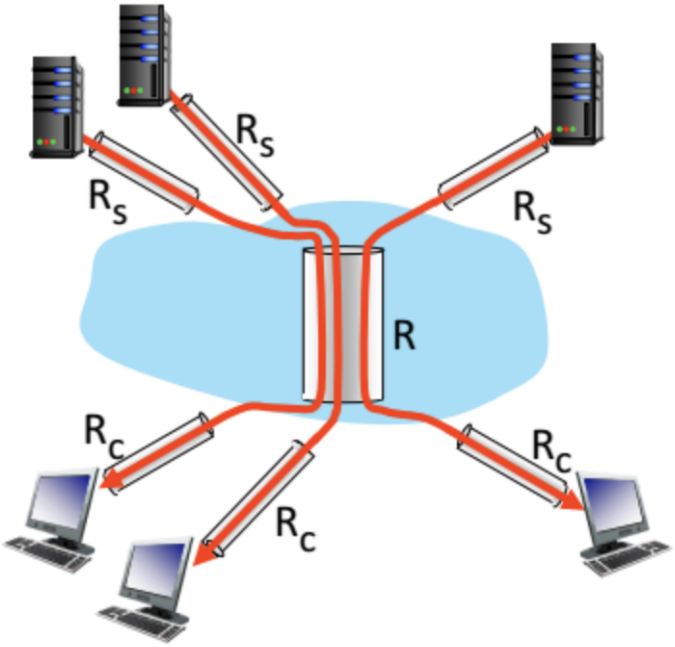
1. lost packet 은 retransmitted(재전송) 된다.
   a. by previous node
   b. by source end system
   c. not at all
2. Throughput : 송신자가 수신자에게 1초에 몇 bit 를 보내는지



   a. pipe 가 작으면 느림 (물이 수도관을 지나간다고 생각해보자)

      → 서버가 빨라도 link가 작으면 원할하지 않을 수 있음 (bottleneck)
      * bottleneck : 전체 시스템의 성능이나 용량이 하나의 구성 요소로 인해 제한을 받는 현상

   b. 이에 분산컴퓨팅을 적용하기도 함.



Network Security

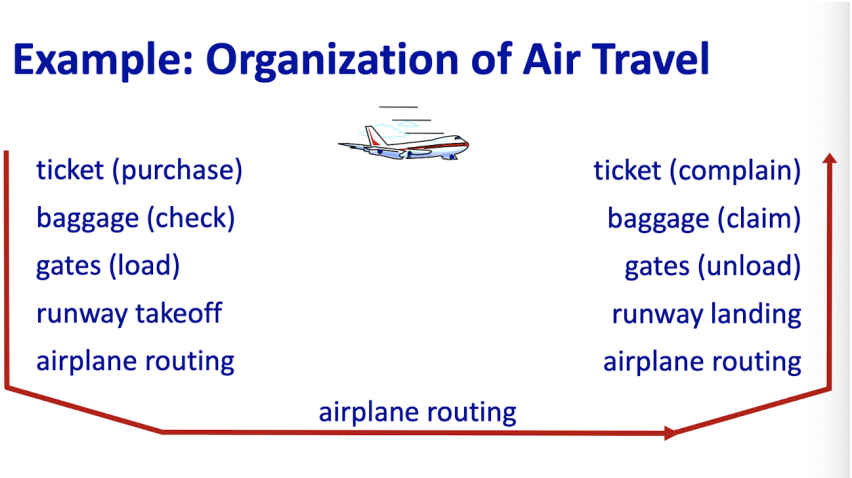
1. Bad guys : put malware into hosts via Internet
    a. virus : 스스로 복제하여 감염시킴
    b. worm : 스스로 실행하여 감염시킴
2. Bad guys : attack server, network infrastructure
    a. Denial of Service(DoS) : 다수의 좀비컴퓨터가 하나의 target 컴퓨터를 1대1로 공격

i. DDos : DoS 공격을 분산시스템을 이용하여 공격하는 기술

3. Bad guys : packets sniffing
   a. 호스트에서 서버로 packet을 보낼때 중간에 packet을 가로채서 읽음
4. Bad guys : IP spoofing
   a. 가짜 주소로 위장하여 packet을 보냄.

Protocal layers, service models

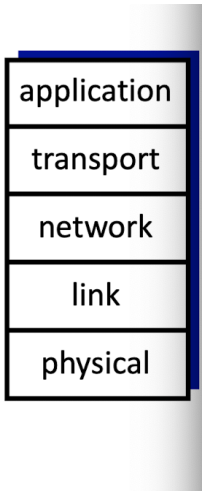
1. Why Layering?



- 여행갈때 절차를 나누는 것처럼 네트워크도 절차가 복잡해서 역할을 가진 layer들로 나누는 것이다.

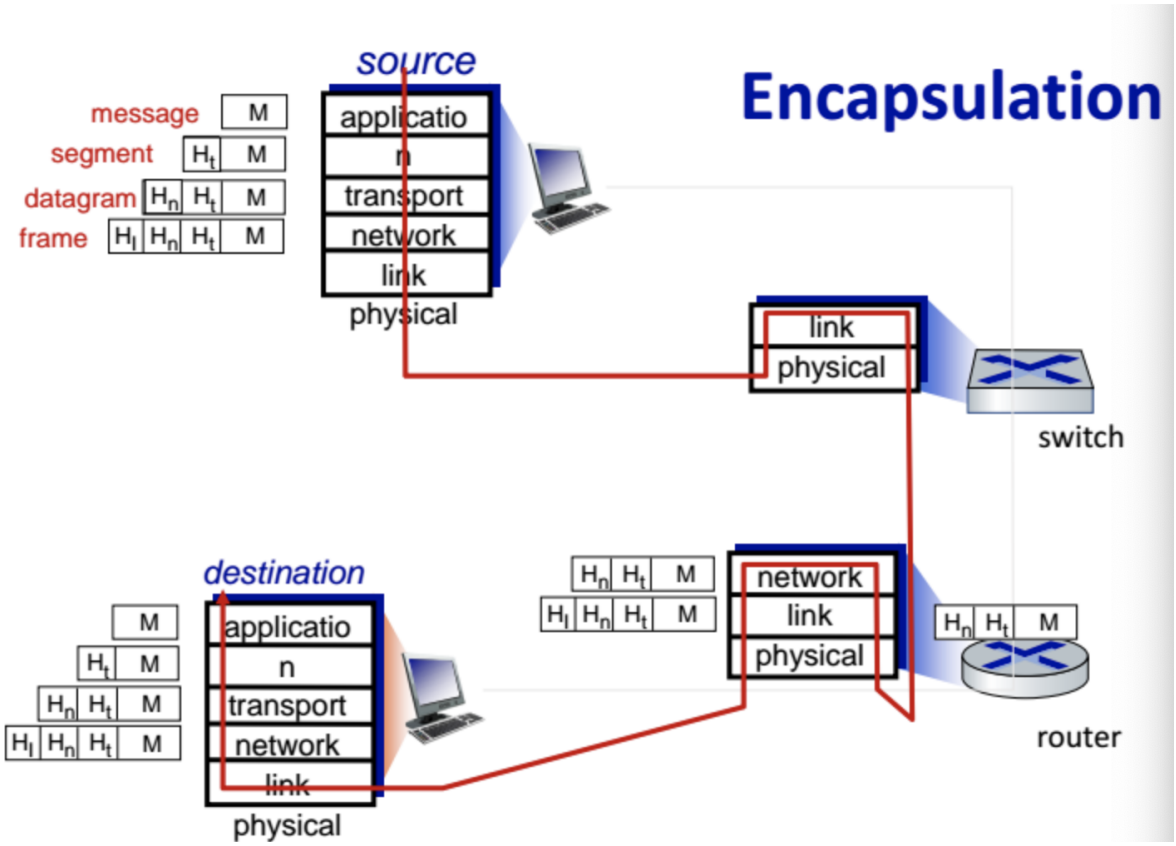
2. Internet Protocal Stack 5계층

- *application:* supporting network applications
  - IMAP, SMTP, HTTP
- *transport:* process-process data transfer
  - TCP, UDP
- *network:* routing of datagrams from source to destination
  - IP, routing protocols
- *link:* data transfer between neighboring network elements
  - Ethernet, 802.11 (WiFi), PPP
- *physical:* bits "on the wire"



1. application 계층을 application, presentation, session layer로 나눈 것이 osi 7 layer이다.

3. Encapsulation



a. application layer에서는 message(body)를 첨부하고 다음 layer로 갈때마다 해당 header를 붙여서 준다. destination 에서는 header를 읽고 처리를 한 다음 header를

날린 후 그 다음 header를 보고 처리하는 식으로 진행된다.

b. switch / router
    i. router는 경로를 선택하는 역할을 해줌
    ii. switch는 2계층 / router는 3계층

Chapter 2 : Applications

Principles of network applications

1. Some network apps
    a. social network
    b. text message
    c. email
    d. multi-user network games (MMORPG 등)
    e. streaming stored video (Youtube, Netflix)
    f. P2P file sharing(Torrent, 등)
    g. voice over IP (Skype 등)
    h. real-time video conferencing (Zoom 등)
    i. Internet serach
    j. remote login

Application architectures
(어플리케이션이 여러 node와 통신하는 방식 2가지)

1. client - server (HTTP, IMAP. FTP 등)
    a. server
        i. always-on host

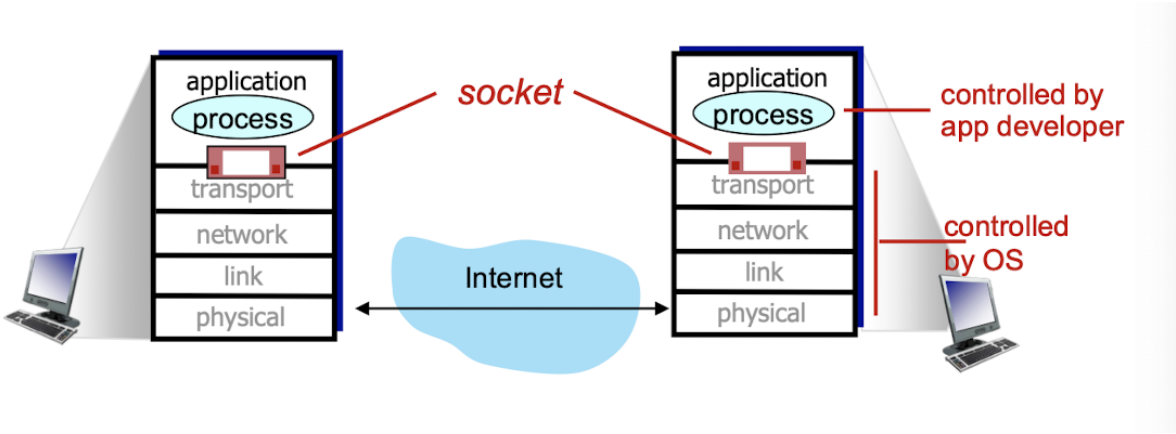
        ii. 영구적인 IP 주소
        iii. 데이터 크기 조정(scaling)을 위해 data center를 두기도 함.
    b. client
        i. server와 통신
        ii. 간헐적 연결(필요할때만 연결)
        iii. dynamic IP 주소 가져도 댐
        iv. client 간 통신 하지 않음
2. p2p (peer-to-peer)
    a. server가 없음 → 탈중앙화
    b. end systems(hosts) 들끼리 주고 받음
    c. 간헐적으로 IP가 바뀜.

Processes Communicating

1. process : 어떤 host 안에서 실행되는 프로그램
2. inter-process communication (ipc) : 같은 host 에서 2개의 process 진행
3. 다른 host 간 process 진행 할 때에는 message 를 교환하는 방식으로 진행
4. client-server
    a. client process : 통신을 처음 시작한 프로세스 (웹 브라우저 실행 등)
    b. server process : 상대방으로 부터 contact가 되길 기다리는 process
5. p2p 기준에서는 host 가 client, server process 를 다 다룰줄 알아야 함.

Sockets

- 구현은 application에서 , transport 에서는 tcp, udp 설정을 해야 하므로 그 중간단에 있음



1. process는 통신을 각 application process 들이 socket 을 통해서 message 를 주고 받는다.
2. Addressing Process
    a. socket 을 통해 다른 process 와 통신할 때 어떻게 그 주소를 알아내는지에 대한 방법
    b. 각각의 process 는 identifier가 필요하다. 또한 host 는 32비트 unique 한 IP address를 가진다.
    * IP adress : host device를 구별하기 위한 주소 (IPv4로 부족하기 때문에 IPv6가 나옴. (비용적 문제 등으로 아직 IPv6을 모두 사용하진 않음)
    c. 여러 process 가 진행되는 경우 IP 주소로만은 구분이 불가능하기 때문에 Port 라는 개념을 사용한다.
        i. HTTP server : 80
        ii. mail server(SMTP) : 25 등의 자주 사용하는 process 는 고정 포트(well-known port)를 사용한다.

Application-layer Protocol Defines (message를 주고 받기 위한 규칙)

1. types of messages exchanged (교환 타입)
   e.g. request (요청), response (응답)
2. message syntax (메세지 문법, 필드 정의)
3. message semantics (각각의 필드가 무슨 의미 인지)
4. rules (process 들이 언제 어떻게 페세지를 주고 받을지)

5. open protocols (공개된 프로토콜, RFC에 정의 되어 있음)
   e.g. HTTP, SMTP, 등
6. proprietary protocols(독점 프로토콜)
   e.g. Skype

What Trasport service does an app need?

1. data integrity : 데이터 무결성 → 데이터가 그대로 전달 되어야함
2. timing : 정확한 시간에 상대방에게 전달 되어야 함
3. throughput : 처리량(속도가 어느정도는 되어야함)
4. security : 보안 → 데이터 무결성도 중요하지만 권한이 없는 사람에게 정보가 가면 안 됨.

application	data loss	throughput	time sensitive?
file transfer/download	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5Kbps-1Mbps video:10Kbps-5Mbps	yes, 10's msec
streaming audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	Kbps+	yes, 10's msec
text messaging	no loss	elastic	yes and no

Internet transport protocals service

1. TCP (Transmission Control Protocol)
   a. reliable transport : **신뢰성!** (데이터 손실이 없음)
   b. flow control : sender 가 receiver를 넘으면 안 됨. 너무 빠르면 receiver가 다 받지 못함.
   c. congestion control : 네트워크 상황에 따라 혼잡 제어
   d. does not provide : timing, minimum throughput guarantee, security
   e. connection-oriented : 연결 지향적, data를 보내기 전에 process 끼리 미리 연결

2. UDP (User Datagram Protocol)
    a. unreliable data transfer : 데이터의 손실이 일어날 수 있음
       → flow control 이 udp에서 필요하다? 틀림!
    b. does not provide : reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup
    c. UDP가 더 안 좋은데 왜 쓰는 거지? → 훨씬 빠른 속도!

→ **TCP** 는 데이터 손실이 없는 **신뢰성**!

→ **UDP** 는 데이터의 손실이 일부 일어 날 수 있지만 **빠른 속도**!

application	application layer protocol	transport protocol
file transfer/download	FTP [RFC 959]	TCP
e-mail	SMTP [RFC 5321]	TCP
Web documents	HTTP 1.1 [RFC 7320]	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary	TCP or UDP
streaming audio/video	HTTP [RFC 7320], DASH	TCP
interactive games	WOW, FPS (proprietary)	UDP or TCP

→ data loss, throughput, time sensitive, 등을 고려하여 transport protocol 을 설정을 해준다.

Securing TCP

1. TCP & UDP
    a. 암호화 X
    b. password가 cleartext로 전달되어 sniffing 등에 취약
2. Transport Layer Security (TLS)
    a. TCP 연결을 암호화
    b. 서로 인증을 통해 데이터를 주고 받음