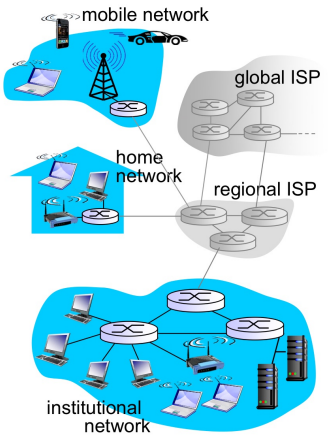
Basics

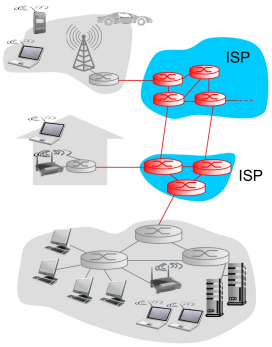
- Internet: a network of connected computing devices (eg. PC, Server, laptop)
  - hosts / end systems.
- Hosts:
  - run network applications, (eg. Whatsapp, browser, zoom)
    Communicate over links.
  - access the Internet through access networks. (network edge)
    - home network
    - enterprise access networks (Ethernet)
    - wireless access networks
      - Connects hosts to router via access points
  - Connects to access network over diff physical media.
    - Guided media
      - Signal propagate in solid media.
      - eg. Fibre optics.
    - Unguided media
      - Signal propagate freely
      - eg. radio.

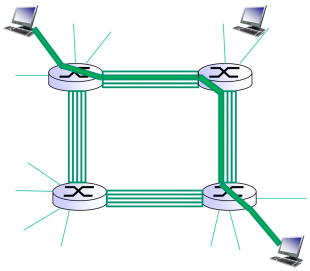
Network Edge



- Network Core
  - A mesh of interconnected routers
  - Data transmitted via
    - Circuit Switching – dedicated circuit per call.
      - end-end resources allocated to and reserved for "call" between source & dest.
        - Call setup required.
        - Circuit-like (guaranteed) performance
        - Circuit segment idle if not used by call (no sharing)
        - Commonly used in traditional telephone networks

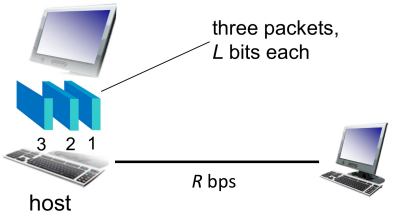
Network Core





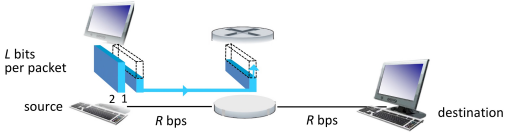
Circuit Switching.

    - Packet Switching – data sent through network in discrete "chunks"
      - Host sending function:
        - breaks application message into smaller chunks (packets) of length L bits. (sequence of bits)
        - transmits packets onto the link at transmission rate R.
          - link transmission rate = link capacity = link bandwidth
          - i.e. how fast to push packet onto link.
          - packet transmission delay = time needed to transmit L-bit packet into link = $\frac{L \text{ (bits)}}{R \text{ (bits/sec)}}$



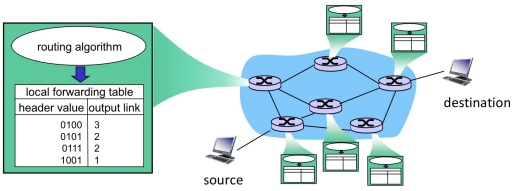
Packet switching.

      - Packets are passed from one router to the next, across links on path from source to destination.
      - Store and Forward – entire packet must arrive at a router before it can be transmitted on the next link

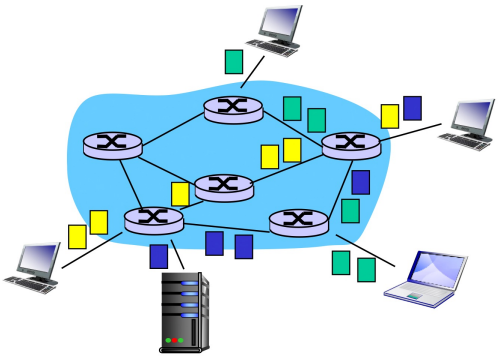


End-to-end delay = 2*L/R (assuming no other delay)

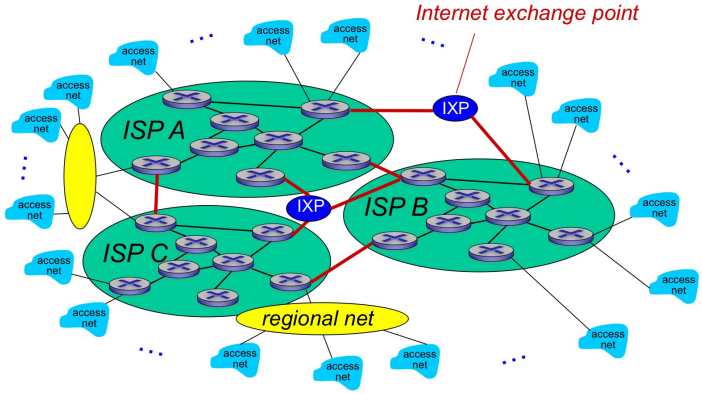
- Packet Switching
  - Routing and Addressing.
    - Routers determines source-destination route taken by packets
      - Routing algorithms.
    - Addressing — each packet needs to carry source and destination information.

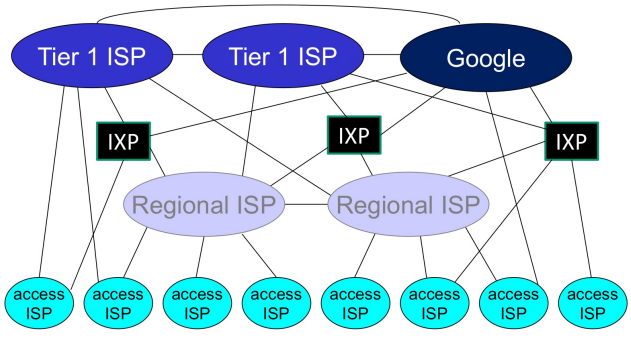


  - Summary
    - The Internet is a packet switching network
    - User A, B, ... 's packets share network resources
    - Resources are used on demand
    - Excessive congestion is possible
    - No:
      - Bandwidth division into "pieces"
      - Dedicated allocation
      - Resource reservation



- Internet Structure: Network of Networks
  - Hosts connect to Internet via access ISPs (Internet Service Providers)
    - eg. Residential, Company and University ISPs
  - Access ISPs in turn must be interconnected.
  - Resulting network of networks is very complex
    - evolution was driven by economics and national policies.
  - ∴ The Internet is a "network-of-networks" organised into autonomous systems (AS), each owned by an organisation.



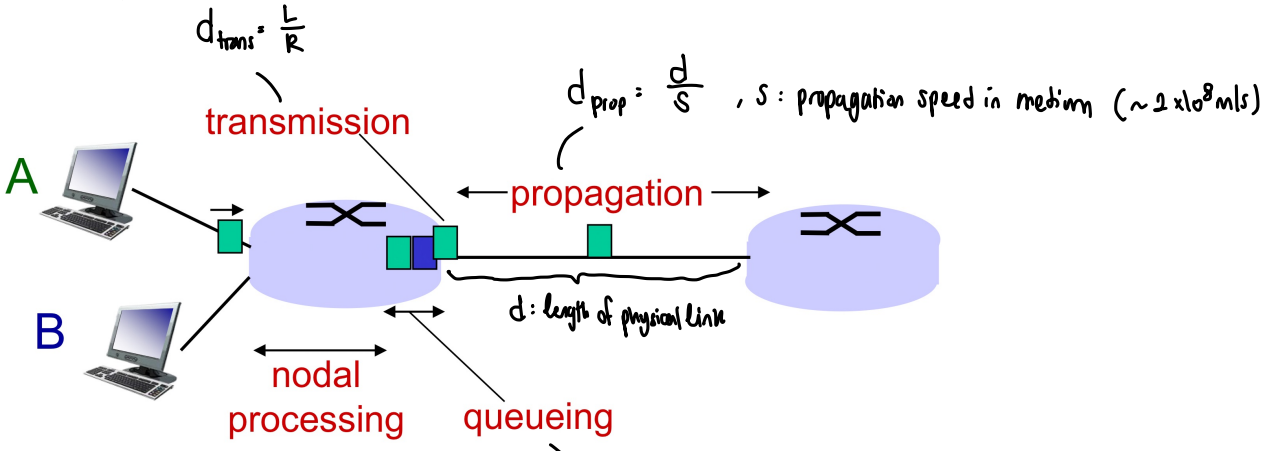


Who runs the Internet?

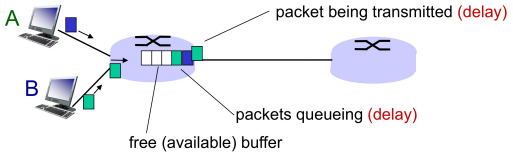
- IP address & Internet Naming administered by Network Information Centre (NIC)
  - Refer to: www.sgnic.net.sg; www.apnic.org
- The Internet Society (ISOC) - Provides leadership in Internet related standards, education, and policy around the world.
- The Internet Architecture Board (IAB) - Authority to issue and update technical standards regarding Internet protocols.
- Internet Engineering Task Force (IETF) - Protocol engineering, development and standardization arm of the IAB.
  - Internet standards are published as RFCs (Request For Comments)
    - Refer to: www.ietf.org; for RFCs: http://www.ietf.org/rfc.html

Delay and Loss

- Packet Delay
  - L: packet length (bits)
  - R: link bandwidth (bps)

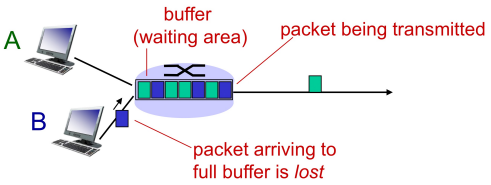


- Packets queue in router buffers
  - wait for turn to be sent out one by one



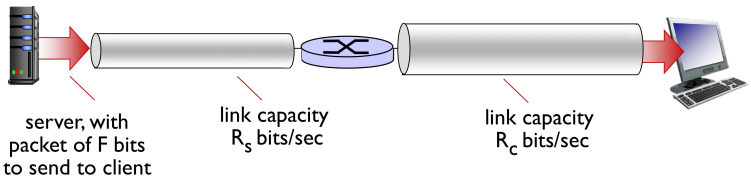
Queue (or buffer) of a router has finite capacity

- Packet arriving to full queue will be dropped (lost).



- Throughput
  - How many bits can be transmitted per unit time
  - Measured for end-end communication.
    - time taken for a packet to travel from source to destination. (may have all the delays)

(Link capacity / bandwidth is meant for a specific link)



Metric Units

- 1 byte = 8 bits

Exp.	Explicit	Prefix	Exp.	Explicit	Prefix
10^{-3}	0.001	milli	10^{3}	1,000	Kilo
10^{-6}	0.000001	micro	10^{6}	1,000,000	Mega
10^{-9}	0.000000001	nano	10^{9}	1,000,000,000	Giga
10^{-12}	0.000000000001	pico	10^{12}	1,000,000,000,000	Tera
10^{-15}	0.000000000000001	femto	10^{15}	1,000,000,000,000,000	Peta
10^{-18}	0.000000000000000001	atto	10^{18}	1,000,000,000,000,000,000	Exa
10^{-21}	0.000000000000000000001	zepto	10^{21}	1,000,000,000,000,000,000,000	Zetta
10^{-24}	0.000000000000000000000001	yocto	10^{24}	1,000,000,000,000,000,000,000,000	Yotta

The principal metric prefixes

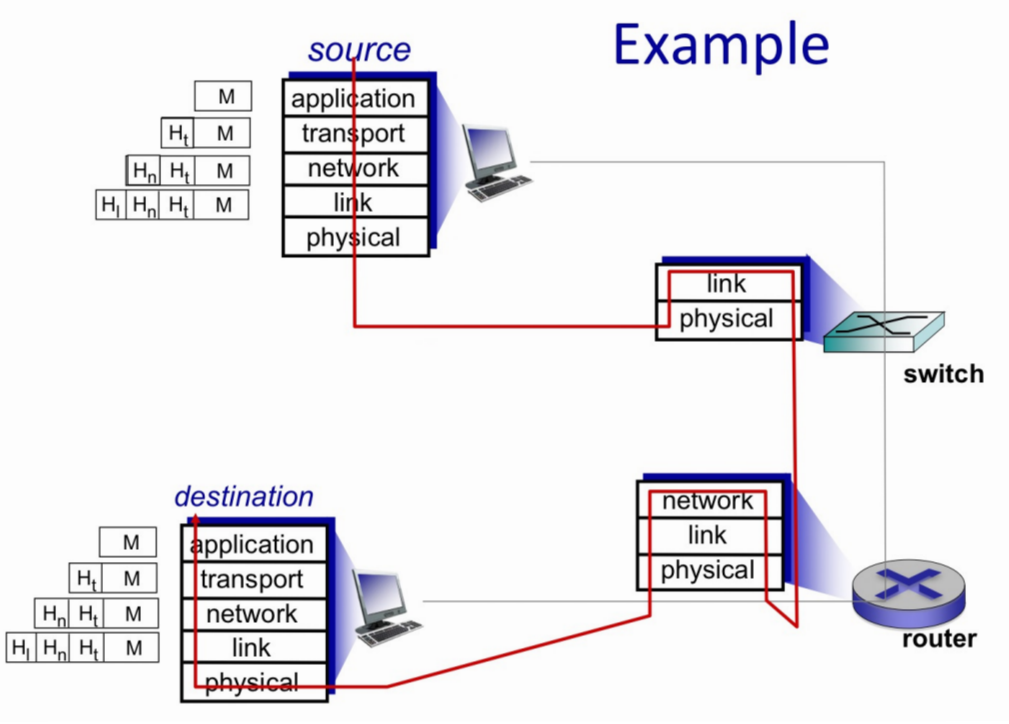
Network Protocols

- The internet supports various kinds of network applications
  (eg. Web, VoIP, email, games ...)

- Network applications exchange messages and communicate among peers according to **protocols**.
  - It defines **format** and **order** of messages exchanged and the **actions** taken after messages are sent or received.

Protocols in the Internet are logically organised into 5 layers:
(According to their purposes)

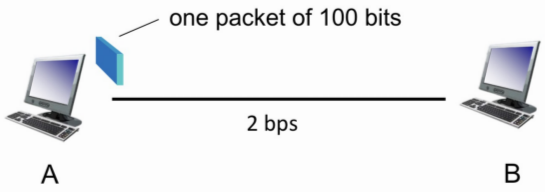
- **application**: supporting network applications.
  - FTP, SMTP, HTTP.
- **transport**: process-to-process data transfer
  - TCP, UDP.
- **network**: routing of datagrams from source to destination
  - IP, routing protocols.
- **link**: data transfer between neighbouring network elements
  - Ethernet,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



Qns on Delay

Q1

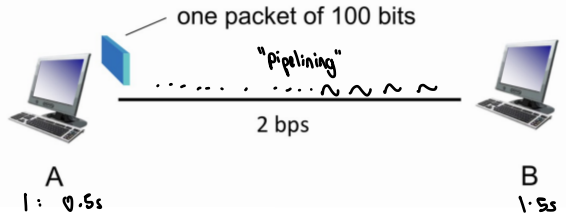
Hosts A and B are connected by a direct link of 2 bps. A sends a packet of 100 bits to B. Ignore other unmentioned delay, when will B receive the packet?



100 bits / 2 bps = 50s. //

Q2

Hosts A and B are connected by a direct link of 2 bps. A sends a packet of 100 bits to B. The propagation delay over the link is 1s. Ignore other unmentioned delay, when will B receive the packet?



bits 1: 0.5s
2: 1s
3: 1.5s

1.5s
2s
2.5s

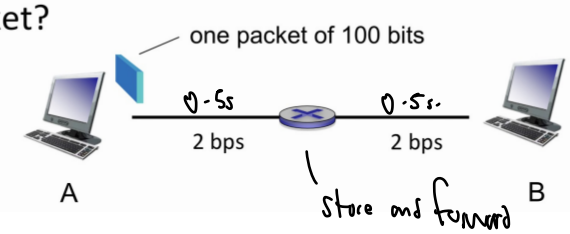
$\frac{100}{2} + 1 = 51s.$

d_{trans} d_{prop}

Q3

Hosts A and B are connected by a router in between. The bandwidth of each link is 2 bps. Propagation delay over each link is 0.5 s.

A sends a packet of 100 bits to B. When will B receive the packet?



store and forward

↳ check integrity of package.

Last bit.

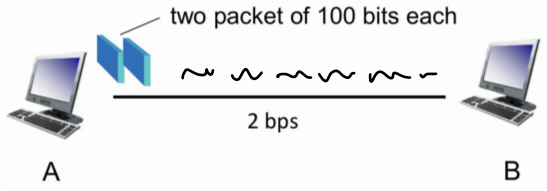
leaves A at 50s — d_{prop1} — leaves router. — d_{prop2}

$(100/2 + 0.5) + (100/2 + 0.5)$

$= 101s.$

Q4

Hosts A and B are connected by a direct link of 2 bps. A sends two consecutive packets of 100 bits each to B. The propagation delay over the link is 1s. Ignore other unmentioned delay, when will B receive both packets?



100/2 → first packet leaves A, start sending 2nd packet

+

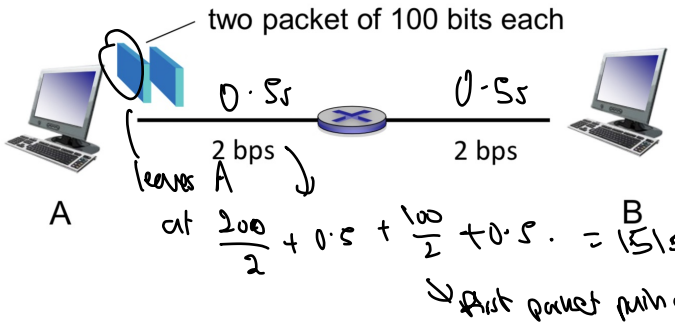
100/2 → will take this long for d_{trans}.

+

1 → d_{prop} = 101s.

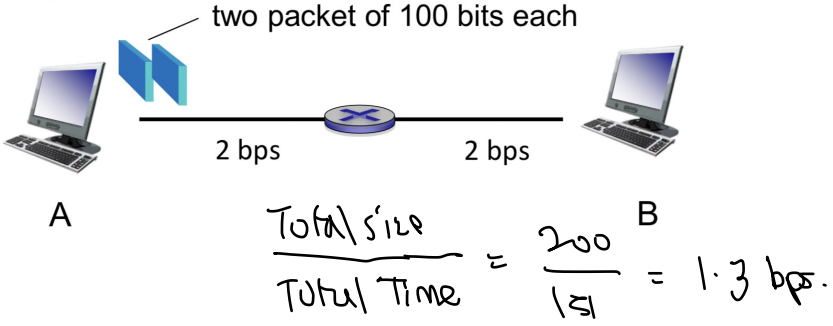
Q5

Hosts A and B are connected by a router in between. The bandwidth of each link is 2 bps. Propagation delay over each link is 0.5 s. A sends two consecutive packets of 100 bits each to B. When will B receive two packets?



Q6

Hosts A and B are connected by a router in between. The bandwidth of each link is 2 bps. Propagation delay over each link is 0.5 s. A sends two consecutive packets of 100 bits each to B. What is the throughput of transmission?



A packet switch receives a packet and determines the outbound link to which the packet should be forwarded. When the packet arrives, one other packet is **halfway done being transmitted** on this outbound link and **four other packets are waiting** to be transmitted. Packets are transmitted in order of arrival. Suppose all packets are 1,500 bytes and the link rate is 2 Mbps.

a) What is the queuing delay for the packet?

Link rate, R= 2Mbps, Packet Length, L = 1500 bytes

Bytes of packets remaining to be send : (0.5 +4) *1500 = 6,750 bytes or 54,000 bits

Since these bits are transmitted at 2 Mbps,

Queuing delay is = 54000 / (2*10^6) = 27 msec. **(length of packet remaining/rate)**

b) More generally, what is the queuing delay when all packets have length ***L*** (bits), the transmission rate is ***R***, ***x*** bits of the currently-being-transmitted packet have been transmitted, and ***n*** packets are already in the queue?

Remaining bits in currently transmitting packet = L - x

Number of bits for n packets in queue = n.L

Generally, queuing delay *is* $\frac{((n \times L) + (L - x))}{R}$.

Application Layer

- Network applications run on (different) hosts, communicate over network (eg. web server software ⇔ browser software)
- Classic structure of network applications: Client-server, Peer-to-peer (P2P)

- **Client-Server Architecture** *
  - **Server**:
    - Waits for incoming requests
    - provides requested service to client
    - data centres for scaling.
  - **Client**:
    - Initiates contact with server ("speaks first")
    - Typically requests service from server
    - For Web, client is usually implemented in the browser.

- **P2P Architecture** *
  - No always-on server
  - Arbitrary end systems directly communicate
  - Peer request service from other peers, provide service in return to other peers
    - **Self scalability** – new peers bring new service capacity, as well as new service demands.
  - Peers are intermittently connected and change IP addresses.
    - Complex management

- Transport Service for an app:
  - **Data Integrity** (reliability) *
    - Some apps (eg. file transfer, web transactions) require 100% reliable data transfer.
    - other apps (eg. audio streaming) can tolerate some data loss.
  - **Timing** (delay) *
    - some apps (eg. online interactive games) require low delay to be "effective".
  - **Throughput** *
    - Some apps (eg. multimedia) require minimum amount of bandwidth to be "effective"
    - Other apps (eg. file transfer) make use of whatever throughput available
  - **Security** *
    - encryption, data integrity, authentication...

Application	Data loss	Throughput	Time-sensitive
File transfer	No loss	Elastic	No
Electronic mail	No loss	Elastic	No
Web documents	No loss	Elastic	No
Real-time audio/video	Loss-tolerant	Audio: 5kbps-1Mbps Video:10kbps-5Mbps	Yes: 100s of msec
Stored audio/video	Loss-tolerant	Same as above	Yes: few seconds
Interactive games	Loss-tolerant	Few kbps – 10 kbps	Yes: 100s of msec
Text messaging	No loss	Elastic	Yes and no

- App-layer protocol define:
  - types of messages exchanged
    - eg. request, response
  - message syntax
    - what fields in messages & how fields are delinated.
  - message semantics
    - meaning of information in fields
  - rules for when and how applications send and response to messages.

- Open protocols
  - defined in RFCs
  - allows for interoperability
  - eg. HTTP, SMTP...
- proprietary protocols
  - eg. skype.

- App-layer protocols ride on transport layer protocols:
  - TCP service*: (connection-oriented)
    - reliable data transfer
    - flow control
      - sender won't overwhelm receiver
    - congestion control
      - throttle sender when network is overloaded
    - does not provide:
      - timing / minimum throughput guarantee, security.

UDP service*: (connection-less)
- unreliable data transfer
- no flow control.
- no congestion control
- does not provide:
  - timing / throughput guarantee, security.

Application	Application Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP [RFC 5321]	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)	Typically TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary (e.g., Skype)	TCP or UDP

Web / HTTP *

- A web page typically consists of:
  - base HTML file
  - several referenced objects
    - HTML file / JPEG image / Java applet / audio file
  - each object is addressable by a URL
    - e.g. www.comp.nus.edu.sg / ~cs2105 / img / done.jpg
      - www.comp.nus.edu.sg: host name
      - ~cs2105 / img / done.jpg: path name

- HTTP
  - Hypertext transfer protocol
  - Web's application layer protocol
  - Client / Server model
    - client: usually is browser that requests, receives and displays Web objects.
    - server: Web server sends objects in response to requests.
  - http 1.0: RFC 1945 ; http 1.1: RFC 2616

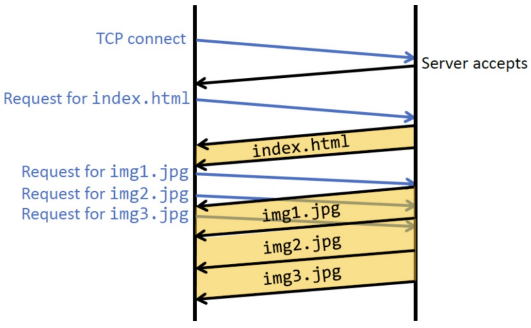
- HTTP over TCP
  - Client initiates TCP connection to server.
  - Server accepts TCP connection request from client.
  - HTTP messages are exchanged between browser (HTTP client) and Web server (HTTP server) over TCP connection.
  - TCP connection closed.

Non-persistent HTTP (1.0)

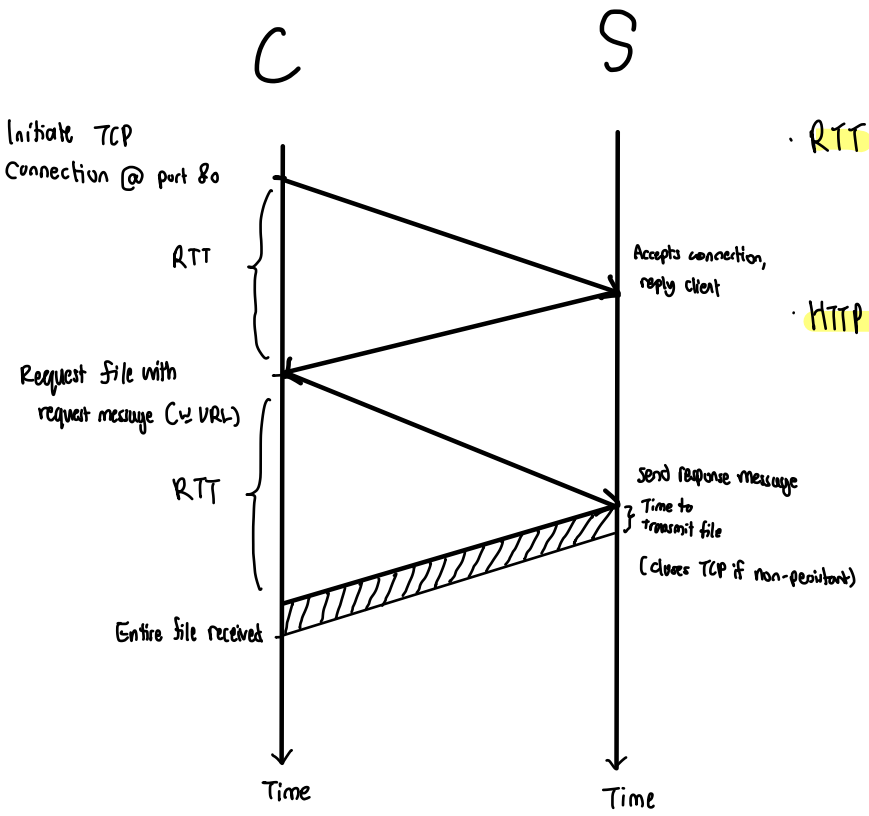
- at most one object sent over a TCP connection
  - connection then closed.
- downloading multiple objects required multiple connections.
- requires 2 RTTs per object
- OS overhead for each TCP connection
- browser often open parallel TCP connections to fetch referenced objects.

Persistent HTTP (1.1)

- multiple objects can be sent over single TCP connection between client, server.
- server leaves connection open after sending response.
- subsequent HTTP messages between same client / server sent over the same TCP connection
- moreover, client may send requests as soon as it encounters a referenced object (persistent with pipelining)
  - as little as one RTT for all the referenced objects.



- HTTP example:



- RTT: time for a packet to travel from client to server and go back.

- HTTP response time:
  - one RTT to establish TCP connection
  - one RTT for HTTP request and the first few bytes of HTTP response to return
  - file transmission time
  - non-persistant HTTP response time
    = 2 * RTT + file transmission time.

- HTTP Messages

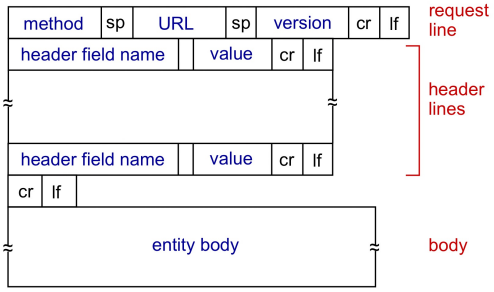
- 2 types: request, response.

request line (GET method)

```
GET /index.html HTTP/1.1\r\n
Host: www.example.org\r\n
Connection: keep-alive\r\n
...
\r\n
```

header lines

Extra "blank" line indicates the end of header lines



status line (protocol status code)

```
HTTP/1.1 200 OK\r\n
Date: Wed, 23 Jan 2019 13:11:15 GMT\r\n
Content-Length: 606\r\n
Content-Type: text/html\r\n
...
\r\n
data data data data data ...
```

header lines

data, e.g. requested HTML file

- Status code appears in 1st line in server-to-client response message.
- Some sample codes:
  - **200 OK**
    - request succeeded, requested object later in this msg
  - **301 Moved Permanently**
    - requested object moved, new location specified later in this msg (Location:)
  - **403 Forbidden**
    - server declines to show the requested webpage
  - **404 Not Found**
    - requested document not found on this server

- HTTP Request Method Types
  - HTTP/1.0
    - GET
    - POST
      - Web page often includes form input
      - Input is uploaded to server in entity body.
    - HEAD
      - ask server to leave requested object out of response.
  - HTTP/1.1
    - GET, POST, HEAD
    - PUT
      - upload file in entity body to path specified in URL field.
    - DELETE
      - deletes file specified in the URL field.

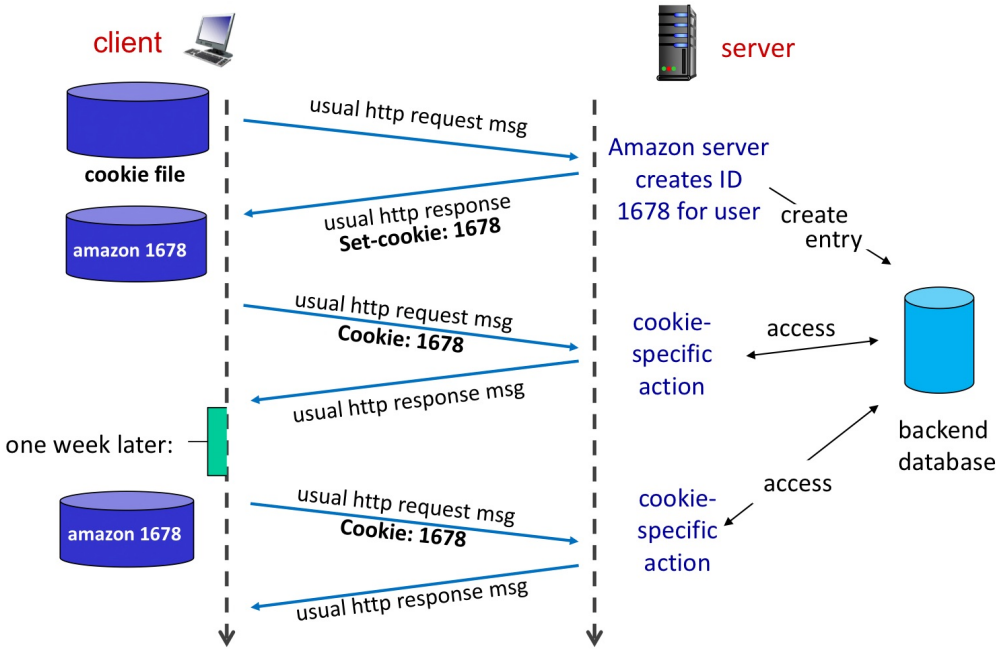
Cookies

HTTP is designed to be "stateless"

- Server maintains no information about past client requests.

- Sometimes its good to maintain states (history) at server/client over multiple transactions.
  - eg. shopping carts.

- Cookies: http messages carry "state".
  1. cookie header field of HTTP request/response messages.
  2. Cookie file kept on user's host, managed by user's browser.
  3. back-end database at Website.

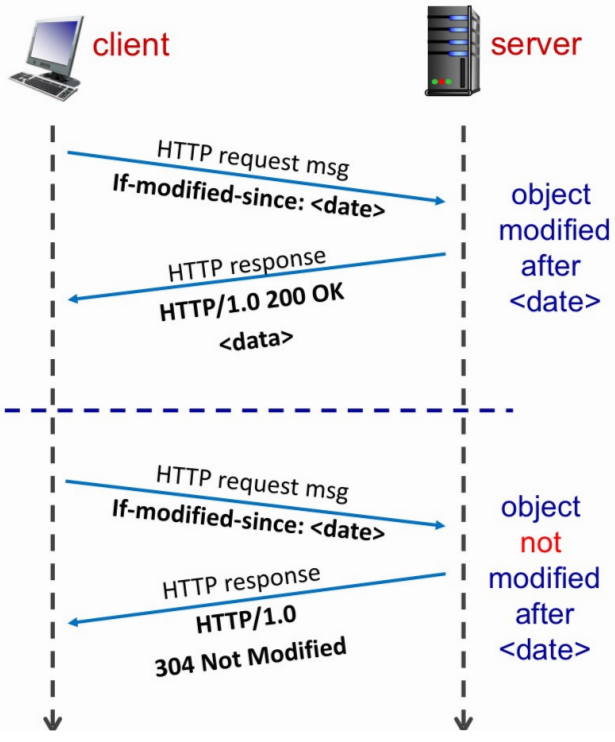


Conditional GET

- *Goal:* don't send object if (client) cache has up-to-date cached version
- *cache:* specify date of cached copy in HTTP request

  `If-modified-since: <date>`
- *server:* response contains no object if cached copy is up-to-date:

  `HTTP/1.0 304 Not Modified`



DNS *

- Domain Name System
- 2 ways to identify a host:
  1. Hostname: www.example.org.
  2. IP address: 93.184.216.34.

  } get mapping with nslookup
- DNS translates between the 2.

  A client must carry out a DNS query to determine the IP address corresponding to the server name (eg. www.example.org) prior to the connection.

DNS Resource Records (RR)

- Mapping between host names and IP addresses (and others) are stored as resource records (RR)
- RR format: (name, value, type, ttl)

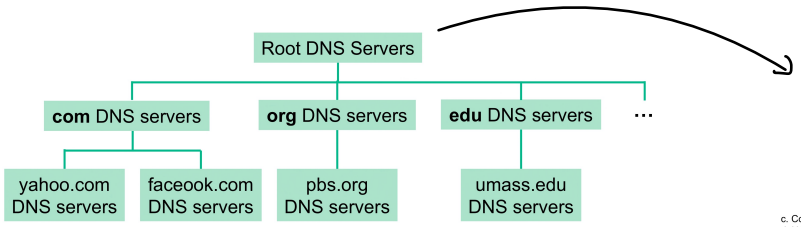
A
- name is hostname
- value is IP address

NS
- name is domain (eg. nus.edu.sg)
- value is hostname of authoritative name server for this domain.

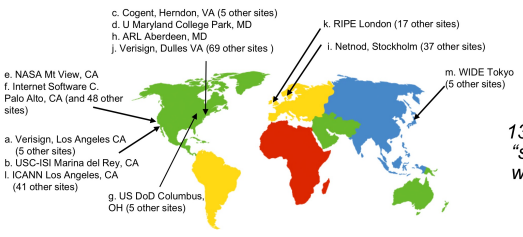
CNAME
- name is alias name (eg. www.nus.edu.sg) for some "canonical" (the real) name.
- value is canonical name (eg. myorzsqc.x.incapdns.net)

MX
- value is name of mail server associated with name.

DNS stored RR in distributed databases implemented in hierarchy of many name servers



- Answers requests for records in the root zone by returning a list of the authoritative name servers for the appropriate top-level domain (TLD)

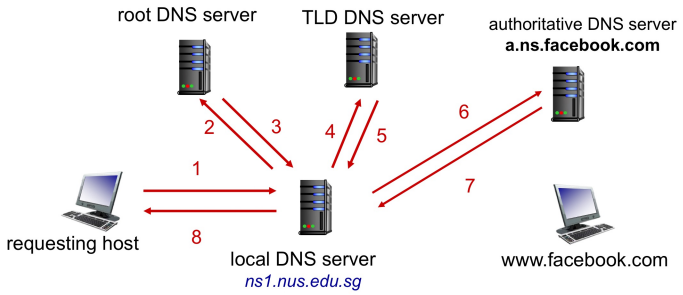


13 root name "servers" worldwide

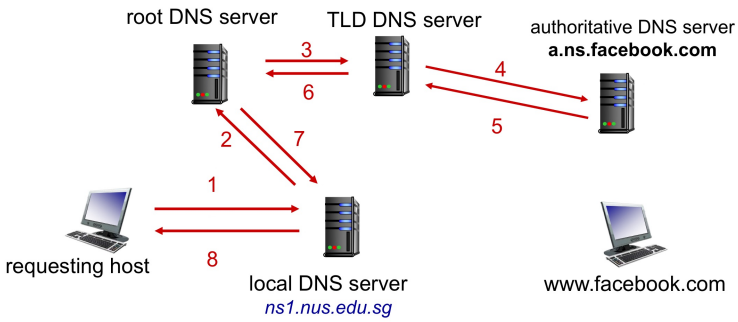
- eg A client wants IP address for www.facebook.com:
  1. Client queries root server to find .com DNS server.
  2. Client queries .com DNS server to get facebook.com DNS server
  3. Client queries facebook.com DNS server to get IP address for www.facebook.com.

- Top-level domain (TLD) servers:
  - responsible for com, org, net, edu, ... and all top-level country domains eg. uk, sg, jp...
- Authoritative servers:
  - organisation's own DNS server(s), providing authoritative hostname to IP mappings for organisation's named hosts (eg. Web, mail)
  - can be maintained by organization or service provider.
- Local DNS server
  - Does not strictly belong to hierarchy
  - Each ISP (residential, company, university) has one local DNS server.
    - aka "default name server"

  When a host makes a DNS query, query is sent to its local DNS server.
  - Retrieve name-to-address translation from local cache.
  - Local DNS acts as proxy and forwards query into hierarchy if answer is not found locally.
- DNS caching
  - Once a name server learns mapping, it caches mapping.
    - Cached entries may be out-of-date (best-effort name-to-address translation!)
    - cached entries expire after some time (TTL).
    - if name host changes IP address, may not be known Internet-wide until all TTLs expire.
  - DNS runs over UDP.
- DNS Name Resolution



  - Iterative Query.

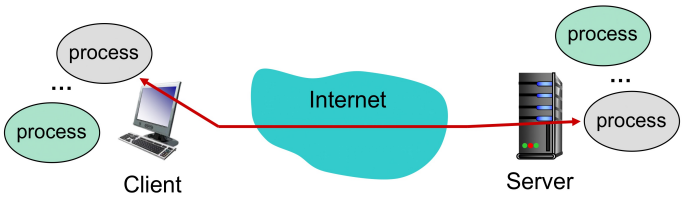


  - Recursive Query (rarely used).

Socket Programming

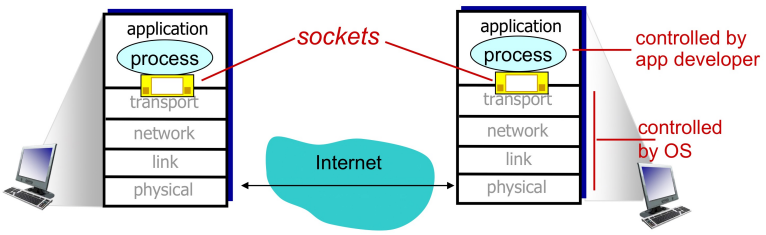
- Processes
  - Process: program running within a host
    - Within a same host, 2 processes communicate using inter-process communication (defined by OS)
    - Processes in different hosts communicate by exchanging messages (according to protocols)
    - In C/S model:
      - server process waits to be contacted.
      - client process initiates the communication.
  - Addressing processes

    IP address is used to identify a host
    - A 32-bit integer (eg. 137.132.21.27)
    - IP address of a host is not suffice to identify a process running inside the host.
      - Since many processes may run concurrently in a host.



      - Solution: port number
    - A process is identified by (IP address, port number)
      - Port number is 16-bit integer (1-1023 are reserved for standard use)
    - Example port numbers:
      - HTTP server: 80
      - SMTP server: 25.
    - IANA coordinates assignment of port numbers.

- Sockets — a combination of IP address and Port number.
  - A socket is the software interface between app processes and transport layer protocols.
    - Processes sends/receive messages to/from its socket.
    - Programming-wise : a set of APIs.



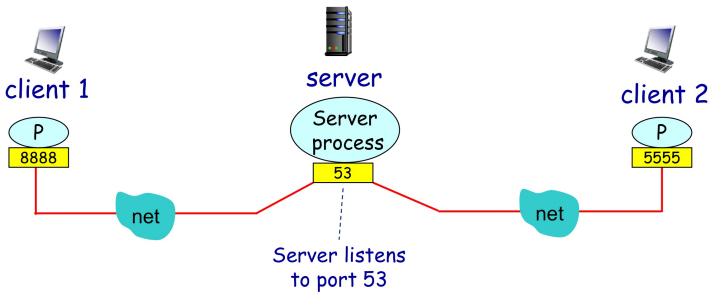
  - Socket Programming
    - Applications (or processes) treat the Internet as a black box, sending and receiving messages through sockets.
    - 2 types of sockets
      - TCP: reliable, byte-stream oriented socket
      - UDP: unreliable datagram socket.
    - UDP: no "connection" between client and server
      - Sender (client) explicitly attaches destination IP address and port number to each packet.
      - Receiver (server) extracts sender IP address and port number from received packet.

Example: UDP Echo Server

```
from socket import *                    ← include Python's socket library

serverPort = 2105
                                  IPv4      UDP socket
# create a socket
serverSocket = socket(AF_INET, SOCK_DGRAM)

# bind socket to local port number 2105
serverSocket.bind(('', serverPort))
print('Server is ready to receive message')
                                                  receive datagram
                                                  buffer size: 2048B
# extract client address from received packet
message, clientAddress = serverSocket.recvfrom(2048)

serverSocket.sendto(message, clientAddress)

serverSocket.close()
```

Example: UDP Echo Client

```
from socket import *

serverName = 'localhost' # client, server on the same host
serverPort = 2105

clientSocket = socket(AF_INET, SOCK_DGRAM)

message = input('Enter a message: ')
                                          convert message
                                          from string to
                                          bytes and send it
# send msg to server address
clientSocket.sendto(message.encode(), (serverName, serverPort))

receivedMsg, serverAddress = clientSocket.recvfrom(2048)

print('from server:', receivedMsg.decode())
                                          convert from bytes
                                          to string
clientSocket.close()
```



Server

Create serverSocket,
port = x

read datagram
from serverSocket

write reply to serverSocket
specifying client address, port number.

Client

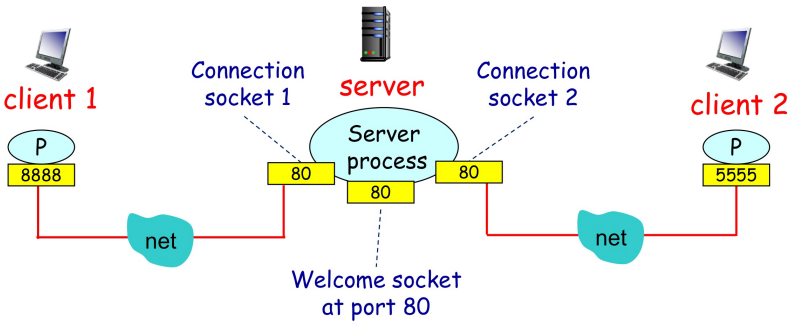
Create clientSocket

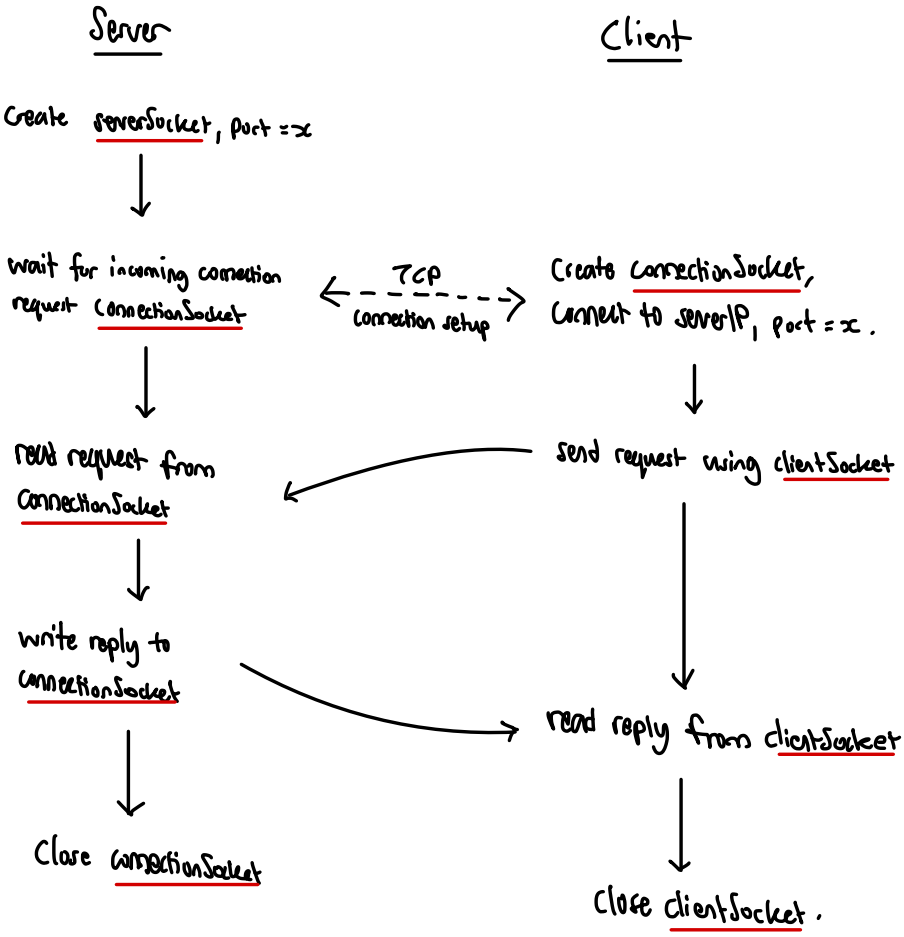
Create packet with serverIP and port = x;
send packet via clientSocket.

read datagram from clientSocket

Close clientSocket.

- Socket Programming with TCP
  - When client creates socket, client TCP establishes a connection to server TCP.
  - When contacted by client, server TCP creates a new socket for server process to communicate with that client. allows server to talk with multiple client individually.





Example: TCP Echo Server

```
from socket import *

serverPort = 2105

serverSocket = socket(AF_INET, SOCK_STREAM)

serverSocket.bind(('', serverPort))

serverSocket.listen()
print('Server is ready to receive message')

connectionSocket, clientAddr = serverSocket.accept()
message = connectionSocket.recv(2048)

connectionSocket.send(message)

connectionSocket.close()
```

TCP socket

listens for incoming TCP request (not available in UDP socket)

returns a new socket to communicate with client socket

Example: TCP Echo Client

```
from socket import *

serverName = 'localhost'
serverPort = 2105

clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))

message = input('Enter a message: ')

clientSocket.send(message.encode())

receivedMsg = clientSocket.recv(2048)

print('from server:', receivedMsg.decode())

clientSocket.close()
```

establish a connection

no need to attach server name, port

- TCP Socket vs UDP Socket
  - In TCP, 2 processes communicate as if there is a pipe between them.
    The pipe remains in place until one of the 2 processes close it.
    - When one of the processes want to send more bytes to the other process, it simply writes data to that pipe.
    - The sending process doesn't need to attach a destination IP address and port number to the bytes in each sending attempt as the logical pipe has been established (which is also reliable).
  - In UDP, programmers need to form UDP datagram packets explicitly and attach destination IP address / port number to every packet.

- Summary
  - TCP socket
    - When contacted by a server, server TCP creates new socket
    - Server use (Client IP + port no.) to distinguish clients.
    - When client creates its socket, client TCP establishes connection to server TCP.

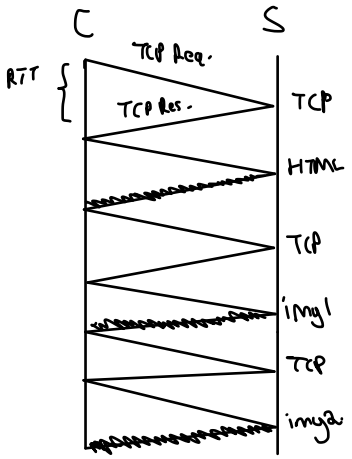
  UDP socket
    - Server uses one socket to serve all clients
    - No connection is established before sending data.
    - Sender explicitly attaches destination IP address and port number to each packet.
    - Transmitted data may be lost or received out-of-order.

Qns - App layer

Q1

A Web server stores a webpage that comprises a base HTML file and 2 images referenced by the base HTML file. The HTML file is 10 bytes and each image is 20 bytes. A client is connected to the Web server through a direct link of 100 bps. RTT is 100 milliseconds.

Suppose HTTP header is of negligible size. How long does the client take to download the entire webpage, assuming non-persistent and non-parallel HTTP is engaged?



$t_{HTML} = 0.1 + 0.1 + \frac{10 \times 8}{100}$

$= 1s$

$t_{img1} = 0.1 + 0.1 + \frac{20 \times 8}{100}$

$= 1.8s$

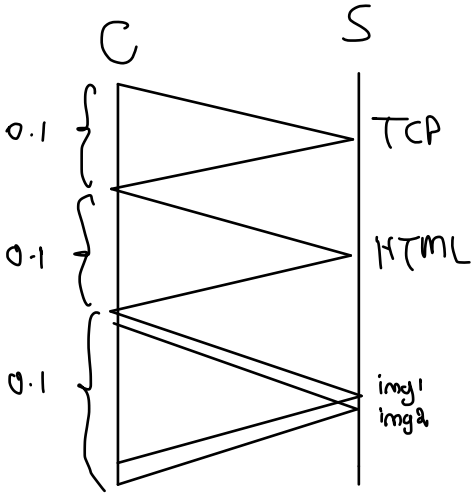
$t_{img2} = 1.8s$

$\therefore t_{Total} = 4.6s$

Q2

A Web server stores a webpage that comprises a base HTML file and 2 images referenced by the base HTML file. The HTML file is 10 bytes and each image is 20 bytes. A client is connected to the Web server through a direct link of 100 bps. RTT is 100 milliseconds.

Suppose HTTP header is of negligible size. How long does the client take to download the entire webpage, assuming persistent HTTP with pipelining is engaged?



$t_{HTML} = 1s$

$t_{img2} = \frac{20 \times 8 + 20 \times 8}{100} + 0.1$

$= 3.3s.$

$t_{Total} = 4.3s.$

1. Which of the following is the Web's application layer protocol?

   A. TCP } tpt layer
   B. UDP
   C. HTTP
   D. DNS - general service for translation.
   E. None of the others

2. Study the following Python code snippet.

```
...
serverSocket.bind(('', 8080))
connectionSocket, clientAddr = serverSocket.accept()
...
```

TCP only

   Suppose no exception is raised when the above code snippet is executed, which port number is **`connectionSocket`** bound to?

   A. TCP port 8080
   B. UDP port 8080
   C. A random port number assigned by server's operation system
   D. It depends on the client's port that's making the connection.
   E. None of the others

3. In a _____ connection, one TCP connection is made for each HTTP request.

   A. persistent
   B. non-persistent
   C. conditional
   D. stateful
   E. None of the others

4. A Web server supports both HTTP/1.0 and HTTP/1.1. So far 120 clients have downloaded a web page from the server, which contains 1 HTML file and 3 images. Half of the clients run HTTP/1.0 and the other half run HTTP/1.1. → TCP

   How many sockets has the Web server ever created?

   A. 120
   B. 121
   C. 300
   D. 301
   E. None of the others

   HTTP1.0 → 60 Client * 4 sockets = 240 sockets
   HTTP1.1 → 60 Client * 1 socket = 60 socket
   240 + 60 + 1 = 301.
   1
   welcome socket.

5. Study the following excerpt of the resource records of NUS's network (TTL field is skipped):

Name	Type	Value
comp.nus.edu.sg	NS	ns1.comp.nus.edu.sg
comp.nus.edu.sg	NS	ns1.nus.edu.sg
comp.nus.edu.sg	MX	postfix0.comp.nus.edu.sg
ns1.comp.nus.edu.sg	A	137.132.90.2
postfix0.comp.nus.edu.sg	A	192.168.21.67
stu.comp.nus.edu.sg	CNAME	stu1.comp.nus.edu.sg
stu1.comp.nus.edu.sg	A	192.168.49.55
ivle.nus.edu.sg	A	137.132.10.10

   If you send an email to Soc.cat@comp.nus.edu.sg, to which IP address will the email be delivered?

   A. 192.168.21.67
   B. 192.168.49.55
   C. 137.132.90.2
   D. 137.132.10.10
   E. None of the others

Consider the command

nslookup www.yahoo.com DNS ⇔ IP addr.

that produces the following output:

```
Server:         137.132.85.2
Address:        137.132.85.2#53

Non-authoritative answer:
Name:   www.example.org
Address: 93.184.216.34
```

What is the IP address of the DNS server that answers query?

- ✓ 137.132.85.2
- 93.184.216.34
- None of the above

Dig and nslookup are two tools that can be used to query DNS servers. They both perform similar functions, but there are some key differences. For example, nslookup can only be used to query one DNS server at a time, while dig can query multiple DNS servers simultaneously. Additionally, dig provides more detailed information about the DNS records that are returned, while nslookup only displays the A and AAAA records.

Dig can show DNS trace tree.

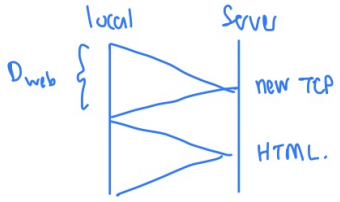
4. **[Modified from KR, Chapter 2, P7]** Suppose within your Web browser, you click on a link to obtain a Web page. The IP address for the associated URL is not cached in your local host, so a DNS lookup is necessary to obtain the IP address.

DNS over UDP ⇒ no need connection unlike TCP

Suppose that ***n*** DNS servers are visited before your host receives the IP address from DNS; visiting them incurs an RTT of D_{DNS} per DNS server. Total = nD_{DNS}

Further suppose that the Web page associated with the link contains ***m*** very small objects (in addition to the HTML page). Suppose the HTTP running is non-persistent and non-parallel. Let D_{Web} denote the RTT between the local host and the server of each object. Total = $2D_{web}(m+1)$

Assuming zero transmission time of each object, how much time elapses from when the client clicks on the link until the client receives all the objects?



Time taken = $nD_{DNS} + 2D_{web}(m+1)$ ✓

12. A Web server stores a webpage that comprises a base HTML file and 5 images referenced by the base HTML file. The HTML file is 200 bytes and each image is 1,000 bytes. A client is connected to the Web server through a direct link of 1 Mbps. RTT is 100 milliseconds.

    Suppose HTTP/TCP headers and control packets are of negligible size; time to close a TCP connection can be omitted. Which of the following correctly calculates the time the client uses a browser to download the webpage from the Web server?

    - ✓ (i) 341.6 milliseconds for persistent HTTP with pipelining
    - ✓ (ii) 741.6 milliseconds for persistent HTTP with no parallel requests (i.e. the next HTTP request is sent after the response for the previous HTTP request is received.)
    - ✓ (iii) 1241.6 milliseconds for non-persistent HTTP with no parallel TCP connections
    - ✓ (iv) 641.6 milliseconds for non-persistent HTTP with maximum 3 parallel TCP connections allowed

    A. (i) only
    B. (i), (ii) and (iii) only
    C. (ii) and (iii) only
    D. (ii), (iii) and (iv) only
    E. (i), (ii), (iii) and (iv) only

i) Handshake 100ms
HTML $100ms + \frac{200\times8}{10^6} = 0.0016$
5 imgs $100ms + \frac{1000\times5\times8}{10^6} = 0.08\times5$
$= 341.6ms.$

ii) HS 100ms
HTML 101.6ms
5 imgs $(100 + \frac{100\times8}{10^6})\times5 = 108\times5ms$

iii) HTML $100 + 100 + 1.6$
5 imgs $(100 + 100 + \frac{1000\times8}{10^6})\times5$

iv) HTML 201.6
3 img $100 + 100 + \frac{3\times1000\times8}{10^6}$
2 img: $100 + 100 + \frac{2\times1000\times8}{10^6}$

Suppose that your department has a local DNS server for all computers in the department. You are an ordinary user (i.e., not a network/system administrator). Can you determine if an external Web site was likely accessed from a computer in your department a couple of seconds ago? Explain.

- **dig –t a www.abc.com**
- **If IP address of this Web page has been queried by another computer seconds ago, your local DNS server should keep this knowledge in local DNS cache and is able to answer your query quickly. Otherwise, the query time will be long.**

A Web server supports both HTTP/1.0 and HTTP/1.1. So far 100 clients have downloaded a web page from the server, which contains 1 HTML file and 2 images. Half of the clients run HTTP/1.0 and the other half run HTTP/1.1.

How many sockets has the Web server ever created?

(A.) 201
B. 200
C. 100
D. 101
E. None of the above

Suppose that a Web server runs in Host C on port 80. Suppose this Web server uses persistent connections, and is currently receiving requests from two different Hosts, A and B. Are all of the requests being sent through the same socket at Host C? If they are being passed through different sockets, do both of these sockets have port 80? Discuss and explain.

For each persistent connection, the Web server creates a separate "connection socket". Each connection socket is identified with a four-tuple: (source IP address, source port number, destination IP address, destination port number). When host C receives an IP datagram, it examines these four fields in the datagram/segment to determine to which socket it should pass the payload of the TCP segment. Thus, the requests from A and B pass through different sockets.

[1 mark] Consider the following Java code snippet.

```
DatagramSocket s = new DatagramSocket(80);
byte[] buf = new byte[1000];
DatagramPacket pkt = new DatagramPacket(buf, buf.length);
s.receive(pkt);
```

UDP

Which of the following statement is TRUE?

A. `s` can be used for both sending and receiving TCP segments.
B. The code snippet will throw an exception at runtime, if port 80 is already in use.
C. A packet received must contain exactly 1000 bytes of data.
D. A packet received must contain at most 1000 bytes of data, inclusive of the size of UDP header.
E. HTTP requests can now be received through `s`.

Transport Layer

- Transport layer services
  - Deliver messages between application processes running on different hosts
    - 2 popular protocols: TCP and UDP.
  - Transport layer protocols run in hosts.

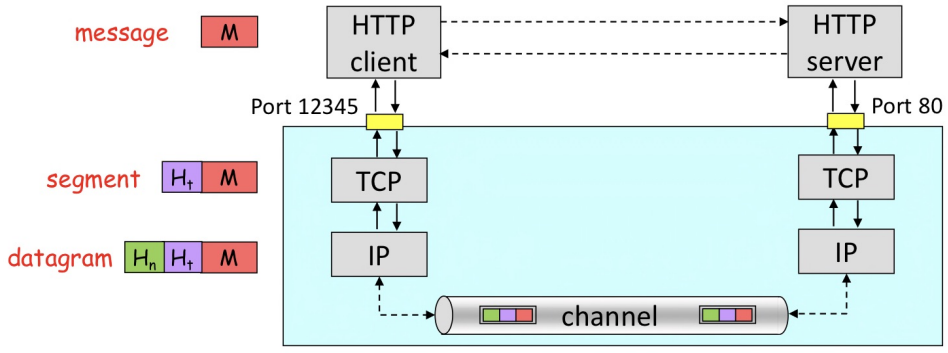
    Sender side: breaks app message into segments (as needed), passes them to network layer (aka IP layer)

  - Receiver side: reassembles segments into message, passes it to app layer.

    Packet switches (routers) in between: only check destination IP address to decide routing.

- Transport / Network layers

  Each IP datagram contains source and dest IP addresses.
  - Receiving host is identified by dest IP address
  - Each IP datagram carries one transport-layer segment.
  - Each segment contains source and dest port numbers.



UDP

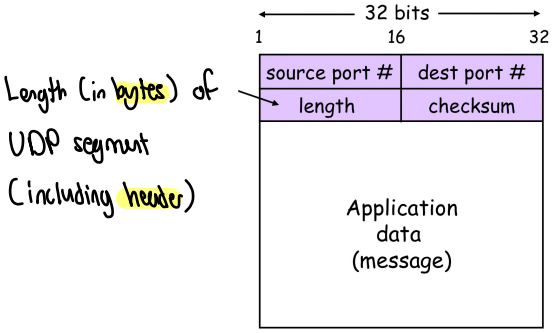
- UDP: User Datagram Protocol (Connectionless transport)
  - UDP adds very little service on top of IP:
    - Multiplexing at Sender:
      - UDP gathers data from processes, forms packets and passes them to IP.
    - De-multiplexing at receiver:
      - UDP receiver passes from lower layer and dispatches them to the right processes.
    - Checksum

  UDP transmission is unreliable.
  - Often used by streaming multimedia apps (loss tolerant & rate sensitive)

Connectionless De-multiplexing

- When UDP receiver receives a UDP segment:
  1. Checks dest. port no. in segment.
  2. Directs UDP segment to the socket with that port number.
  3. IP datagrams (from different sources) with the same destination port number will be directed to the same UDP socket at the destination.

- UDP header



UDP segment format

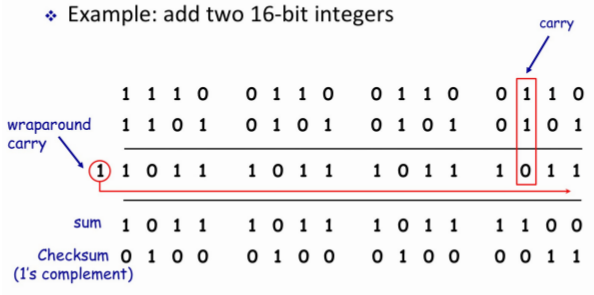
- UDP advantages:
  - No connection establishment (which can add delay)
  - Simple; no connection state at sender, receiver.
  - Small header size.
  - No congestion control:
    UDP can blast away as fast as desired.

- UDP checksum
  - Goal: To detect "errors" (i.e. flipped bits) in transmitted segment.
  - Sender: Compute checksum value,
    put checksum value into checksum field.
  - Receiver: Compute checksum of received segment
    Check if computed checksum equals checksum field value
    - No - error detected
    - Yes - may still have error.

- How is UDP checksum computed?
  1. Treat UDP segment as a sequence of 16-bit integers.
  2. Apply binary addition on every 16-bit integer (checksum field is currently 0).
  3. Carry (if any) from the most significant bit will be added to the result.
  4. Compute 1's complement to get UDP checksum.

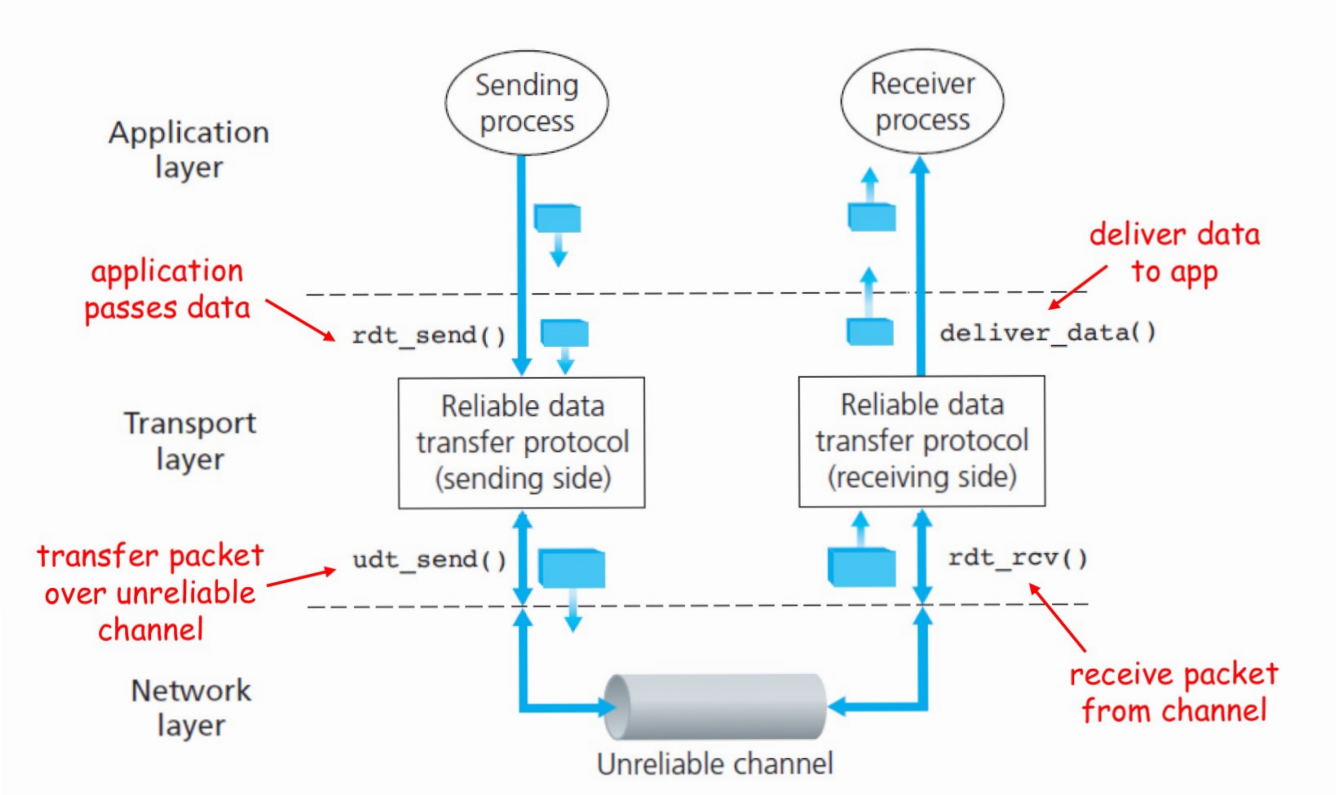
x	y	x ⊕ y	carry
0	0	0	-
0	1	1	-
1	0	1	-
1	1	0	1

- Example: add two 16-bit integers



Principles of Reliable Data Transfer

- Transport vs Network Layer
  - Transport layer resides on end hosts and provide process-to-process communication.
  - Network layer provides host-to-host, best-effort and unreliable communication.
  - ∴ Build a reliable transport protocol on top of unreliable communication.
- Reliable transfer over unreliable channel
  - Underlying network may:
    1. corrupt packets
    2. drop packets
    3. re-order packets
    4. deliver packet after an arbitrarily long delay.
  - End-to-end reliable transport service should:
    1. guarantee packets delivery and correctness.
    2. deliver packets (to receiver application) in the same order they are sent.



Reliable Data Transfer (RDT) Protocols

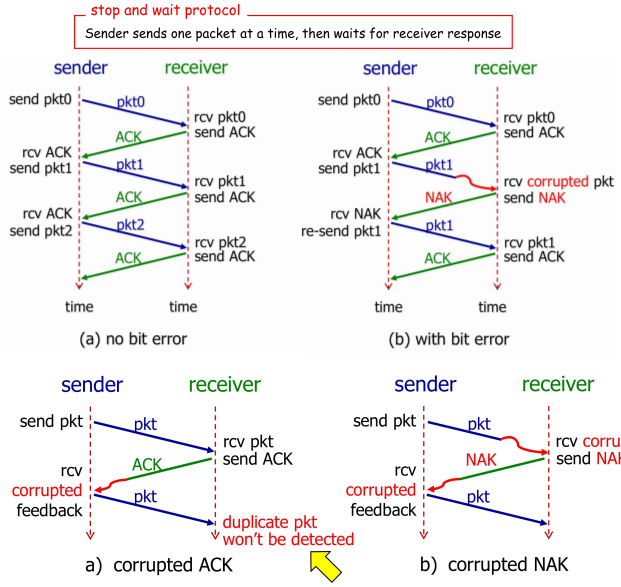
- Characteristics of unreliable channel will determine the complexity of reliable data transfer protocols (rdt).
- Considering only unidirectional data transfer.

rdt 1.0

- Assume underlying channel is perfectly reliable

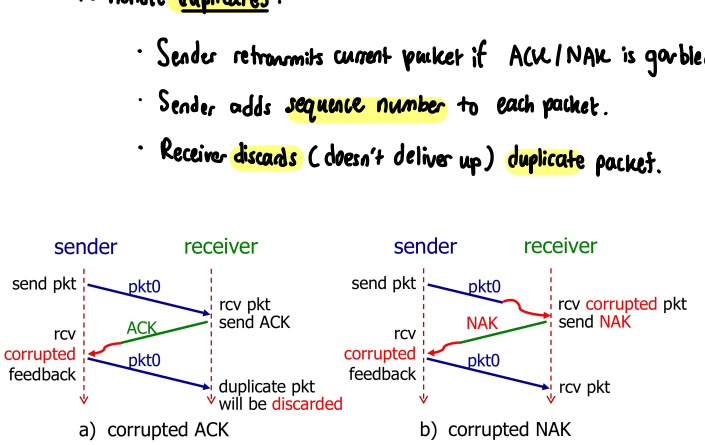
rdt 2.0 (Bit Errors)

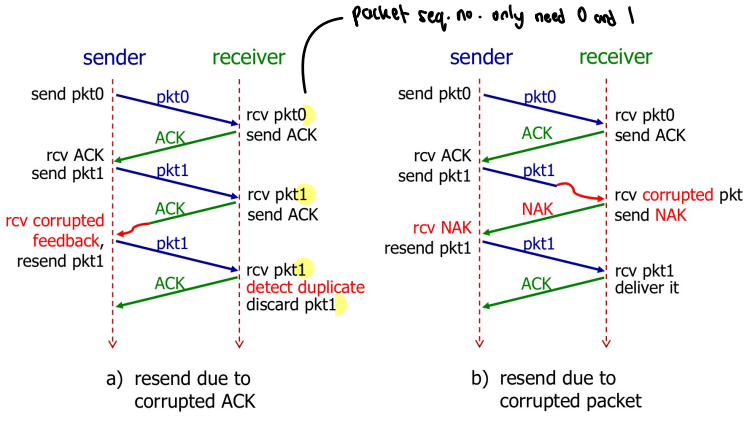
- Assume underlying channel may flip bits in packets
- Receiver use checksum to detect bit error.
- Recover from bit errors.
  - Acknowledgement (ACK):
    receiver explicitly tells sender that packet received is OK.
  - Negative Acknowledgement (NAK):
    receiver explicitly tells sender that packet has errors.
- Flaws:
  - NAK/ACK Corrupted – sender doesn't know what happens at receiver.
  - Sender just retransmit when received garbled feedback
    - ↳ will cause retransmission of correctly received packet
    - ↳ receiver detects duplicate packet ? → rdt 2.1.



rdt 2.1

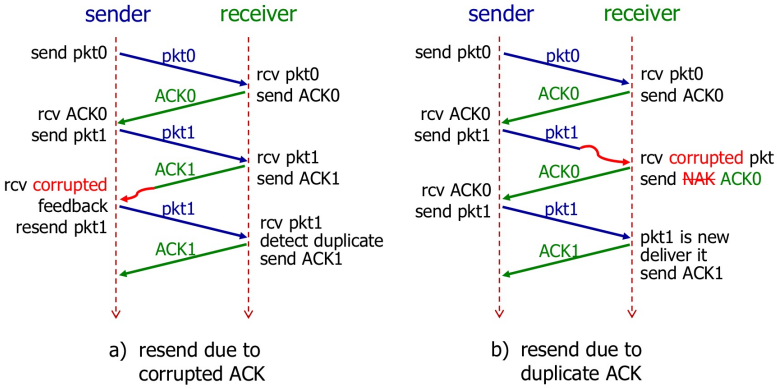
- rdt 2.1 = rdt 2.0 (ACK/NAK) + packet sequence number.
- To handle duplicates:
  - Sender retransmits current packet if ACK/NAK is garbled.
  - Sender adds sequence number to each packet.
  - Receiver discards (doesn't deliver up) duplicate packet.





rdt 2.2 (NAK-free protocol)

- Same function and assumption as 2.1, but use ACKs only.
- Instead of sending NAK, receiver sends ACK for the last package received OK.
  - Now receiver must explicitly include seq. no. of the packet being ACKed.
- Duplicate ACKs at sender results in same action as NAK: retransmit current packet.



rdt 3.0 (Channel with Errors and Loss)

- To handle packet loss:
  - Sender waits "reasonable" amount of time for ACK.
  - sender retransmit if no ACK received till timeout.
- What if packet/ACK is delayed, not lost?
  - Timeout will trigger retransmission.
  - Retransmission will generate duplicates in this case, but receiver may use sequence number to detect it.
  - Receiver must specify sequence number of the packet being ACKed
- Performance:
  - very bad.
  - Eg. Packet size = 8000 bits, link rate = 1 Gbps

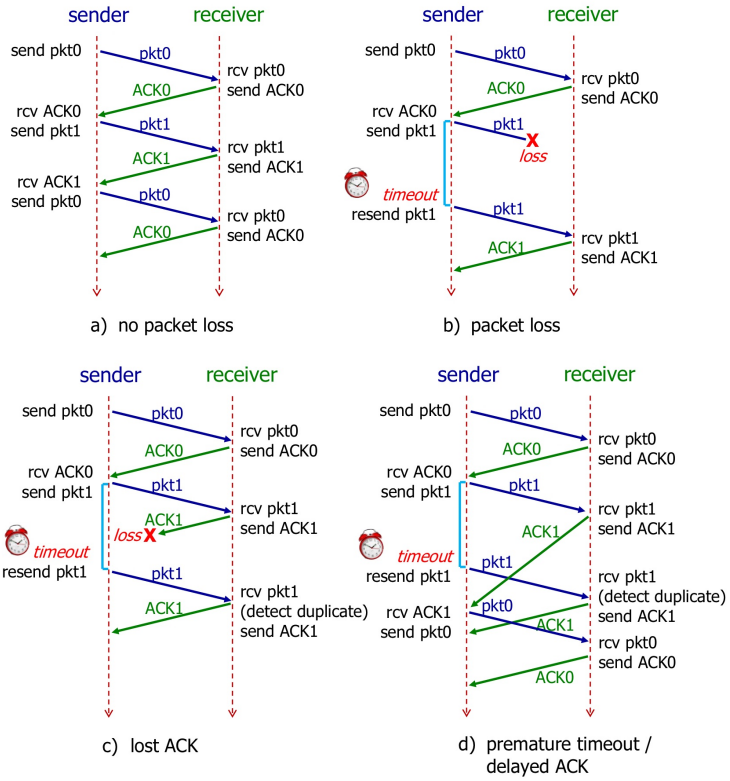
$$d_{trans} = \frac{8000}{10^9} = 0.008 \text{ ms}$$

RTT = 30 ms,
sender sends 8000bits every 30.008 ms,

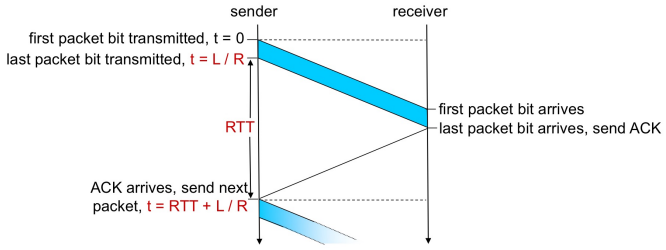
$$\text{throughput} = \frac{L}{RTT + d_{trans}} = \frac{8000}{30.008} = 267 \text{kbps}$$

U_{sender}: Utilisation – fraction of time sender is busy sending

$$U_{sender} = \frac{d_{trans}}{RTT + d_{trans}} = \frac{0.008}{30.008} = 0.00027$$



rdt Version	Scenario	Features Used
1.0	no error	nothing
2.0	data Bit Error	checksum, ACK/NAK
2.1	data Bit Error ACK/NAK Bit Error	checksum, ACK/NAK, sequence Number
2.2	Same as 2.1	NAK free
3.0	data Bit Error ACK Bit Error packet Loss	checksum, ACK, sequence Number, timeout/re-transmission

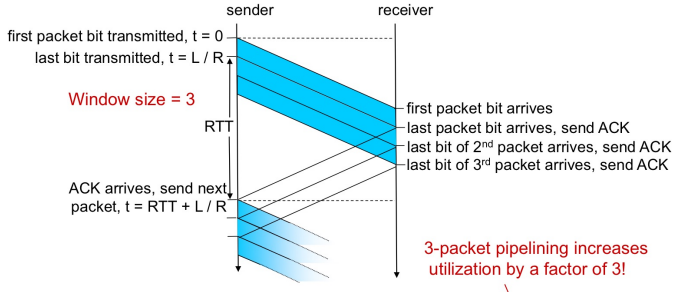


$$U_{sender} = \frac{L / R}{RTT + L/R} = \frac{0.008}{30 + 0.008} = 0.00027$$

⇓

Pipelining: Increased Utilisation → sender allows multiple, "in-flight", yet-to-be-acknowledged packets.

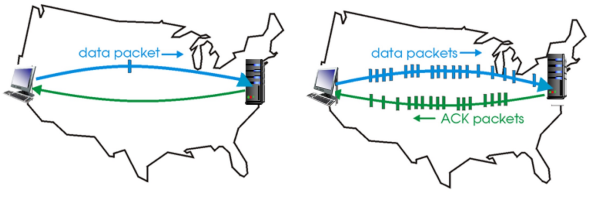
⇓



$$U_{sender} = \frac{3 * L / R}{RTT + L/R} = \frac{0.024}{30 + 0.008} = 0.00081$$

range of seq. no. must be increased

buffering at sender and/or receiver



(a) a stop-and-wait protocol in operation

(b) a pipelined protocol in operation

Pipelined Protocols

- Assumptions (underlying channel)
  1. may flip bits in packets
  2. may lose packets
  3. may incur arbitrarily long packet delay
  4. but won't reorder packets.

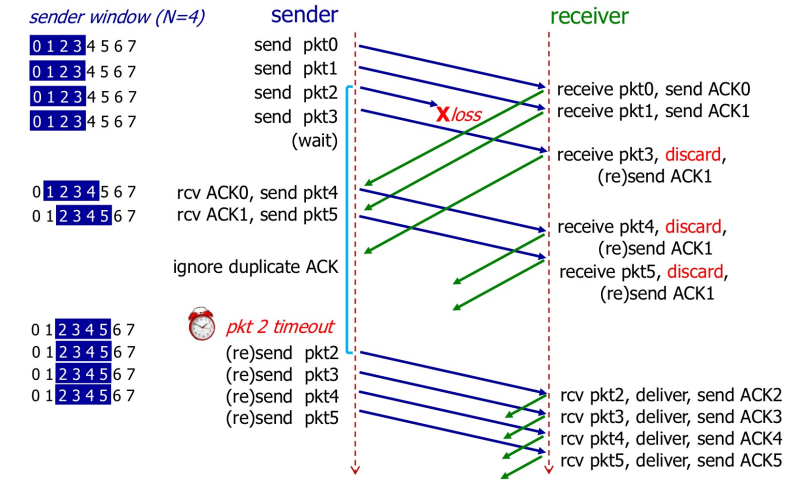
Pipelined Protocols

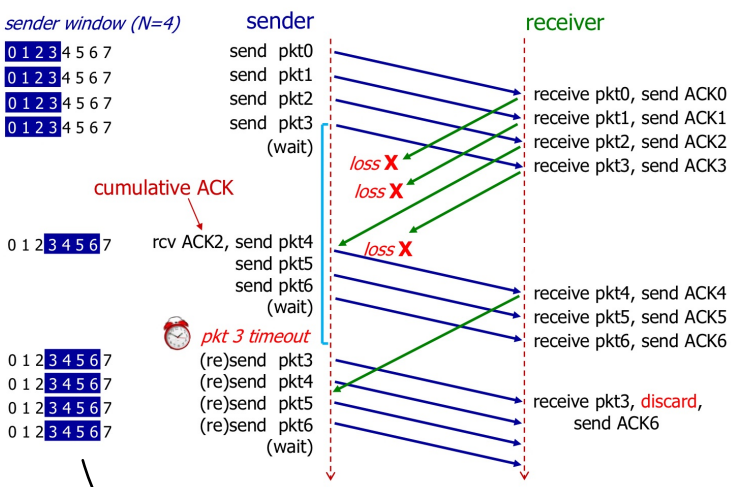
- Same channel assumptions as rdt 3.0
- Aim to increase **utilization** of link
- Range of seq # needs to be increased

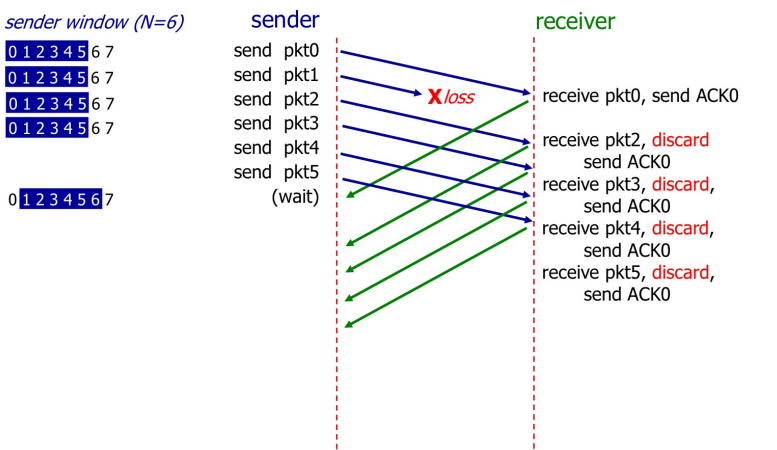
	Go-Back-N	Selective Repeat
#unACK packets	N packets in pipeline	N packets in pipeline
ACK style	cumulative	selective
out-of-order	discarded	buffered
timer	oldest unACK	each unACK
retransmit	all unACK	one unACK

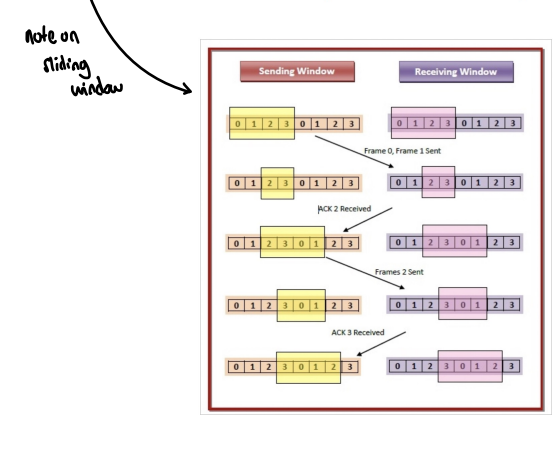
Go-Back-N (GBN)

- GBN Sender
  - Can have up to N unACKed packets in pipeline
  - k-bits sequence number in a packet header (seq. no. resets once max reach).
  - use a "sliding window" to keep track of unACKed packets
  - keep a timer for the oldest unACKed packet
  - On timeout(n): retransmit packet n and all subsequent packets in the window.
- GBN Receiver
  - Only ACK packets that arrive in order.
    - Simple receiver: need only remember expectedSeqNum
  - Discard out-of-order packets and ACK the last in-order sequence number.
    - Cumulative ACK: "ACK m" means all packets up to m are received.









Selective Repeat (SR)

- Receiver individually acknowledges all correctly received packets.
  - Buffers out-of-order packets, as needed, for eventual in-order delivery to the upper layer.
- Sender maintains a timer for each unACKed packet.
  - When timer expires, retransmit only that unACKed packet.

Sender

data from above:
- if next available sequence number in window, send packet.

timeout (n):
- resend packet n, restart timer

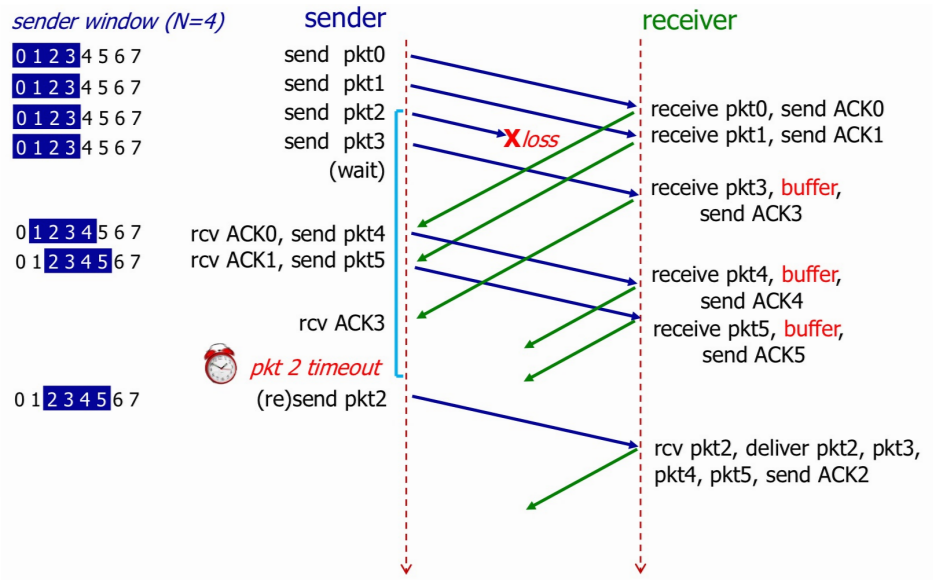
ACK(n) in [sendbase, sendbase+N]:
- mark packet n as received
- if n is smallest unACKed packet, advance window base to next packet seq. no.

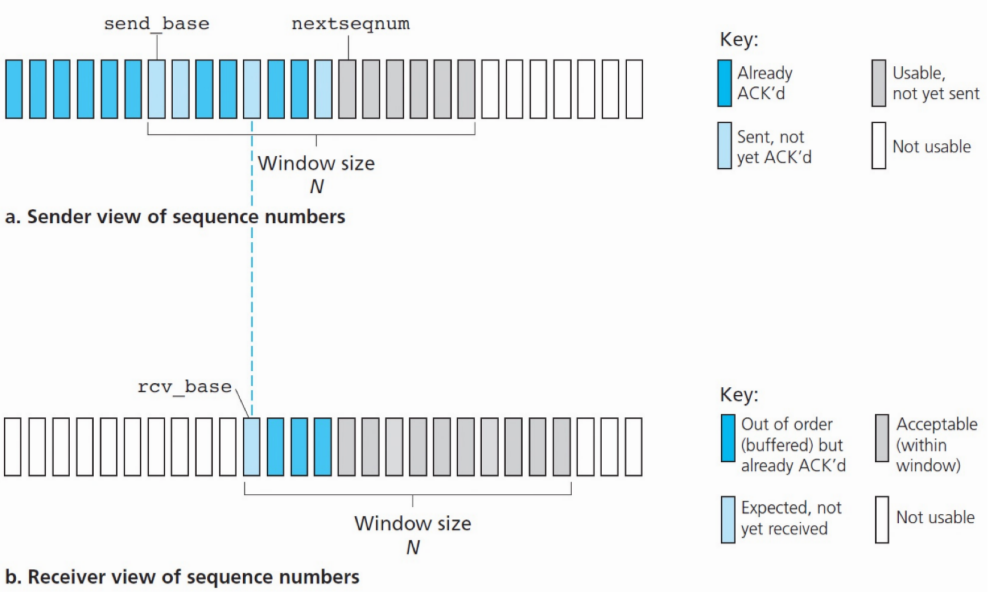
Receiver

Packet n in [rcvbase, rcvbase+N-1]:
- send ACK(n)
- out-of-order: buffer
- in-order:
  deliver (also deliver buffered, in-order packets), advance window to next, not yet received packet.

Packet n in [rcvbase-N, rcvbase-1]:
- ACK(n)

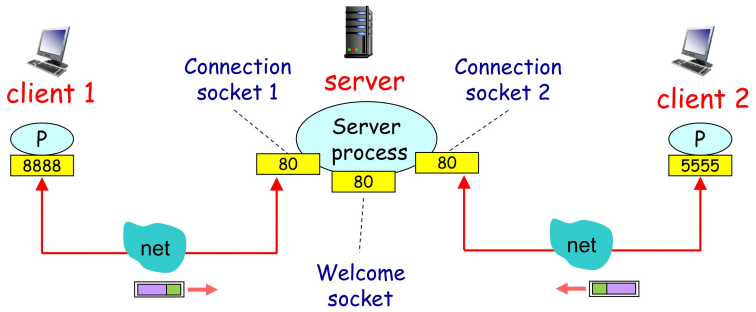
Otherwise:
- ignore.





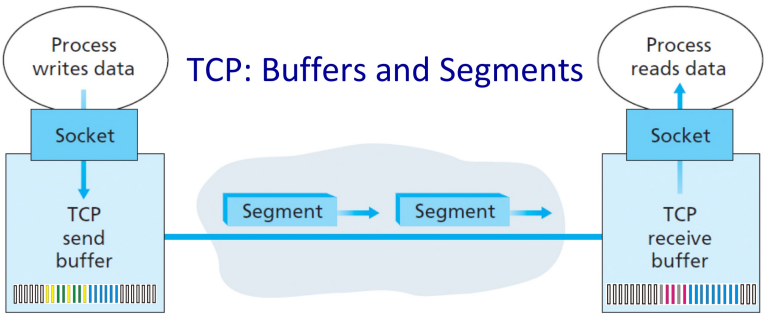
TCP

- TCP: Transmission Control Protocol
- Point-to-Point: one sender, one receiver.
- Connection-Oriented: hand-shaking (exchange of control messages) before sending app data.
  Full duplex service: bi-directional data flow in the same connection.
- Reliable, in order byte stream: use sequence numbers to label bytes.
- A TCP connection (socket) is identified by 4-tuple:
  (srcIPAddr, srcPort, destIPAddr, destPort)
  - Receiver uses all 4 values to direct a segment to the appropriate socket.

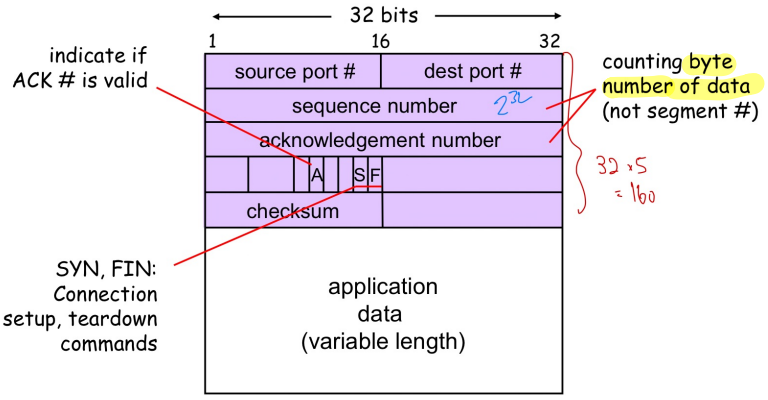


- Buffers and Segments
  - TCP send and receive buffers
    - 2 buffers are created after handshaking at any side.
  - TCP app-layer data
    - Maximum Segment Size (MSS) is typically 1460 bytes.
    - App passes data to TCP and TCP forms packets in view of MSS.

does not include headers.
⇒ only payload.



TCP header

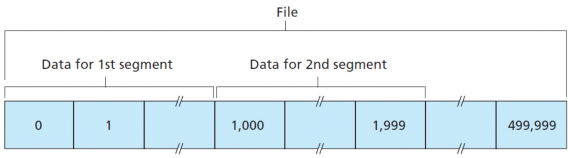


(large seq. no. - 32 bits lowers chance of reuse)

(some fields are not shown)

TCP Sequence numbers

- "Byte number" of the first byte of data in a segment
  - Example: send a file of 500,000 bytes; MSS is 1,000 bytes.



Dividing file data into TCP segments

- Sequence numbers of 1st TCP segment: 0
  2nd TCP segment: 1000
  3rd TCP segment: 2000
  4th TCP segment: 3000
  ⋮

TCP ACK numbers

- Sequence number of the next byte of data expected by the receiver.

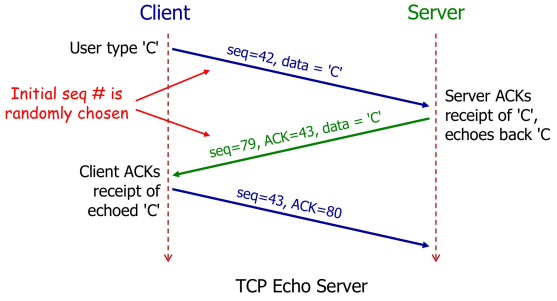
Sequence number of a segment	Amount of data carried	Corresponding ACK number
0	1,000	1,000
1,000	1,000	2,000
2,000	1,000	3,000
3,000	1,000	4,000
...	...	...

i.e. Seq 0 → Ack 1000
Seq 1000 → Ack 2000
Seq 2000 → X
Seq 3000 → Ack 2000.
∴ timeout = 1 retrans.

- TCP ACKs up to the first missing byte in the stream (cumulative ACKs)
  - Note: TCP doesn't say how receiver should handle out-of-order segments; it's up to implementer.

Example: TCP Echo Server

- TCP (UDP) are fully duplex protocols
-



TCP Echo Server

TCP Sender Events (simplified)

```
NextSeqNum=InitialSeqNumber
SendBase=InitialSeqNumber

loop (forever) {
    switch(event)

    event: data received from application above
        create TCP segment with sequence number NextSeqNum
        if (timer currently not running)
            start timer
        pass segment to IP
        NextSeqNum=NextSeqNum+length(data)
        break;

    event: timer timeout
        retransmit not-yet-acknowledged segment with
            smallest sequence number
        start timer
        break;

    event: ACK received, with ACK field value of y
        if (y > SendBase) {
            SendBase=y
            if (there are currently any not-yet-acknowledged segments)
                start timer
        }
        break;

} /* end of loop forever */
```

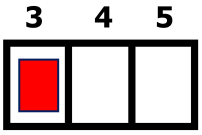
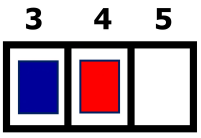
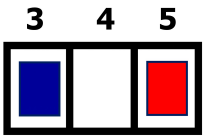
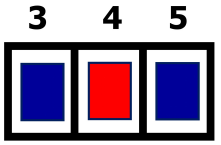
Sender keeps one timer only

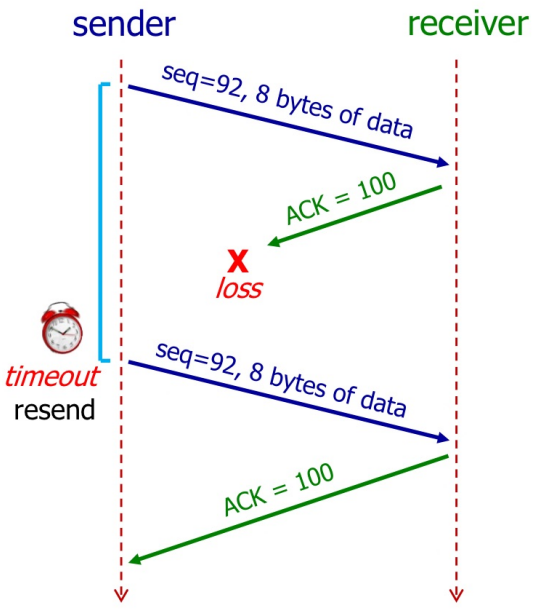
Retransmit only oldest unACKed packet

first byte of data to be ACKed.

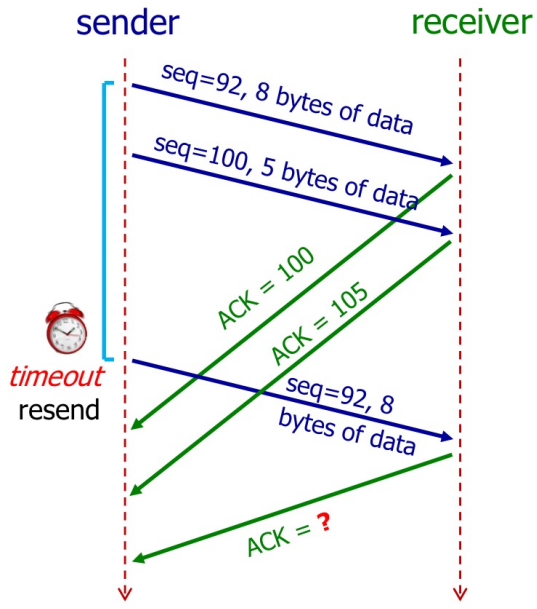
Cumulative ACK

· TCP ACK Generation

Event at TCP Receiver	TCP Receiver action	
1. Arrival of in-order segment with expected sequence number. All data up to expected seq. no. alr. ACKed.	Delayed ACK: wait up to 500ms for next segment. If no next segment, send ACK.	
2. Arrival of in-order segment with expected sequence number. One other segment has ACK pending.	Immediately send single cumulative ACK, ACKing both in-order segments. *	
3. Arrival of out-of-order segment with higher than expected seq. no. (gap detected)	Immediately send duplicate ACK, indicating seq. no. of next expected byte.	
4. Arrival of segment partially or completely fills gap.	Immediately send ACK, provided that segment starts at lower end of gap.	



a) Lost ACK



b) premature timeout

· TCP Timeout Value

- Too Short: premature timeout and unnecessary retransmission
- Too Long: slow reaction to segment loss.
- Timeout interval must be longer than RTT, but RTT varies.
- TCP computes (and keeps updating) timeout interval based on estimated RTT

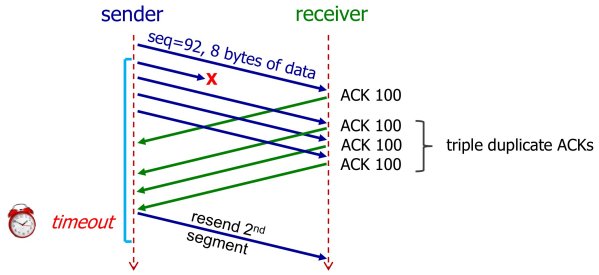
$$RTT_{Est} = (1-\alpha)RTT_{Est} + \alpha RTT_{sample}, \quad \alpha \approx 0.125.$$

$$RTT_{Dev} = (1-\beta)RTT_{Dev} + \beta|RTT_{sample} - RTT_{Est}|, \quad \beta \approx 0.25$$

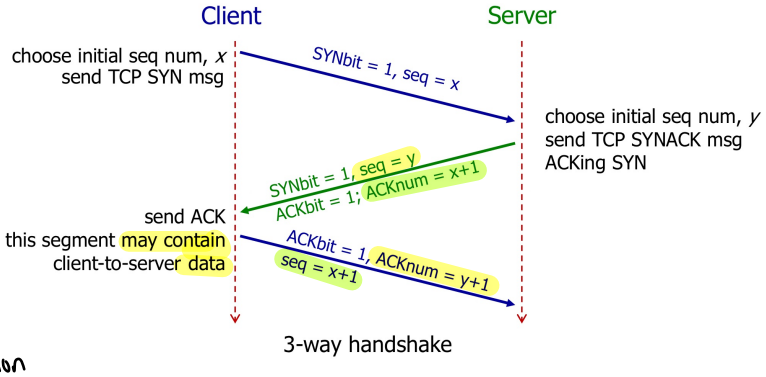
$$\text{Timeout Interval} = RTT_{Est} + 4 * RTT_{Dev}$$

("safety margin".)

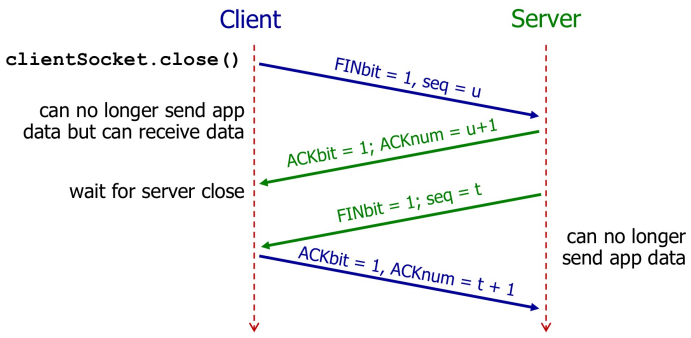
- TCP Fast Retransmission.
  - Timeout period is often relatively long
    - Long delay before resending lost packet.
  - Fast retransmission:
    - Event: If sender receives 4 ACKs for the same segment, (3 duplicate ACKs.)
      it suppose that segment is lost.
    - Action: Resend segment (even before timer expires)



- Establishing Connection
  - Before exchanging data, TCP sender and receiver "shake hands".
    - Agree on connection and exchange connection parameters.



- Closing Connection
  - Client, server each close their side of connection
    - send TCP segment with FIN bit = 1



· Summary

Go-back-N

- Sender can have up to *N* unACKed packets in pipeline
- Receiver only sends *cumulative ACKs*
  - Out-of-order packets discarded
- Sender sets timer for the oldest unACKed packet
  - when timer expires, retransmit ***all*** unACKed packets

Selective Repeat

- Sender can have up to *N* unACKed packets in pipeline
- Receiver sends *individual ACK* for each packet
  - Out-of-order packets buffered
- Sender maintains timer for ***each*** unACKed packet
  - when timer expires, retransmit only that unACKed packet

- Connection-oriented transport: TCP
  - Segment structure
  - Reliable data transfer
  - Sequence number
  - Acknowledgement number
  - Cumulative ACK
  - Setting and updating retransmission time interval
  - Fast retransmission
  - 3-way handshake

We mentioned in lecture that GBN and SR have been designs. TCP takes some design from GBN and some others from SR.

For N outgoing data packets,

-- GBN has only one timer for the oldest outgoing packet

-- SR has N timers, each associated with one outgoing packet

-- TCP has only one timer for the oldest outgoing packet (same as GBN)

Qns-Tpt Layer

Q1

A DNS query is directly encapsulated in a _____ segment.

A. TCP
B. UDP
C. IP

router only IP.

Q2

In the Internet protocol stack, network layer is responsible for _____ communication while transport layer is responsible for _____ communication.

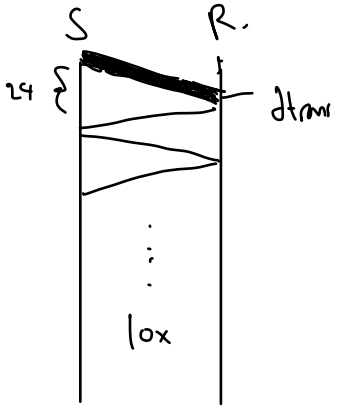
A. host-to-host; process-to-process
B. process-to-process; host-to-host

Q3

Suppose two hosts are connected by a direct link of 1 Mbps. A stop-and-wait protocol is used to transfer 10 packets from the sender to the receiver. Each packet is 1000 bytes long. RTT is 24 milliseconds. Assume that transmission is error-free and ACK packets are of negligible size.

When will the sender receive the last ACK?

A. 32 milliseconds
B. 160 milliseconds
C. 320 milliseconds



$d_{trans} = \frac{8000}{10^6} = 8ms.$

$\therefore$ Total $= 10 \times (8 + 24) = 320ms.$

Q4 *

Suppose we want to design a stop-and-wait, reliable protocol for communication between a sender and a receiver over a channel with the following characteristics: data packets may be lost or corrupted but will not be reordered. Feedback packets are always received in good order. Moreover, the maximum RTT between sender and receiver is known.

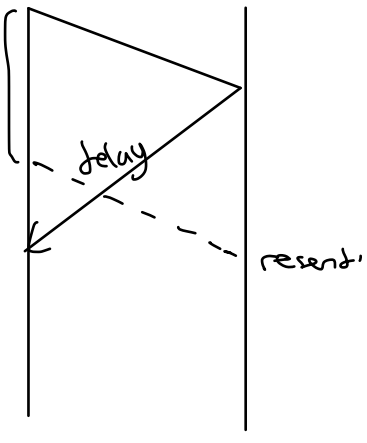
never corrupted

no duplicate.

Which of the following statements about the reliable protocol is TRUE?

A. Sender must attach a sequence number to every data packet.
B. If sender set the timer properly, receiver definitely won't receive duplicate packets. timer > max RTT
C. Receiver should discard corrupted data packet but must acknowledge it.

never the case



will still retransmit.

Q1

1s complement is used as checksum in _____.

A. TCP but not UDP
B. UDP but not TCP
C. Both TCP and UDP

Q2

Which of the following does UDP implement or guarantee?

A. Flow control
B. Congestion control } TCP.
C. Minimum throughput
D. Maximum end-to-end delay } have to reserve links | routers.
E. None of the above

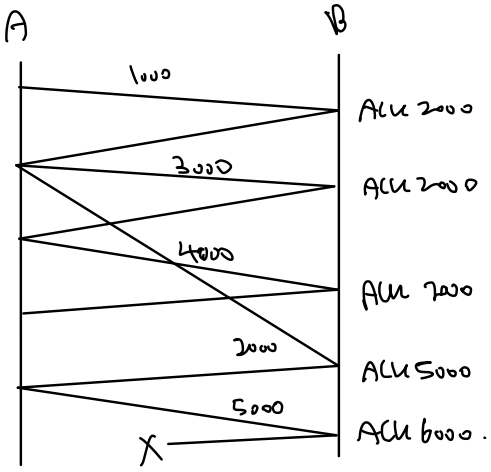
Q3

A TCP sender transmits 5 segments with the sequence numbers 1000, 2000, 3000, 4000 and 5000. After a short while, the sender receives 4 acknowledgements with the ACK numbers 2000, 2000, 2000 and 5000.

What is the sequence number of the TCP segment that is lost?

A. 2000
B. 3000
C. 4000
D. 5000 — no loss

have reordering



Suppose 4 packets are received by a **Go-back-N** receiver. Their sequence numbers (shown in the order of arrival) are 0, 3, 4 and 6 respectively. Packet 0 is the very first packet sent by the **Go-back-N** sender and no packet is corrupted during transmission.
What are the ACK numbers of the first and third feedback packets sent by the receiver?

(1 mark)
You scored 1 / 1 mark

- 0; 0
- 0; 1
- 1; 1

Suppose 4 packets are received by a **Selective Repeat** receiver. Their sequence numbers (shown in the order of arrival) are 0, 3, 4 and 6 respectively. Packet 0 is the very first packet sent by the **Selective Repeat** sender and no packet is corrupted during transmission.
What are the ACK numbers of the first and third feedback packets sent by the receiver?

(1 mark)
You scored 1 / 1 mark

- 0; 1
- 0; 4 ✓
- 1; 5

Which of the following statements regarding TCP is TRUE?

(1 mark)
You scored 1 / 1 mark

- Maximum segment size (MSS) indicates the maximum size of a TCP segment, inclusive of TCP header.
- If a TCP segment has sequence number *m*, then ACK for this segment must have acknowledgement number *m* + 1.
- TCP uses triple duplicate ACKs as a loss indicator to trigger retransmission. ✓

Host A sends one TCP segment to host B. This segment has the sequence numbers 38 and contains 4 bytes of data. Suppose host B has already received all the previous segments. If this segment is also accepted by the receiver, what will be the ACK number in the corresponding ACK segment?

(1 mark)
You scored 1 / 1 mark

- 41
- 42 ✓
- 43

Host A sends two consecutive TCP segments to host B. The first segment has the sequence numbers 38 and the second segment has the sequence number 42. What is the amount of application data in the first TCP segment?

(1 mark)
You scored 1 / 1 mark

- 3
- 4 ✓
- 5

Host A sends two consecutive TCP segments to host B. Each segment contains 10 bytes of data and the first segment has the sequence number 38. Suppose host B has already received all the previous segments. If these two segments are also accepted by the receiver, what will be the ACK number in the second ACK segment?

(1 mark)
You scored 1 / 1 mark

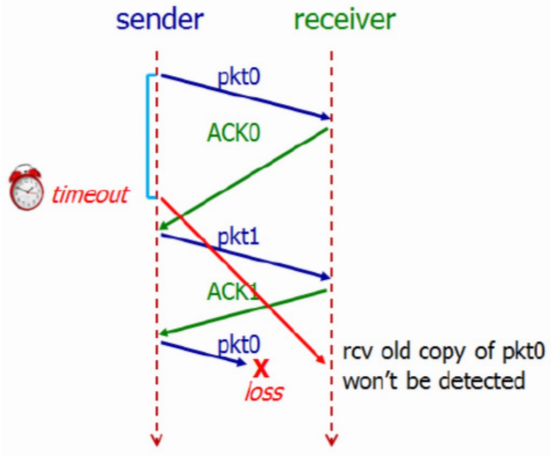
- 48
- 58 ✓
- 68

Refer to the events/examples on lecture 5 notes page 66. Suppose a TCP receiver previously received bytes 3 and 6. Now byte 5 arrives at the receiver. What will be the ACK number in the corresponding ACK segment?

(1 mark)
You scored 0 / 1 mark

- 3 ✗
- 4
- 5

Show an example that if the communication channel between the sender and receiver can reorder messages (i.e., two messages are received in different order they are sent), then protocol **rdt3.0** will not work correctly.



1. Sender send pkt0 and receiver receives it and sends ACK0
2. ACK0 gets delayed due to which pkt0's timeout happens and sender retransmits pkt0.
3. After timeout, ACK0 reaches sender and it sends next packet pkt1 and receiver sends back ACK1.
4. Now receiver is waiting for new pkt0.
5. Sender sends new pkt0 but it get lost and older pkt0 (out of order) reaches receiver and it will be accepted.
6. Receiver cannot differentiate between old pkt0 and new pkt0.

Host A is sending data segments to Host B using a reliable transport protocol (either GBN or SR). Assume timeout values are sufficiently large such that all data segments and their corresponding ACKs can be received (if not lost in the channel) by Host B and the Host A respectively.

Suppose Host A sends **5 data segments** to Host B and **the 2nd data segment is lost.** Further suppose retransmission is always successful. In the end, all 5 data segments have been correctly received by Host B.

How many segments has Host A sent in total and how many ACKs has Host B sent in total if either GBN or SR protocol is used? What are their sequence numbers? Answer this question for both protocols.

GBN: Host A sends 9 segment. They are initially sent segments 1, 2, 3, 4, 5 and later resent segments 2, 3, 4 and 5. Host B sends 8 ACKs. They are 4 ACKs with seq # 1 and 4 ACKs with seq # 2, 3, 4 and 5.

SR: Host A sends 6 segment. They are initially sent segments 1, 2, 3, 4, 5 and later resent segment 2. Host B sends 5 ACKs. They are 4 ACKs with seq # 1, 3, 4, 5 and 1 ACK with seq # 2 (for resent segment).

13. **[2 marks]** Consider a sender and a receiver communicating using **Selective Repeat** protocol. After transmitting for a while, the first and the last sequence numbers in the sender's window are k and $k+3$ respectively. Let a packet with sequence number i be p_i. Which of the following statement MUST be TRUE?

    A. p_k is sent and acknowledged. ✗ windows still at k, possible to be false when P_k not sent.

    B. If p_{k+2} is not sent, p_{k+1} is also not sent. ✗ possible for P_{k+2} and P_{k+1} when ACK is missing.

    C. Receiver is currently expecting p_k. ✗ ACK is on the way back.

    D. If p_{k+3} is sent, it is still unacknowledged. ✗ ACK is on the way back.

    E. None of the above

    Use predicate logic 5 6 7 8 9 10 11 12 13 14 (✓ ✓ ✗ ✓ ✗ ✗)
    ↳ send all tgt
    ↳ send all tgt again.

14. **[2 marks]** A **Go-Back-N** sender just receives an ACK packet with sequence number 14. This ACK number falls within sender's window. Sender's window size is 6. Every packet embeds a k-bit sequence number field. Which of the following definitely CANNOT be the sequence number of the next packet transmitted by the sender?

    A. 9 ✗

    B. 4

    C. 15 ✓

    D. 19 ✓

    E. 20

    6 6
    ~~9 10 11 12 13~~ 14 15 0 1 2 3 4
    anything before
    will still give 14.
    next potentially.
    Since window moves forward.

17. The first segment sent by a sender over TCP has the sequence number 10,000 and the last segment sent has the sequence number 90,000. Assuming that every data segment sent contains exactly 1,000 bytes of data, how many ACK segments could the receiver have **possibly** sent?

    a. 11

    b. 21

    c. 41 → Cumulative ACK.

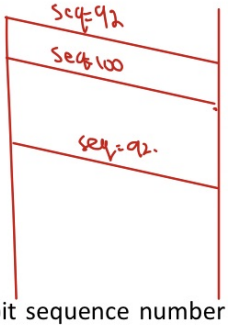
    d. 81

    e. 121 → TCP fast retransmission.

17. Consider the sequence of events recorded by a host in a TCP connection:

→ SEND PKT(seq=92, size=8)
→ SEND PKT(seq=100, size=5)
→ SEND PKT(seq=92, size=8)
← RECV ACK(ack=105)
← RECV ACK(ack=105)

What are the possible events that could have happened?



A. Lost Message
B. Server Failure
C. Premature Timeout
D. Fast Retransmission
E. Lost ACK

18. Consider a Go-Back-N reliable transmission protocol with a k-bit sequence number and sending window of size 3 operating over a channel that can delay, corrupt or lose packets, but not reorder them.

The sender has just sent a new packet with sequence number 2.

Which of the following events could have directly preceded (came before) this (i.e. no other packets were sent or received in between)?

A. Sender received an ACK with sequence number 1
B. Sender received an ACK with sequence number 7
C. Sender sent a packet with sequence number 0
D. Sender sent a packet with sequence number 1
E. Sender sent a packet with sequence number 4

14. A sender and a receiver communicate over an unreliable channel of the following characteristics: **packets may be lost or delayed for arbitrarily long time but won't be corrupted or reordered**.

We would like to design a stop-and-wait, NAK-free protocol called rdt 4.0 to ensure reliable transmission. In rdt 4.0, sender keeps a timer for a sent but unacknowledged packet. When timer expires, sender deems the packet is lost and will retransmit it.

You are to design other features of rdt 4.0 to make it a reliable protocol for the given scenario. The features you design must not conflict with the features/specifications given above.

Which of the following statement about your rdt 4.0 is TRUE?

A. Receiver may receive duplicate packets. However, receiver should simply ignore duplicate packets and do not send ACKs.
B. Receiver must maintain a timer for the ACK sent. If no packet arrives before timer expires, receiver should retransmit the ACK.
C. Sender must resend data packet if a duplicate ACK is received.
D. Sender won't receive duplicate ACKs at all.
E. None of the above

Consider a Selective-Repeat reliable transmission protocol with a k-bit sequence number and sending window of size 3 operating over a channel that can delay, corrupt or lose packets, but not reorder them. The receiver has just sent an ACK before receiving a packet with sequence number 1.

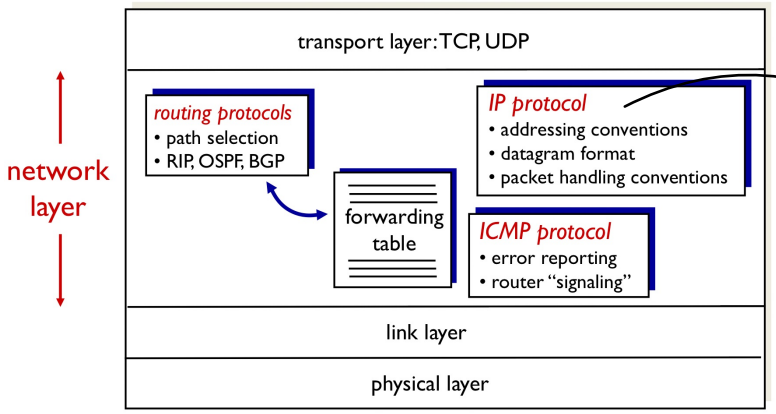
What are the possible sequence numbers for the ACK that was sent?

a. 0
b. 2
c. 3
d. 5
e. 7

Network Layer

- Network layer delivers packets to receiving hosts

  Routers examine header fields of IP datagrams passing it.



IP header specifies TTL (router-hop based) so that packets do not circulate forever.

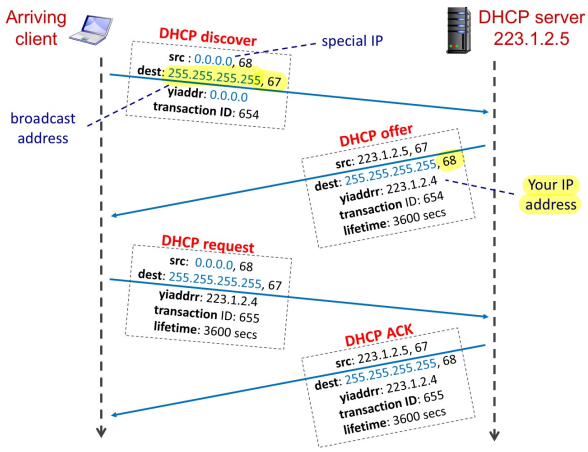
- IP address
  - used to identify a host (or a router)
  - a 32-bit integer expressed in either binary or decimal.
  - Binary: 00000001. 00000010. 00000011. 10000001
    Decimal: 1 2 3 129

    Range –
    Class A -10.0.0.0 to 10.255.255.255
    Class B – 172.16.0.0 to 172.31.255.255
    Class C – 192.168.0.0 – 192.168.255.255
  - Hosts get IP address via:
    1. Manual configuration by system administrator, or
    2. Automatically assigned by a DHCP server.

Special Addresses	Present Use
0.0.0.0/8	Non-routable meta-address for special use
127.0.0.0/8	Loopback address. A datagram sent to an address within this block loops back inside the host. This is ordinarily implemented using only 127.0.0.1/32.
10.0.0.0/8 172.16.0.0/12 192.168.0.0/16	Private addresses, can be used without any coordination with IANA or an Internet registry.
255.255.255.255/32	Broadcast address. All hosts on the same subnet receive a datagram with such a destination address.

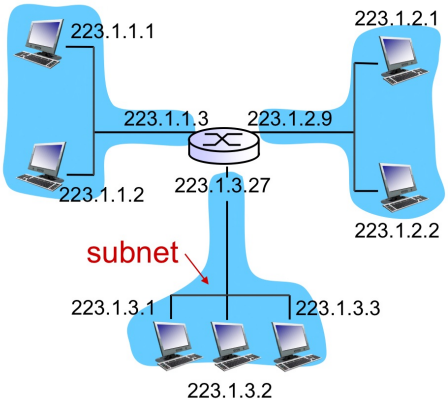
DHCP (Dynamic Host Configuration Protocol)

- DHCP allows a host to dynamically obtain its IP address from DHCP server when it joins network.
  - IP address is renewable
  - Allow reuse of addresses (only hold address while connected)
  - Support mobile users who want to join network.
- DHCP 4-step process:
  1. Host broadcasts "DHCP discover" message
  2. DHCP server responds with "DHCP offer" message
  3. Host requests IP address: "DHCP request" message
  4. DHCP server sends address: "DHCP Ack" message



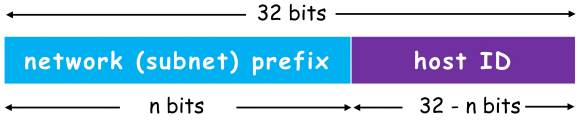
- May provide additional network information:
  - IP address of first-hop router, local DNS server
  - Network Mask (indicates network prefix vs host ID of IP addr.)
- Runs over UDP
  - DHCP server port: 67
  - DHCP client port: 68.

- IP address and Network Interface
  - An IP address is associated with a network interface
    - A host usually has 1 or 2 network interface (eg. Ethernet / WiFi)
    - A router typically has multiple interfaces.



A network consisting of 3 subnets
(first 24 bits of IP addr. are network prefix)

- IP address and Subnet
  - An IP address logically compromises of 2 parts:



  - Subnet is a network formed by a group of "directly" interconnected hosts.
    - Hosts in the same subnet have the same network prefix of IP address.
    - Hosts in the same subnet can physically reach each other without intervening router.
    - Hosts are connected to the outside world through a router.

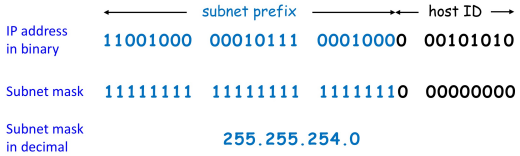
- CIDR (Classless Inter-Domain Routing)
  - The Internet IP assignment strategy
  - Subnet prefix of IP address is of arbitrary length
  - Address format: a.b.c.d/x, where x is the number of bits in the subnet prefix of IP address.

this subnet contains 2^9 IP addresses
subnet prefix: 200.23.16.42/23

/23 indicates the no. of bits of subnet prefix

- Subnet Mask
  - Used to determine which subnet an IP address belongs to.
    - made by setting all subnet prefix bits to "1" and host ID bits to "0"s.

❖ Example: for IP address 200.23.16.42/23:



IP address allocation

- How does an organisation obtain a block of IP addresses?
  - Buy from registry or rent from ISP's address space.

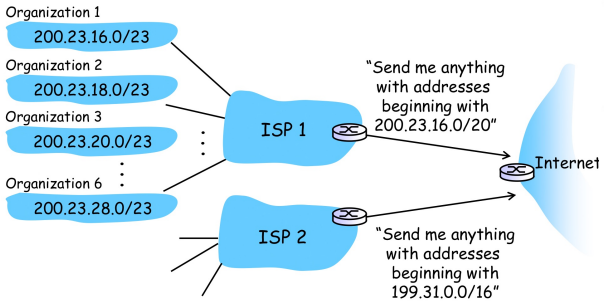
	Binary Address	Decimal Address
ISP's block	11001000 00010111 0001 000 0 00000000	200.23.16.0/20
Organization 1	11001000 00010111 0001 000 0 00000000	200.23.16.0/23
Organization 2	11001000 00010111 0001 001 0 00000000	200.23.18.0/23
Organization 3	11001000 00010111 0001 010 0 00000000	200.23.20.0/23
...	...	...
Organization 6	11001000 00010111 0001 101 0 00000000	200.23.28.0/23

use 3 more bits to differentiate 6 organizations

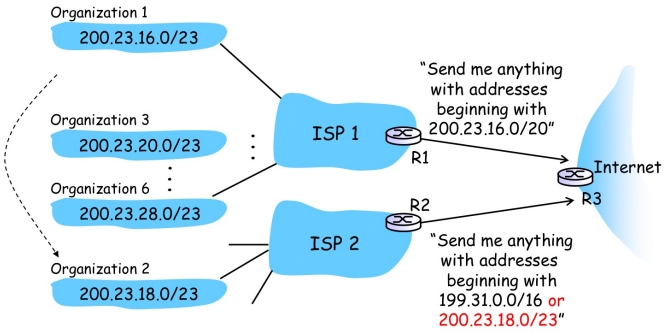
- Q1: How does an organization obtain a block of IP addresses?
- A1: Buy from registry or rent from ISP's address space.
- Q2: How does an ISP get a block of addresses?
- A2: ICANN: Internet Corporation for Assigned Names and Numbers
  - Allocates addresses
  - Manages DNS
  - Assigns domain names, resolves disputes

Hierarchical Addressing

- allows efficient advertisement of routing information.



Suppose Organization 2 now switches to ISP 2, but doesn't want to renumber all of its routers and hosts.

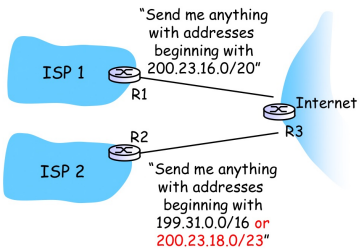


⇓ Longest Prefix Match

- Question: which router to deliver to,
  - if a packet has destination IP 200.23.20.2?
  - if a packet has destination IP 200.23.19.3?

Forwarding Table at R3

Net mask	Next hop
200.23.16.0/20	R1
200.23.18.0/23	R2
199.31.0.0/16	R2
...	...



- Packet with destination IP 200.23.20.2 ⇨ R1
  - (Binary: 11001000 00010111 00010100 00000010)
- Packet with destination IP 200.23.19.3 ⇨ R2
  - (Binary: 11001000 00010111 00010011 00000011)

match the longest prefix

Forwarding Table at R3

Net mask	Net mask in binary	Next hop
200.23.16.0/20	11001000 00010111 00010000 00000000	R1
200.23.18.0/23	11001000 00010111 0001001**0** 00000000	R2
199.31.0.0/16	11000111 00011111 00000000 00000000	R2
...		...

Network Address Translation (NAT)

- Each interface in local network has IP address in reserved range
  - 10.0.0.0 to 10.255.255.255
  - 172.16.0.0 to 172.31.255.255
  - 192.168.0 0 to 192.168.255.255
- Router maintains one external IP address
  - Each local IP/port is mapped to an external port
  - Different hosts on same local network have same external IP address
  - Router needs to store NAT table in memory

Network Address Translation Table	
LAN	WAN
192.168.1.10:4001 ⟺	132.137.83.12:5724
192.168.1.20:4001 ⟺	132.137.83.12:5725

Qns - Network Layer

Q1

Which of the following is a public IP address?

~~A.~~ 10.10.10.10 private
B. 172.10.10.10
~~C.~~ 192.168.10.10 private

Q2

Which of the following is a VALID subnet mask?

~~A.~~ 255.232.0.0
B. 255.240.0.0
~~C.~~ 255.250.255.0

Q3

Which of the following IP addresses belong to the subnet 192.168.0.0/20?

8 8 4

A. 196.168.10.10 → same subnet.
~~B.~~ 196.168.16.10
~~C.~~ 196.168.128.10

Q4

An organization is granted the IP address block 211.80.180.0/24. The administrator wants to create 15 subnets of equal size. How many more bits will be used for the subnet prefix?

A. 3
B. 4
C. 5

0000 → subnet 1
0001 → subnet 2.
⋮
16 subnets

Q5

An IP datagram with destination address 12.10.1.9 reaches a router. Suppose the router engages longest prefix matching and has the following 3 routing entries. Which entry will be used to forward this IP datagram?

~~A.~~ 12.8.0.0/14
B. 12.10.0.0/15 matches
C. 12.10.0.0/20 longest match.