

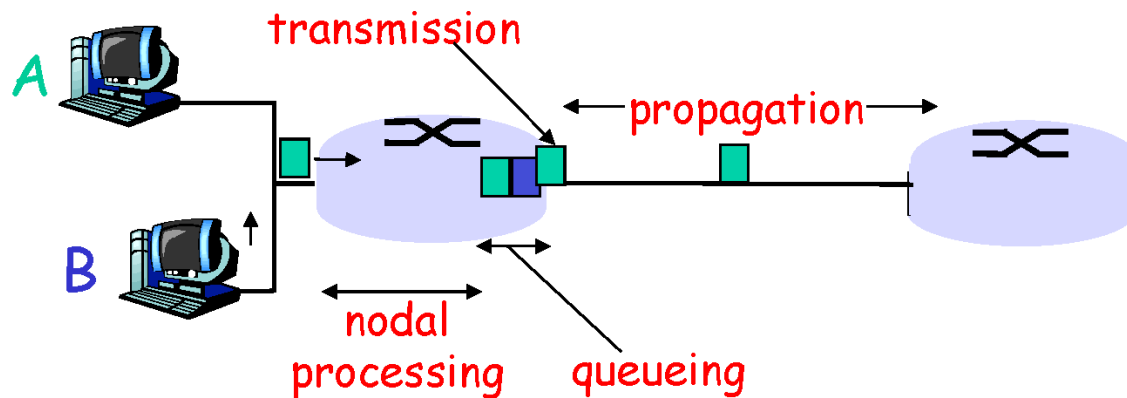
Four sources of packet delay

1. nodal processing:

- check bit errors
- determine output link

2. queueing

- time waiting at output link for transmission
- depends on congestion level of router



Delay in packet-switched networks

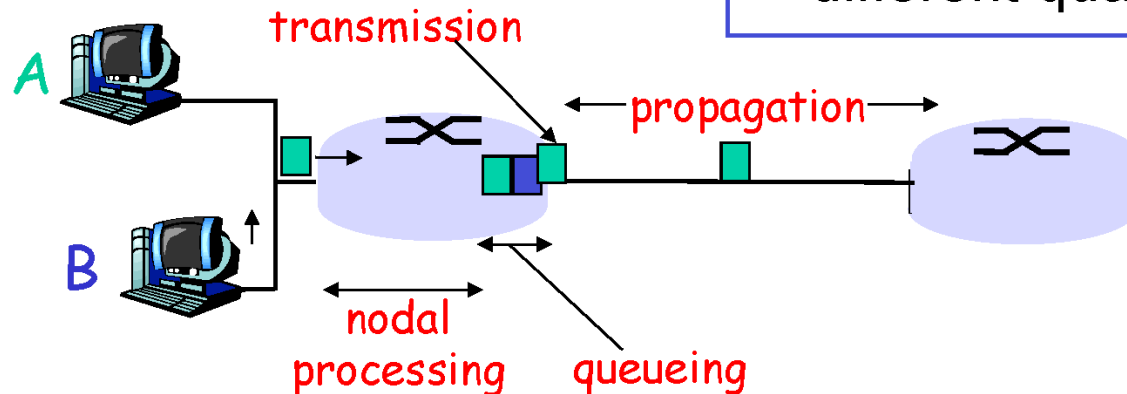
3. Transmission delay:

- R = link bandwidth (bps)
- L = packet length (bits)
- time to send bits into link = L/R

4. Propagation delay:

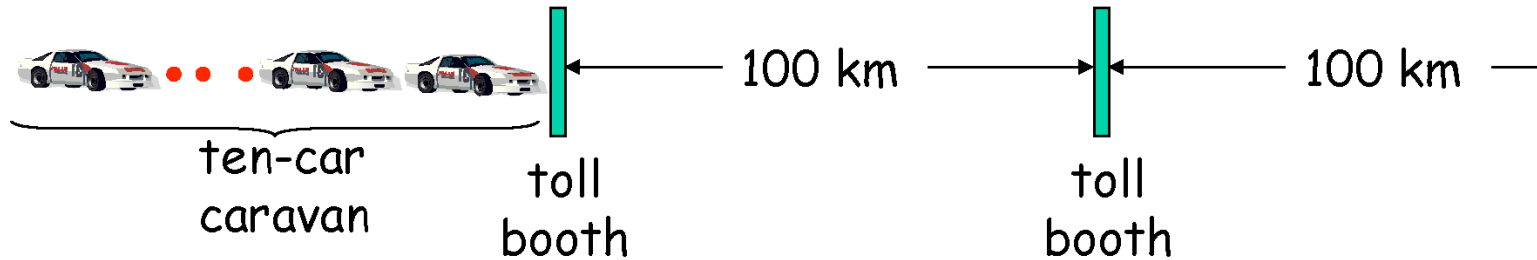
- d = length of physical link
- s = propagation speed in medium ($\sim 2 \times 10^8$ m/sec)
- propagation delay = d/s

Note: s and R are very different quantities!



Caravan analogy

package는 묶음으로 땀
뒤에 배는 다들 때까지 기다림



- ❑ Cars “propagate” at 100 km/hr
- ❑ Toll booth takes 12 sec to service a car (transmission time)
- ❑ car~bit; caravan ~ packet
- ❑ Q: How long until caravan is lined up before 2nd toll booth?
- ❑ Time to “push” entire caravan through toll booth onto highway = $12 \times 10 = 120$ sec
- ❑ Time for last car to propagate from 1st to 2nd toll booth: $100\text{km} / (100\text{km/hr}) = 1$ hr
- ❑ A: 62 minutes

layer

APP

HTTP

Transport

TCP / UDP

Network

IP

Link

WIFI , LTE , Ethernet , 3G

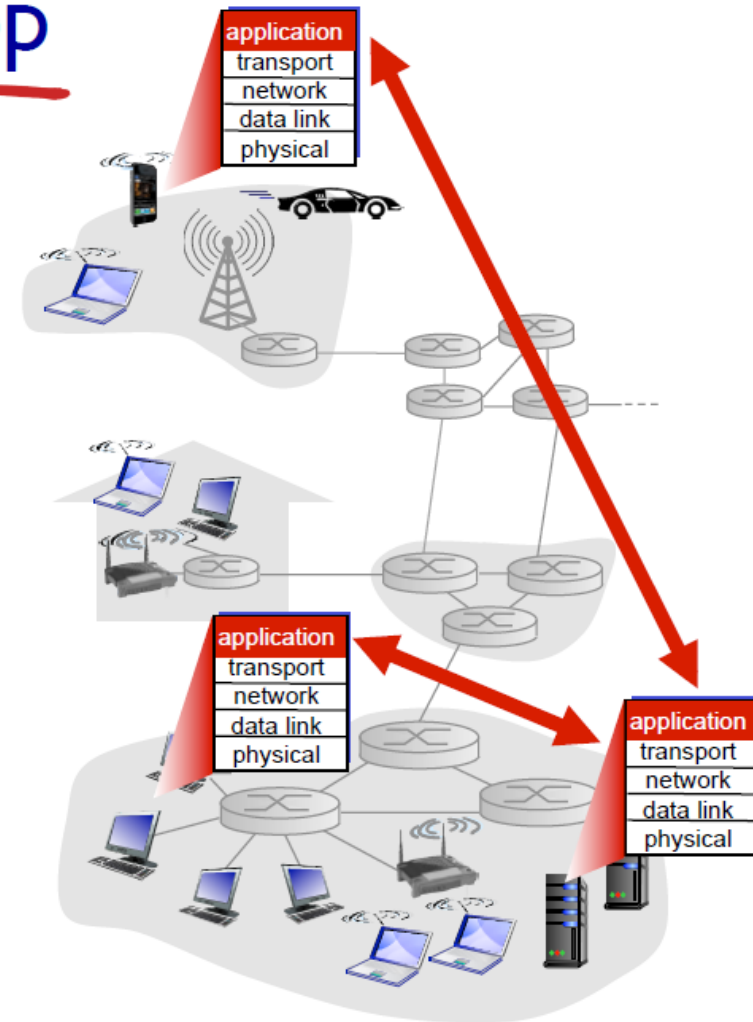
Creating a network app

write programs that:

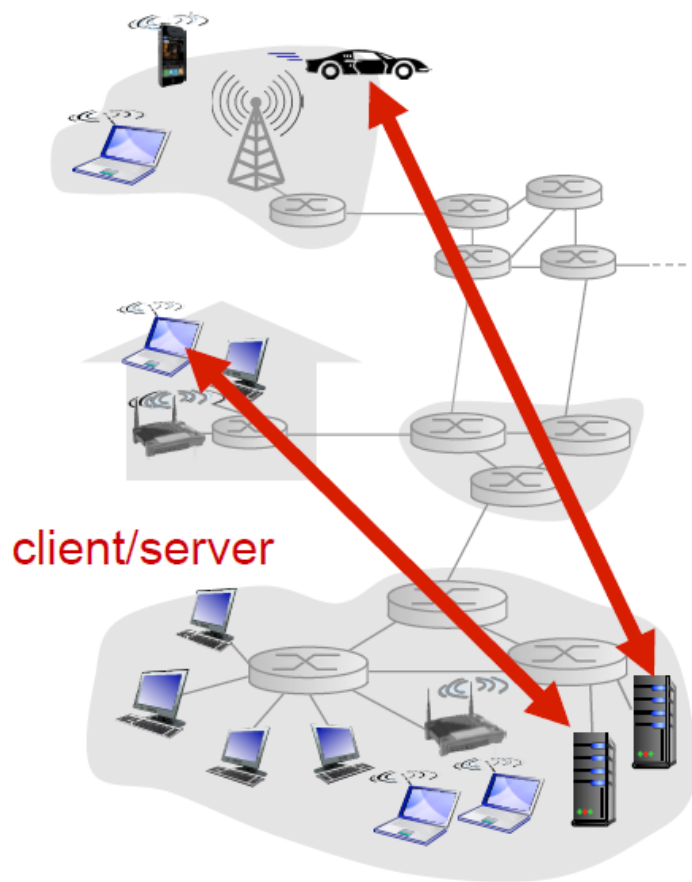
- ❖ run on (different) *end systems*
- ❖ communicate over network
- ❖ e.g., web server software communicates with browser software

no need to write software for
network-core devices

- ❖ network-core devices do not run user applications
- ❖ applications on end systems allows for rapid app development, propagation



Client-server architecture



client/server

24시간 동작해야 함

server:

- ❖ always-on host
- ❖ permanent IP address 고정 주소 가져야 함
- ❖ data centers for scaling

clients:

고정 주소 갖지 x 도 됨

- ❖ communicate with server
- ❖ may be intermittently connected ~ 간헐적으로
- ❖ may have dynamic IP addresses
- ❖ do not communicate directly with each other

Processes communicating

process: program running within a host

- ❖ within same host, two processes communicate using **inter-process communication** (defined by OS)
- ❖ processes in different hosts communicate by exchanging **messages**

clients, servers

client process: process that initiates communication

server process: process that waits to be contacted

- ❖ aside: applications with P2P architectures have client processes & server processes

모든 웹 서비스는 80번 포트 넘버 씬 why? 찾아가기 쉽게.

이렇게 반대와 1번 415에서

서버: 24시간 켜져 있어야 하고, 주소는 같아야 함

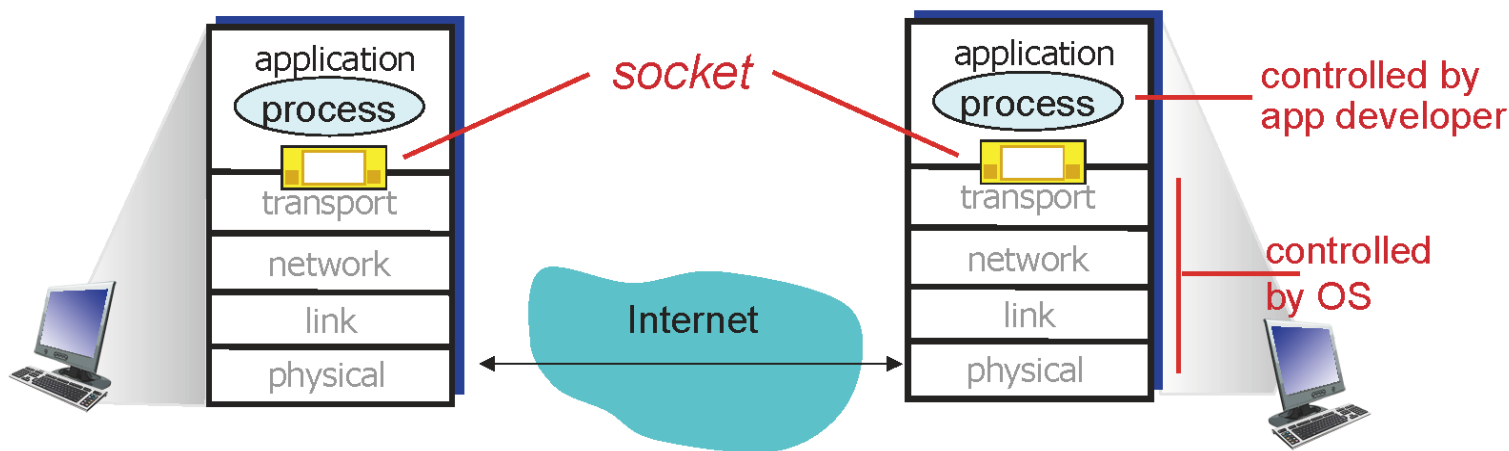
파라미터 다 해줌

APP은 Transport 계층에서 제공하는 서비스 씬

Sockets

IP address, Port
프로세스 주소 프로세스 중
어떤가?

- ❖ process sends/receives messages to/from its **socket**
- ❖ socket analogous to door **— 일치다**
 - sending process shoves message out door
 - sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process



What transport service does an app need?

현재 TCP: ✗
Transport에서
유일하게 제공

⇒ UDP는 이미지도 안해줌

data integrity 유실x

- ❖ some apps (e.g., file transfer, web transactions) require 100% reliable data transfer
- ❖ other apps (e.g., audio) can tolerate some loss

timing 시간 희망사항

- ❖ some apps (e.g., Internet telephony, interactive games) require low delay to be “effective”

throughput 용량

- ❖ some apps (e.g., multimedia) require minimum amount of throughput to be “effective”
- ❖ other apps (“elastic apps”) make use of whatever throughput they get

security 보안

- ❖ encryption, data integrity, ...

Internet apps: application, transport protocols

application	application layer protocol	underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

Web and HTTP

First, a review...

- ❖ *web page* consists of *objects*
- ❖ object can be HTML file, JPEG image, Java applet, audio file,...
- ❖ web page consists of *base HTML-file* which includes *several referenced objects*
- ❖ each object is addressable by a *URL*, e.g.,

`www.someschool.edu/someDept/pic.gif`

host name

path name

HTTP overview

HTTP: hypertext transfer protocol

- ❖ Web's application layer protocol
- ❖ client/server model
 - **client:** browser that requests, receives, (using HTTP protocol) and "displays" Web objects
 - **server:** Web server sends (using HTTP protocol) objects in response to requests

TCP AND
TCP connection



HTTP overview (continued)

— 7/18/19

uses TCP:

- ❖ client initiates TCP connection (creates socket) to server, port 80
- ❖ server accepts TCP connection from client
- ❖ HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- ❖ TCP connection closed

HTTP is “stateless”

- ❖ server maintains no information about past client requests

aside protocols that maintain “state” are complex!

- ❖ past history (state) must be maintained
- ❖ if server/client crashes, their views of “state” may be inconsistent, must be reconciled

HTTP connections

non-persistent HTTP

- ❖ at most one object sent over TCP connection
 - connection then closed
- ❖ downloading multiple objects required multiple connections

persistent HTTP

- ❖ multiple objects can be sent over single TCP connection between client, server

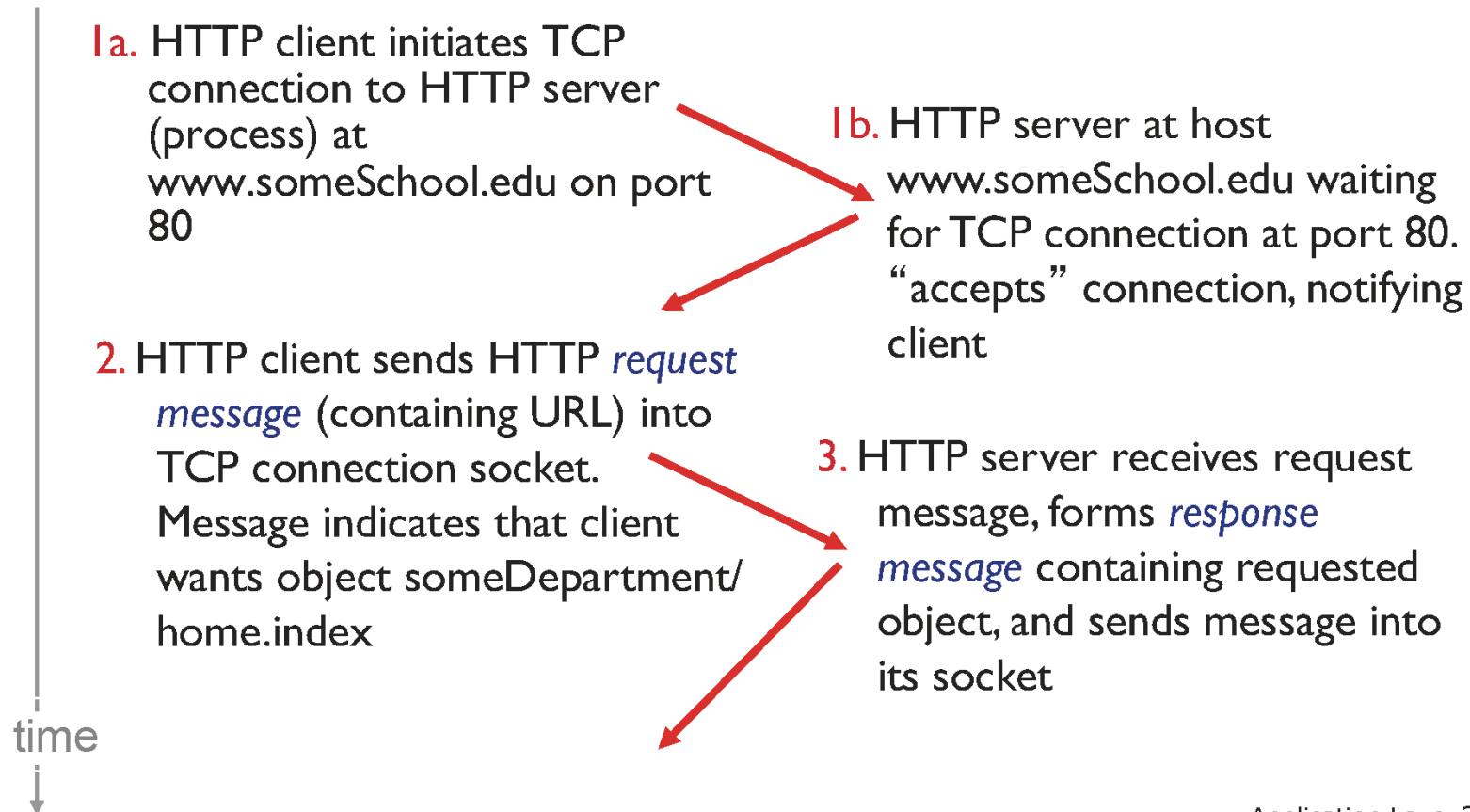
주소 받고 TCP 풀리면 non-persistent
계속 유지되면 persistent

Non-persistent HTTP

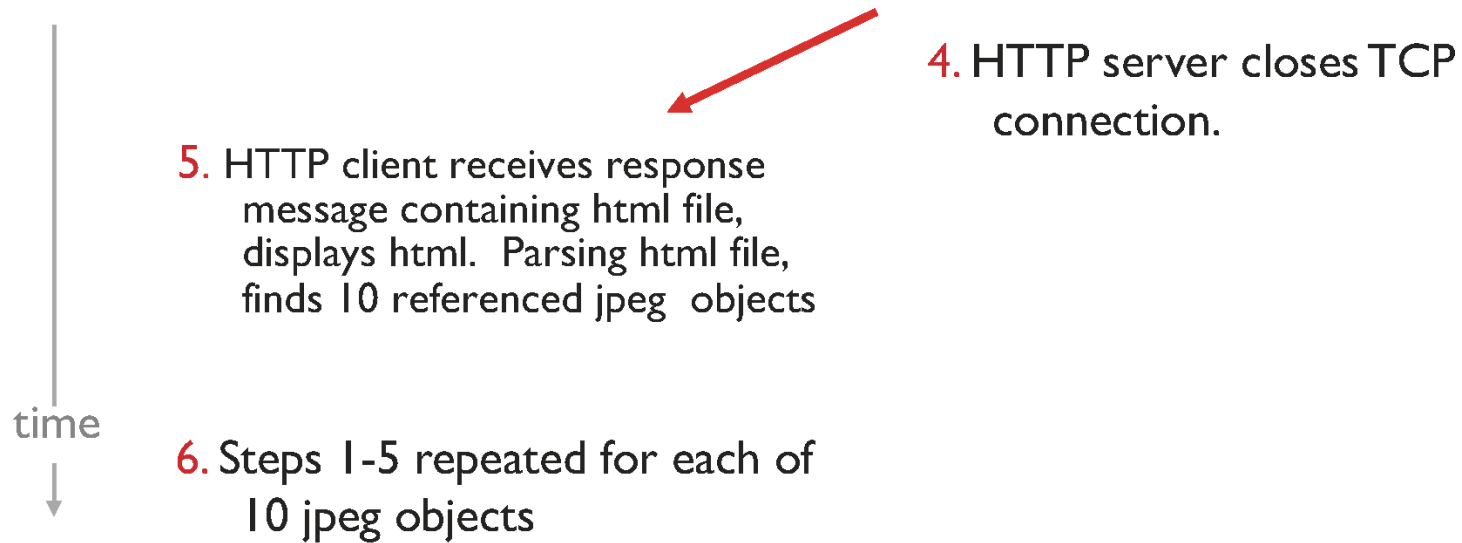
suppose user enters URL:

`www.someSchool.edu/someDepartment/home.index`

(contains text,
references to 10
jpeg images)



Non-persistent HTTP (cont.)



현재 web browser 는 persistent 슴

Non-persistent HTTP: response time

RTT (definition): time for a small packet to travel from client to server and back

HTTP response time:

- ❖ one RTT to initiate TCP connection
- ❖ one RTT for HTTP request and first few bytes of HTTP response to return
- ❖ file transmission time
- ❖ non-persistent HTTP response time =
 $2\text{RTT} + \text{file transmission time}$

