CS2105 Part 1

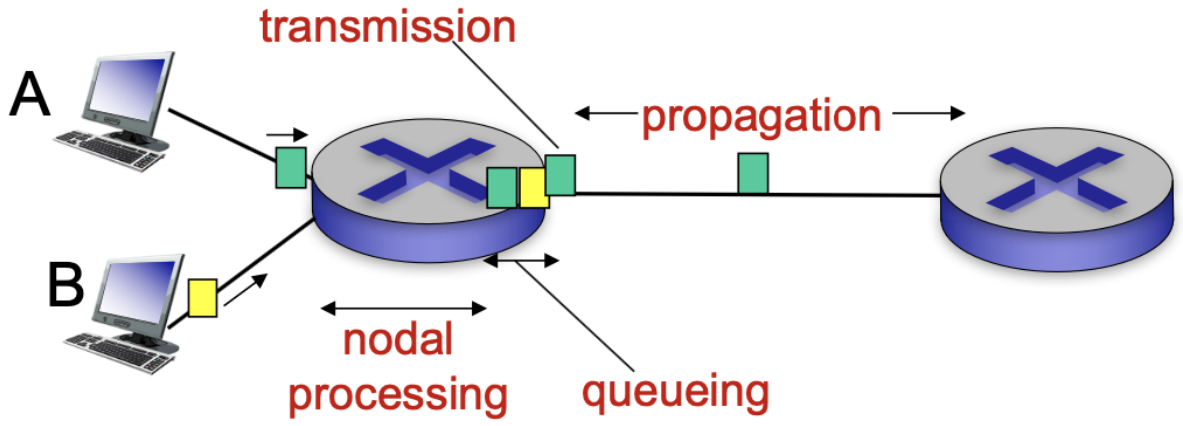
L1

Packet Transmission & Delay

- In packet-switching network, data is first divided into manageable chunks before being sent.

Host (end systems) sending function

- breaks app message into smaller chunks (packets) of length L bits (message segmentation)
    - adv:
        - reduce delay
        - Without message segmentation, if bit errors are not tolerated and there is a single bit error, the whole message has to be retransmitted (rather than a single packet).
        - Without message segmentation, huge packets (containing HD videos, for example) are sent into the network. Routers have to accommodate these huge packets. Smaller packets have to queue behind enormous packets and suffer unfair delays.
    - disadv:
        - Packets have to be put in sequence at the destination (network may re-order packets).
        - Message segmentation results in many smaller packets. Each packet needs to carry packet header of size tens of bytes (e.g. to specific destination address and port number). This is the header overhead of each packet to be discussed in later lectures.
- transmits packet into access network at transmission rate R
    - aka link transmission rate / link capacity / link bandwidth
- Packet-switching - store and forward: entire packet must arrive at router before it can be transmitted on next link
- queuing and loss: arrival rate > transmission rate of link
    - packets will queue and wait to be transmitted in link
    - packets can be dropped (lost) if memory (buffer) fills up



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

d_{proc}: nodal processing

- check bit errors
- determine output link

d_{queue}: queuing delay

- waiting time at output link for transmission

d_{trans}: transmission delay

- time needed to transmit L-bit packets = $L(bits)/R(bits/sec)$

d_{prop}: propagation delay

- d: length of physical link
- s: propagation speed
- $d_{prop} = d/s$
- Round Trip Time (RTT): time for a small packet to travel from client to server and back = 2*prop delay

throughput: rate (bits/time) at which bits are transferred between sender & receiver

Network Core

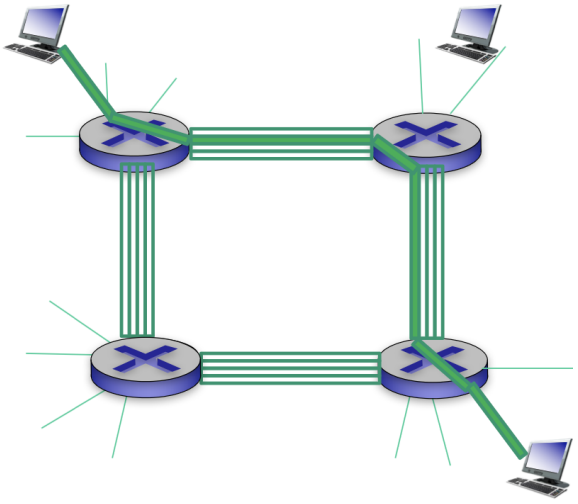
Two key network-core functions

- Routing: determines source destination route taken by packets (routing algo)

- Forwarding: move packets from router's input to appropriate router output

Alternative core: circuit switching

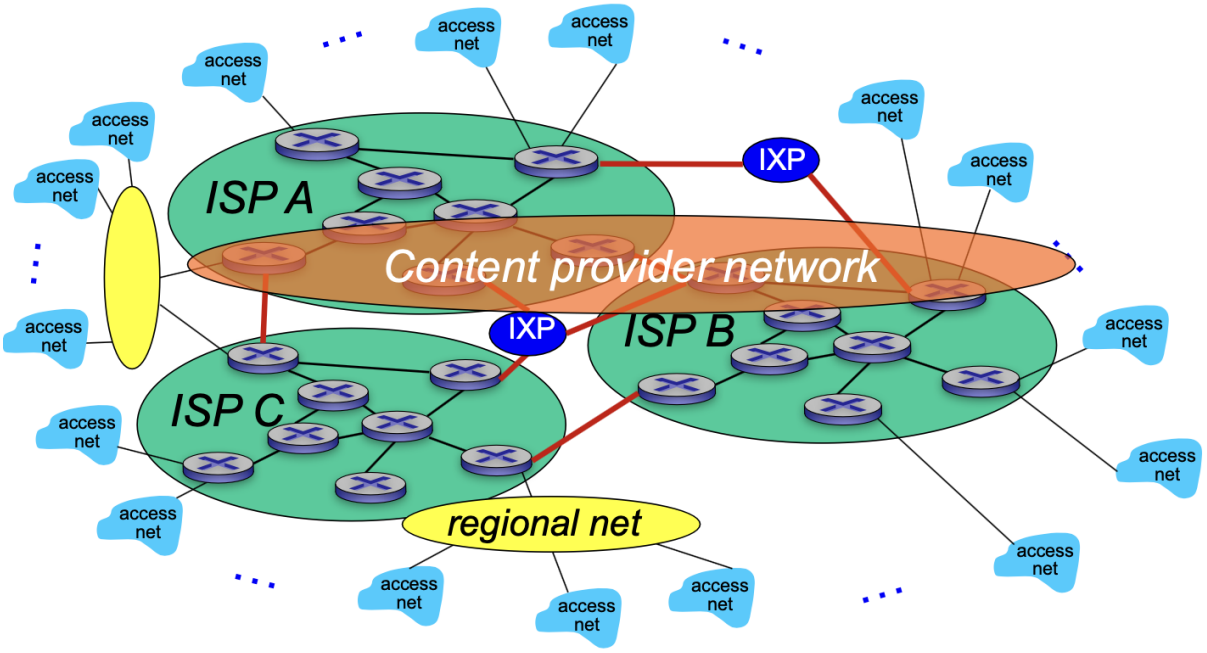
- dedicated resources: no sharing
    - circuit-like (guaranteed) performance
- circuit segment idle if not used by call



call gets 2nd circuit in top link, 1st circuit in right link.

Internet Structure

- end systems connect to Internet via access Internet Service Providers (ISPs)
- ISPs are interconnected
- The Internet is structured as a network of networks



IXP: internet exchange point; content provider network e.g. Google, Microsoft that run their own network to bring services and content close to end users

Min links to connect all devices: N-1

- Simple topology but failure of a single node or link partitions network. Also it tends to have longer paths between 2 nodes.

Max links to connect all devices: N choose 2 = N(N-1)/2

- Most robust topology, 1 hop distance between all nodes, but is most expensive.

Protocols

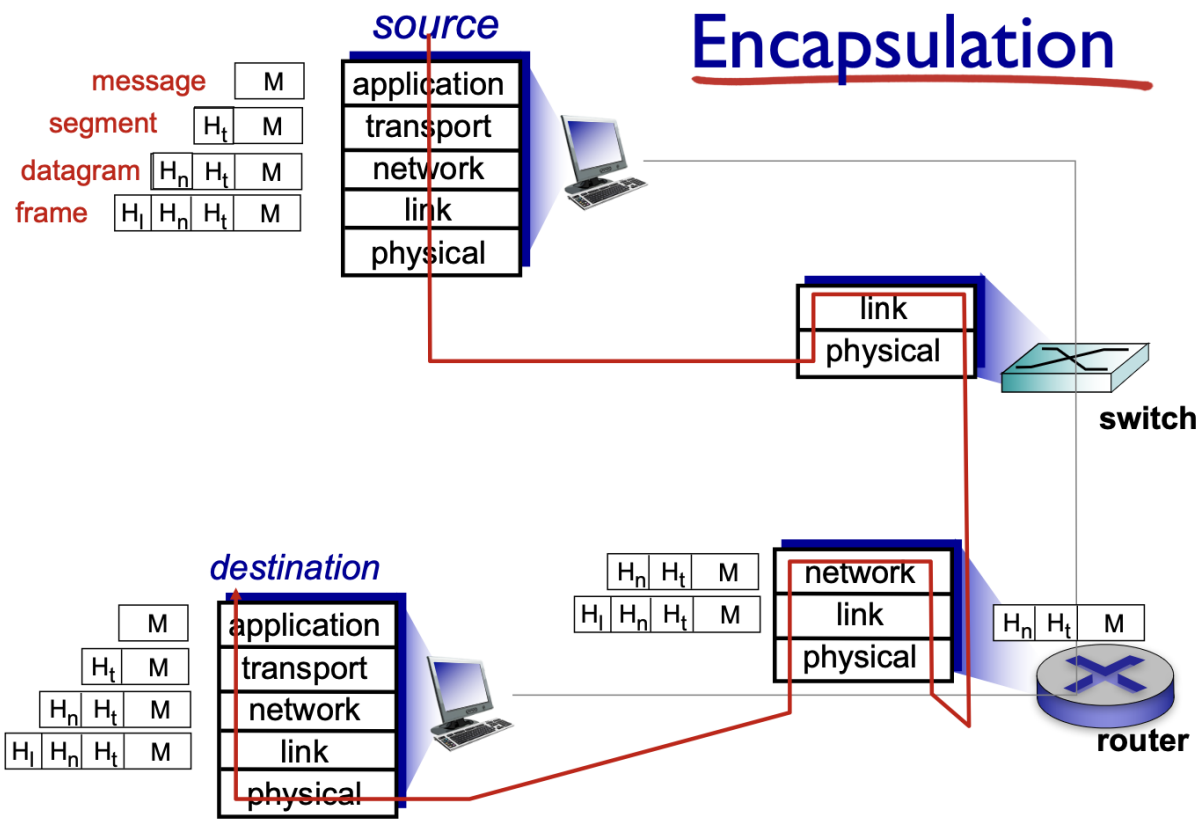
- protocols define format, order of messages sent and received among network entities, and actions taken on message transmission, receipt

Internet Protocol Stack

1. Application: supporting network applications (layer 5)
    - app-lay protocol defines
        - type of messages exchanged: req, res
        - message syntax: fields in messages
        - message semantics: meaning of information in fields
        - rules for when and how processes send and respond to messages

- open protocols: FTP, SMTP, HTTP, DHCP, DNS
- proprietary protocols: RTP (skype)

2. Transport: deliver data from sending process to receiving process (process-process communication)
   - TCP, UDP
3. Network: routing of datagrams from source to destination (host-host communication)
   - IP, routing protocols
4. Link: data transfer between neighbouring network elements
   - Ethernet, 802.111 (WIFI), PPP
5. Physical: bits “on the wire” (layer 1)



L2: Application Layer (message)

Principles of Network Applications

Client-Server Architecture

- Server always on with permanent IP but not client
- client process: initiates communication
- server process: waits to be contacted

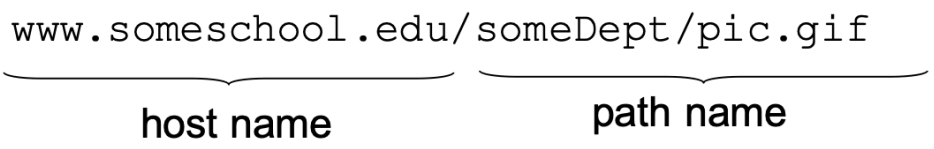
P2P (Peer-Peer) Architecture

- no always-on server
- arbitrary end systems
- peers request service from other peers, provide service in return to other peers
    - self scalability - new peers bring new service capacity & new service demands
- peers are intermittently connected and change IP
- also have server client processes

Addressing processes

- process have identifiers (IP + port numbers) associated with process on host
- transport services need
    - data integrity (data reliability)
    - timing (delay)
    - throughput (how much data can be transferred to the other end)
    - security (encryption)

Web & HTTP (Hypertext Transfer Protocol)



entire thing is URL

- HTTP is stateless: server does not maintain info about past client requests

- A user requests a Web page that consists of some text (1 HTML) and three images: **Download one object per request. Need to send 4 requests to get all objects.**
- HTTP response messages can have an empty message body
- HTTP is not only used to download HTML data from a Web (can download raw binary data also)
- HTTP usually runs over TCP
- The GET method leaves entity body empty when used in the method field, while the HTTP response message leaves out the requested object when HEAD method is used.

Syntax of URL

`scheme://host:port/path?query-string#fragment-id`

- scheme: protocol e.g. http, https, ftp
- hostname: combination of the host's local name with its parent domain's name
  - For example, hostname `www.tutorialrepublic.com` consists of host's machine name `www` and the domain name `tutorialrepublic.com`
- **Port Number** — Servers often deliver more than one type of service, so you must also tell the server what service is being requested. These requests are made by port number.
  - Well-known port numbers for a service are normally omitted from the URL.
  - e.g. HTTP runs by default over port 80, HTTPS runs by default over port 443.
- **Path** — The path identifies the specific resource within the host that the user wants to access. For example, `/html/html-url.php`, `/news/technology/`, etc.
- **Query String** — The query string contains data to be passed to server-side scripts, running on the web server. For example, parameters for a search. The query string preceded by a question mark (`?`), is usually a string of name and value pairs separated by ampersand (`&`), for example, `?first_name=John&last_name=Corner`, `q=mobile+phone`, and so on.
- **Fragment identifier** — The fragment identifier, if present, specifies a location within the page. Browser may scroll to display that part of the page. The fragment identifier introduced by a hash character (`#`) is the optional last part of a URL for a document.

HTTP/1.0 (non-persistent otherwise declared)

- GET: retrieve information from the server.
- POST: requests that a web server accepts and stores the data enclosed in the body of the request message. It is often used when uploading a file or submitting a completed web form.
- HEAD: asks server to leave requested object out of response

HTTP/1.1 (persistent otherwise declared)

- GET, POST, HEAD
- PUT: uploads file in entity body to path specified in URL field
- DELETE: deletes file specified in the URL field

Round Trip Time (RTT): time for a small packet to travel from client to server and back = 2*prop delay

Non-persistent HTTP

- at most one object sent over TCP connection, connection then closed
- downloading multiple objects requires multiple connections
- requires 2 RTT per object

Persistent HTTP

- multiple objects can be sent over single TCP connection between client, server
- two distinct web pages on the same web server (e.g. mit.edu) can be sent over the same persistent connection
- `Connection: keep-alive`
- one RTT for all referenced objects
- need an additional socket connection if the server is not on initially

▼ [Tut 2] 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.

(IP Lookup) 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.

(req/res) Further suppose that the Web page associated with the link contains ***m*** very small objects (in addition to the HTML page). Let ***D_Web*** denote the RTT between the local host and the server of each object.

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?

Suppose the HTTP running is non-persistent and non-parallel.

To map from hostname to IP address: n x D_DNS (note: DNS is over UDP, so no need to establish connection).

To establish TCP connection and get the HTML page = D_Web + D_Web

To establish *m* TCP connection and get all *m* objects = m x (D_Web + D_Web)

Total time = n x D_DNS + (m + 1) x 2 x D_Web

Suppose the HTTP running is non-persistent and parallel.

Total time = n x D_DNS + 2 x D_Web (fetch HTML) + 2 x D_Web (remaining objects can be fetched in parallel)

Suppose the HTTP running is persistent with pipelining.

Total time = n x D_DNS + 2 x D_Web (fetch HTML) + D_Web (remaining objects can be fetched in parallel with one RTT)

1.8 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.

50 50

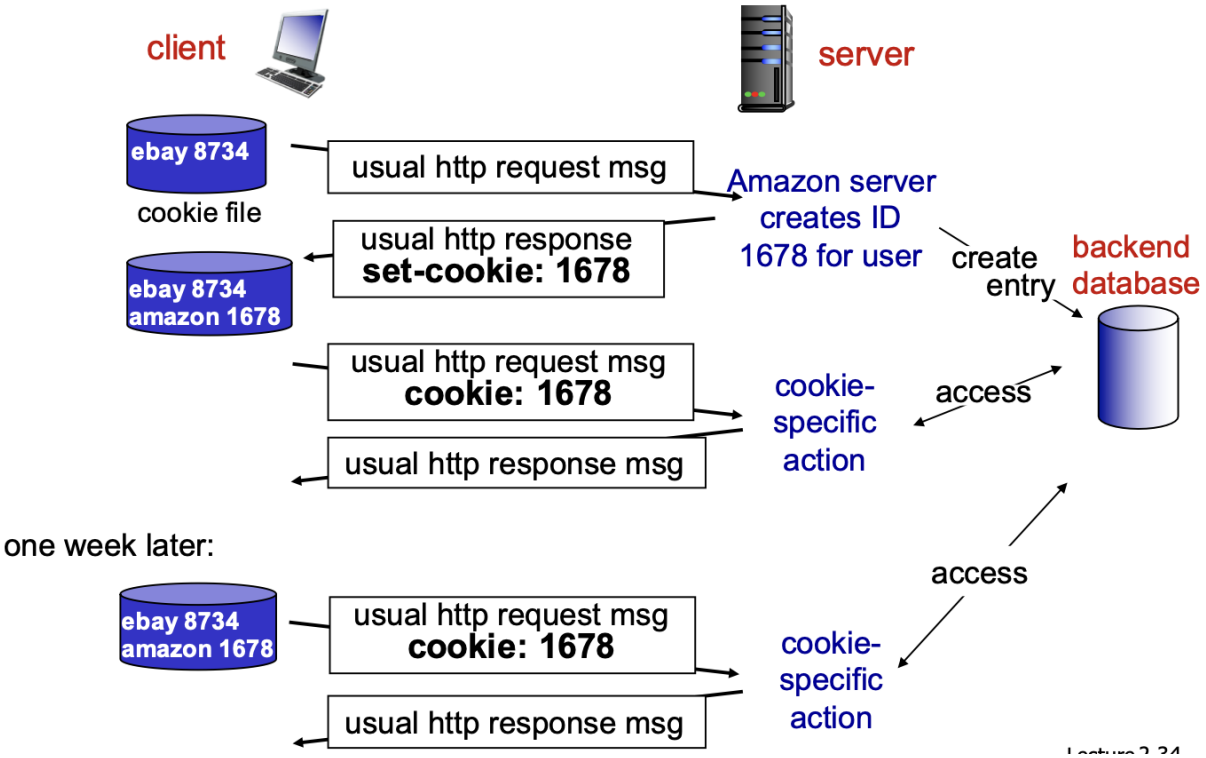
How many sockets has the Web server ever created since it starts running?

A. 201
B. 200 50x3
C. 100
D. 101
E. None of the above

User-Server state: cookies

4 components

1. Cookie header line of HTTP response message
2. Cookie header line in next HTTP request message
3. Cookie file kept on user's host, managed by user's browser
4. Back-end database at Website



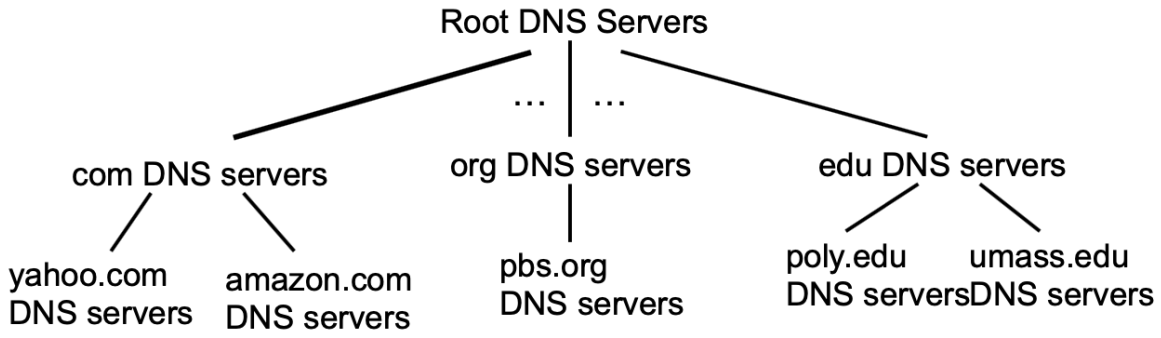
Cache

- goal: satisfy client request without involving origin server
- user sets browser: web accesses via cache (specify date of cached copy)
- browser sends all HTTP requests to cache
  - if object in cache (and updated): cache returns object
  - else: cache requests object from origin server, then returns object to client

Domain Name System (DNS)

- a distributed, hierarchical database

- provides hostname to IP address mapping
- a hostname may be mapped to to multiple IP addresses
- DNS servers typically listen to UDP port 53
- DNS queries do not have to got to the root servers all the time.
- `nslookup` : query DNS to obtain mapping between domain/host name and IP addresses



client wants IP for www.amazon.com; 1st approximation:

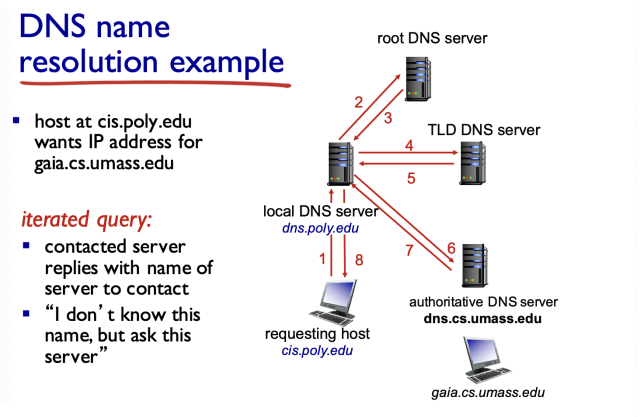
- client queries root server to find com DNS server
- client queries .com DNS server to get amazon.com DNS server
- client queries amazon.com DNS server to get IP address for www.amazon.com

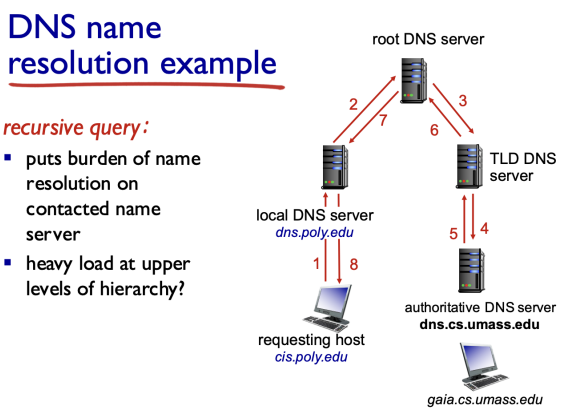
Top-Level Domain (TLD) servers:

- com, org, net, edu...

Authoritative DNS servers:

- organisation's own DNS servers, providing authoritative hostname to IP mappings for organisation's named hosts





▼ How to determine if an external website was likely accessed recently?

```
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 DNS server can respond to a query with respect to:

- Canonical name
- MAC address
- Port number
- Hostname
- IP address

```
dig [server] [name] [type]
```

[server] – the IP address or hostname of the name server to query.

If the server argument is the hostname then dig will resolve the hostname before proceeding with querying the name server.

It is optional and if you don't provide a server argument then dig uses the name server listed in **/etc/resolv.conf**.

[name] – the name of the resource record that is to be looked up.

[type] – the type of query requested by dig. For example, it can be an A record, MX record, SOA record or any other types. By default dig performs a lookup for an A record if no type argument is specified.

L3: Transport Layer (segment)

Socket Programming with UDP and TCP

- both are process-to-process communication
- both uses uses de-multiplexing to dispatch incoming packets to different processes in the same host and multiplexing to send packets

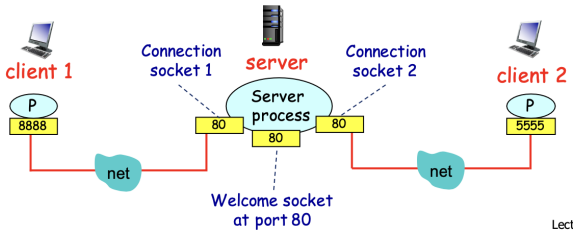
TCP

- reliable *bytestream* transfer between sending and receiving process
- connection-oriented
  - server process must be running first
  - client specifies IP and port number of server process to connect
- flow control: send wont overwhelm receiver
- congestion control: throttle sender where network overloaded
- does not provide:
  - timing,
  - min throughput guarantee
  - security
- adv: ensure files are uploaded/downloaded intact (for FTP)

UDP

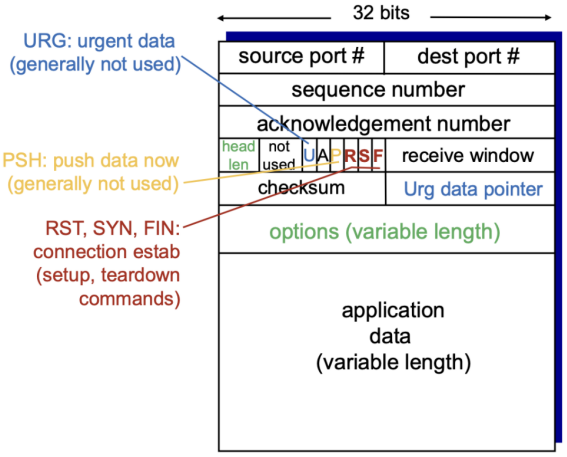
- unreliable *datagram* transfer (lost/out of order)
- no connection setup required between client and server processes
  - sender (client) explicitly attaches destination IP address and port number to every packet
  - receiver (client) extracts sender IP address and port number from the received packet
- does not provide:
  - reliability
  - flow control
  - congestion control
  - timing
  - throughput guarantee
  - security

- identified by the quadruple {src IP, src port, dest IP, dest port}
- a browser can open more than 1 TCP connection at a time

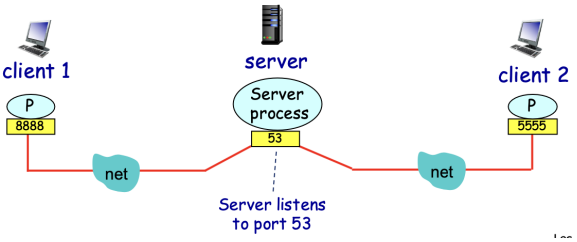


When contacted by client, server TCP creates a new socket for server process to communicate with client; uses connection to identify client

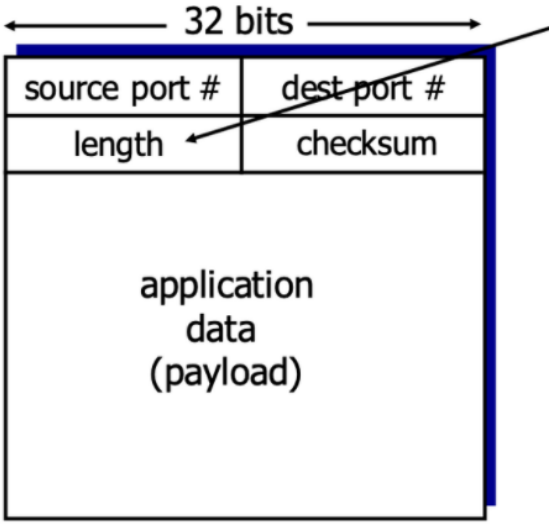
TCP header (min 20 bytes)



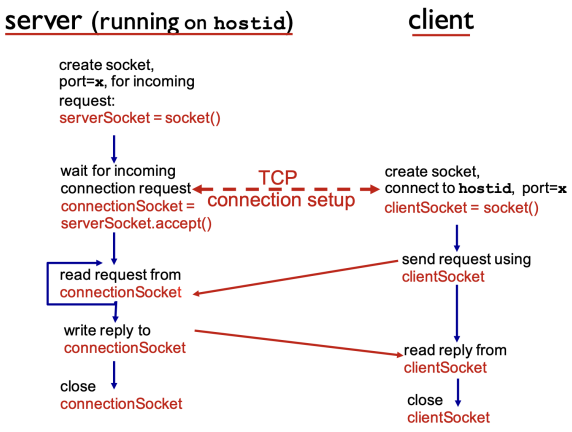
- adv: less overhead and cheaper, fast and relatively stable throughput (for live video streaming)
- identified by tuple {dest IP, dest port}
- When a host received a UDP segment, it
  - checks the dest port num
  - directs the UDP segment to the socket with that port num
  - IP datagram with diff source IP addresses and/or src port num are directed to the same socket. (dest can identify diff hosts with their IP addresses)

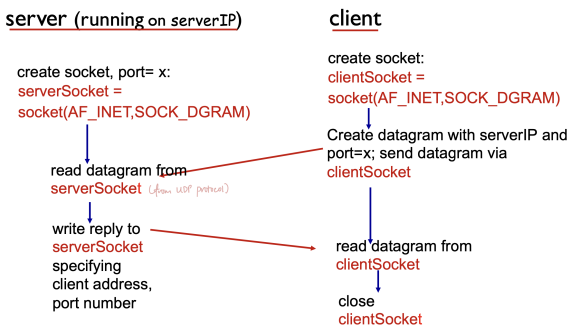


only one socket connection to serve all clients



UDP segment format





Python TCPClient

```
from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET,SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = input('Input lowercase sentence:')
clientSocket.send(sentence.encode())
modifiedSentence = clientSocket.recv(1024)
print('From Server:', modifiedSentence.decode())
clientSocket.close()
```

create TCP socket for server, remote port 12000
Establish a TCP connection with the server process
No need to attach server name, port

Python TCPServer

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
print('The server is ready to receive')
while True:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence.
                          encode())
    connectionSocket.close()
```

create TCP welcoming socket
server begins listening for incoming TCP requests
loop forever
server waits on accept() for incoming requests, new socket created on return
read bytes from socket (but not address as in UDP)
close connection to this client (but *not* welcoming socket)

Python UDPClient

```
from socket import *
serverName = 'hostname'
serverPort = 12000
clientSocket = socket(AF_INET,
                      SOCK_DGRAM)
message = input('Input lowercase sentence:')
clientSocket.sendto(message.encode(),
                    (serverName, serverPort))
modifiedMessage, serverAddress =
                    clientSocket.recvfrom(2048)
print(modifiedMessage.decode())
clientSocket.close()
```

include Python's socket library
create UDP socket for server
get user keyboard input
Attach server name, port to message; send into socket
read reply characters from socket into string
print out received string and close socket

Python UDPServer

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print ("The server is ready to receive")
while True:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.decode().upper()
    serverSocket.sendto(modifiedMessage.encode(),
                        clientAddress)
```

create UDP socket
bind socket to local port number 12000
loop forever
Read from UDP socket into message, getting client's address (client IP and port)
send upper case string back to this client

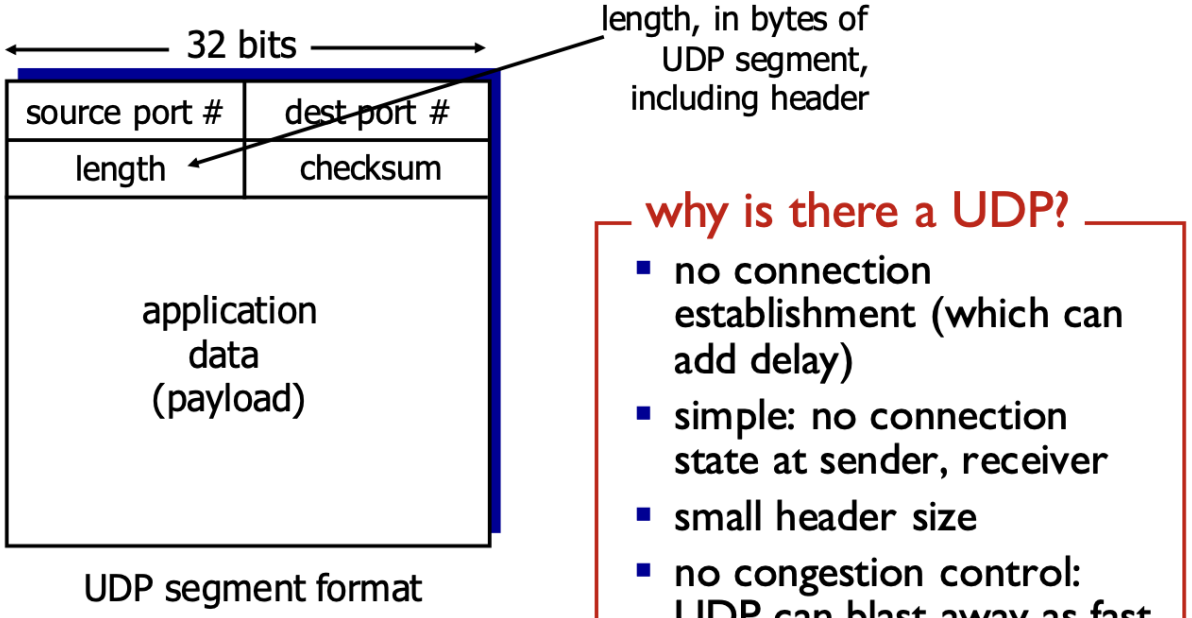
Lecture 3-19

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

L4: UDP & Reliable Data Transfer (RDT)

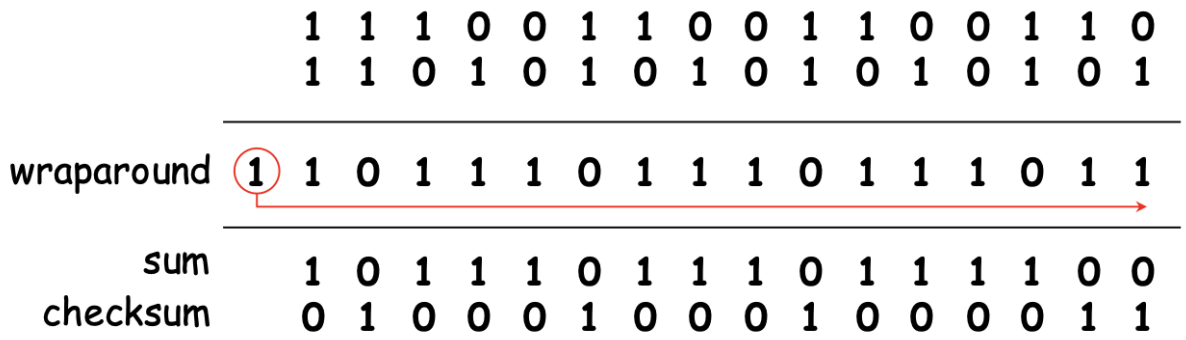
- For an application to enjoy reliable data transfer when the application runs over UDP, one would have to
  - implement reliability checking and recovery mechanisms (ACK, seq #, checksum, timeout, re-transmission, etc.) at application layer.
  - For example, sender needs to include relevant header/trailer fields in every packet (as illustrated below).

Segment UDP header



header is 4 words (each word is 2 bytes) = 8 bytes; if header has no payload ⇒ length = 8 = 0000000000001000 (16 bits)

UDP Checksum (1s complement)



Treat UDP segment as a sequence of 16-bit integers, add carry to result, compute 1s complement (revert all bits)

- Receiver cannot be absolutely certain that no bit errors have occurred even if the checksum is the same
  - e.g. original data: `01011100` and `01100101` **,** Sum: 11000001, checksum: 00111110
  - If sender transmits the following two bytes: 01011100 and 01100101, and the two bits highlighted in red flip, then checksum remains unchanged and receiver will fail to detect this error.
- UDP Checksum is calculated by adding
  - src port number
  - dest port number
  - length
  - data (if no data, no need add)

e.g. Suppose a UDP server process with a port number 40000 sends a UDP segment to a client whose port number is 40001. If the segment does not contain any payload,

- the 16-bit binary value of the length filed is *0000000000001000*
- the 16-bit binary value of the checksum filed is *1100011101110101*

RDT1.0

- perfectly reliable underlying channel with no bit error and loss of packets

RDT2.0

- bit error detection

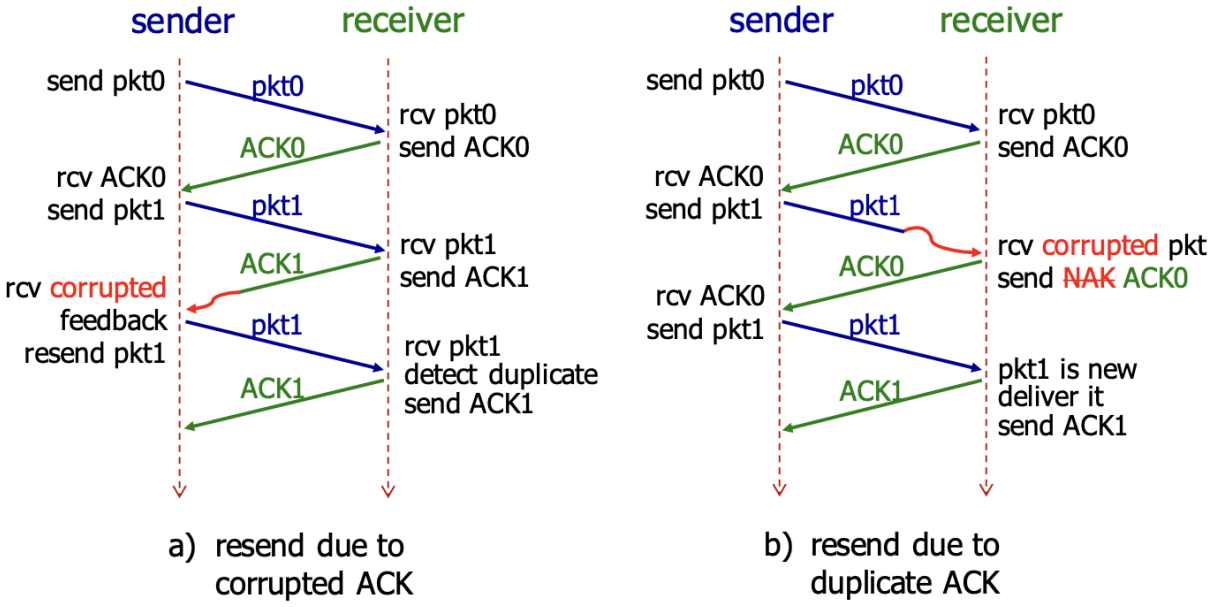
- feedback from receiver to sender
  - acknowledgements (ACKs): received ok
  - negative acknowledgements (NAKs): errors ⇒ retransmit packet

RDT2.1

- ACK/NAK can be corrupted
- sender adds sequence number to each packet
- receiver detects whether an arriving packet contains new data or is a retransmission (discarded)

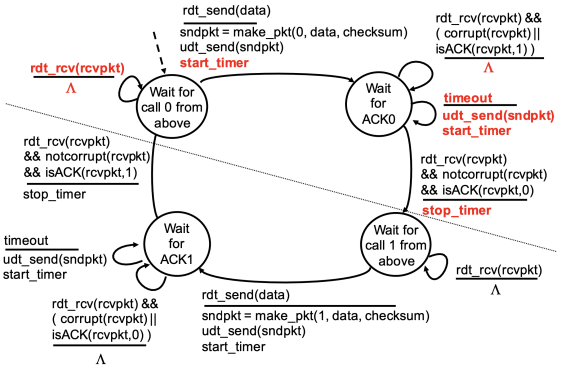
RDT2.2

- receiver sends ACK for last packet that received OK instead of NAK
- receiver explicitly include sequence number of packet being ACKed
- duplicate ACK ⇒ retransmission

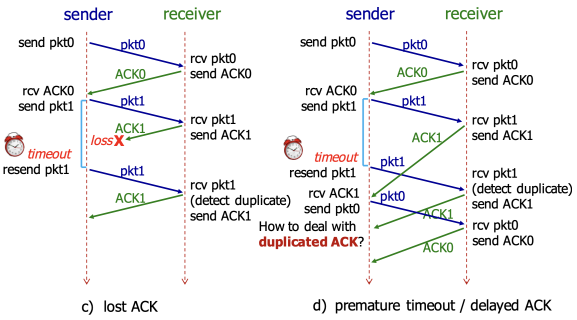


RDT3.0

- underlying channel can lose packets (data, ACKs)
- timer: retransmits if no ACK received
- only 0s and 1s
- if the sender receives a duplicate ACK, it does nothing
- if the receiver receives a duplicate packet, receiver sends ACK for the previous packet.
- if the sender receives a corrupted ACK, it does nothing
- if the receiver receives a corrupted pkt, it sends ACK for the previous pkt



^ means do nth, sender view



- rdt3.0 is correct, but performance stinks
- e.g.: 1 Gbps link, 15 ms prop. delay, 8000 bit packet:

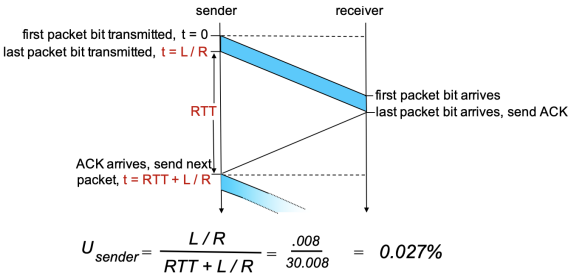
$$D_{trans} = \frac{L}{R} = \frac{8000 \text{ bits}}{10^9 \text{ bits/sec}} = 8 \text{ microsecs}$$

  - U_{sender}: *utilization* – fraction of time sender busy sending

$$U_{sender} = \frac{L/R}{RTT + L/R} = \frac{.008}{30.008} = 0.027\%$$

  - if RTT=30 msec, 1KB pkt every 30 msec: 33kB/sec thruput over 1 Gbps link
- network protocol limits use of physical resources!

utilization equation

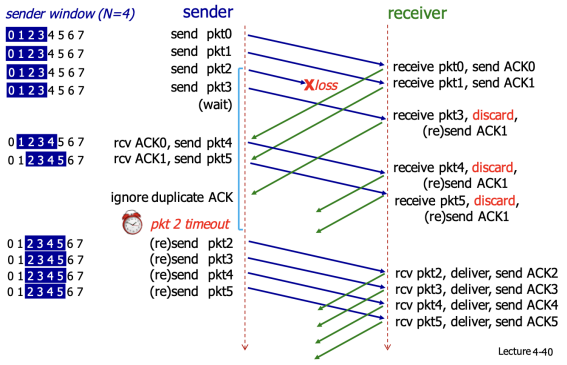


rdt	network layer channel	new features
1.0	no error	-
2.0	bit error in data	sender-side checksum, receiver-side ACK/NAK
2.1	bit error in data/feedback	receiver-side checksum, sender-side sequence number
2.2	same as 2.1	receiver-side sequence number
3.0	bit error/pkt loss in data/feedback	timeout/re-transmission

Pipelined protocols - Go-Back-N/Selective Repeat

Go-back-N

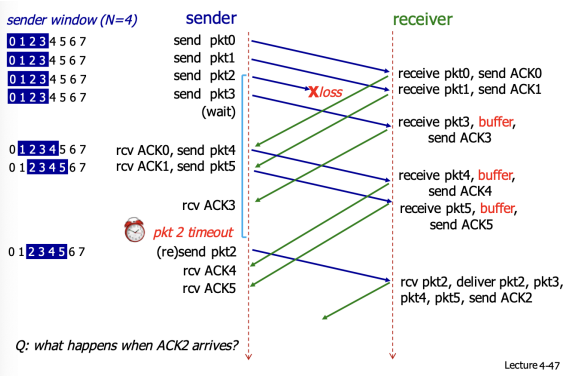
- Receiver
  - no buffer/ack for out-of-order pkts
  - only sends cumulative ack for most-recent pkt
- Sender
  - timer for oldest unacked pkt
  - retransmits all unacked pkts when timeout



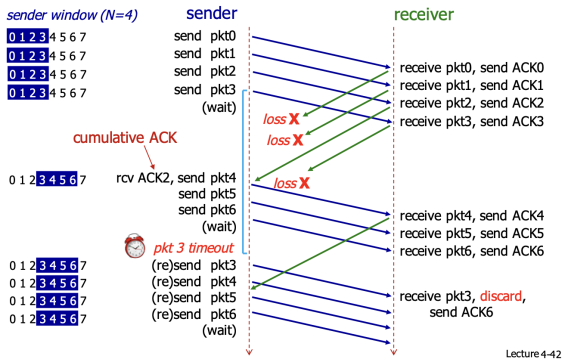
timer for ACK2 starts only after ACK1 has been received (not from the moment when pk2 is sent)

Selective Repeat

- Receiver
  - maintains a sliding window to buffer out-of-order pkts
  - sends individual ack for each packet
- Sender
  - maintains timer for each unacked pkt
  - retransmits only the unacked pkt when timeout



when ACK2 arrives, advances the sliding window to the next unACKed packet 6



▼ It is generally a reasonable assumption, when sender and receiver are connected by a single wire, that packets cannot be reordered within the channel between the sender and receiver. However, when the “channel’

connecting the two is a network, packet reordering may occur. One manifestation of packet reordering is that old copies of a packet with a sequence or acknowledgement number of *x* can appear, even though neither sender's nor receiver's window contains *x*. With packet reordering, the channel can be thought of as essentially buffering packets and spontaneously emitting these packets at any point in the future. What is the approach taken in practice to guard against such duplicate packets?

The approach taken in practice is to ensure that a sequence number is not reused until the sender is "sure" that any previously sent packets with the same sequence number are no longer in the network.

Firstly, TCP use large sequence number field (32-bit) to lower the chance a sequence number is to be reused. Secondly, a packet cannot "live" in the network forever. For example, IP protocol specifies TTL in packet header to ensure that datagrams do not circulate infinitely in the network. This field is decreased by one each time the datagram arrives at a router along the end-to-end path. If TTL field reaches 0, router will discard this datagram. In practice, a maximum packet lifetime of approximately three minutes is assumed in the TCP extensions for high-speed networks.

▼ **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).

A sender S is sending packets to a receiver R using the Go-Back-N protocol. The sender's window is of size 4. After transmitting for a while, the first packet in the sender's window is pk (assume k>1). Let a packet with sequence number i be pi. Assume that packets may be lost or corrupted but won't be reordered.

Which of the following statements are TRUE?

- R might have received p_k already.
- S must have received ACK for p_{k-1} already.
- It is possible that R has sent ACK for p_{k+3} already.
- S must not have received ACK for p_{k+1}.
  - ack is cumulative → if received ack k+1 → sliding window would have moved to k+2
- R must have received p_{k-1} already.

Consider a sender and a receiver communicating using Selective Repeat protocol. Every packet embeds a 3-bit sequence number field. The sender has just sent a packet with sequence number 0. Sender's window size is 3.

Possible sequence number of the next packet transmitted by the sender: 6 7 0 1 2 3

L5: TCP

TCP overview

- point-to-point: one sender, one receiver (both have buffers)
- connection-oriented
- full-duplex data: bi-directional data flow in same connection
- reliable, in-order byte stream: no “message boundaries”
- pipelined: dynamic window size set by congestion/flow controls
- implements
  - timeout event
  - cumulative ACK
  - checksum

    - fast retransmission
- does not implement
    - stop and wait

- Port 80 is the port number assigned to commonly used internet communication protocol, Hypertext Transfer Protocol (HTTP). It is the port from which a computer sends and receives Web client-based communication and messages from a Web server and is used to send and receive HTML pages or data.
- initially sequences number (ISN) is randomly chosen from [0, 2^32 - 1] (if header is 32 bits)

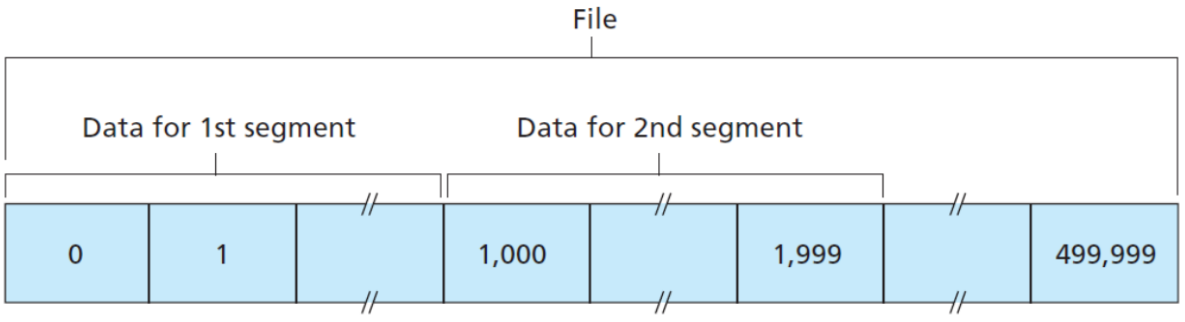
Sequence Number

- byte number of the first byte of data in a segment

Maximum Segment Size (MSS)

- does not include header size
- actual segment size = MSS + header size (negligible if not mentioned)

- **example: send a file of 500,000 bytes; MSS is 1000 bytes.**



Dividing file data into TCP segments

- **Seq. # of 1st TCP segment: 0, 2nd TCP segment: 1000, 3rd TCP segment: 2000, 4th TCP segment: 3000, etc.**

▼ **Consider transferring an enormous file of *L* bytes from Host A to Host B. Assume an MSS of 512 bytes.**

What is the maximum value of *L* such that TCP sequence numbers are not exhausted? Recall that the TCP sequence number field is 32 bits.

Find how long it takes to transmit the max file (L bytes). Assume that a total of 64 bytes of transport, network, and data-link header are added to each packet before the resulting packet is sent out over a 155 Mbps link. Ignore flow control, congestion control and assume Host A can pump out all segments back to back and continuously.

max L = 2^{32} bytes

Number of packets = L / MSS = 232 / 512 = 8,388,608

64 bytes of headers will be added to each packet. Therefore,

Total bytes sent = 232 + 8388608 * 64 = 4,831,838,208 bytes

Transmission delay = 4831838208 * 8 / (155 * 106) ≈ 249 seconds

▼ **A file of size 9990 bytes is transferred over a TCP connection. The connection is still open after file transmission. MSS is 1000 bytes and TCP sends as much data as possible in a segment. TCP also adds 20 bytes header to each segment.**

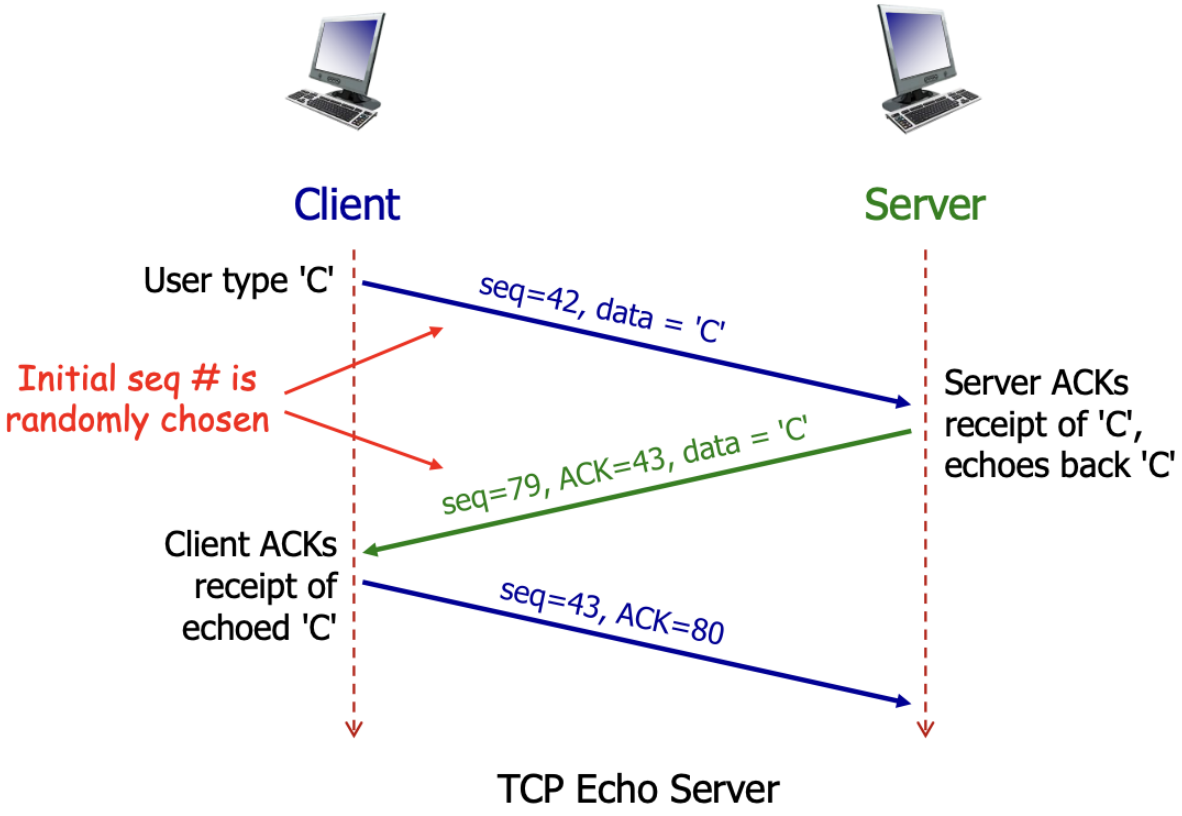
What is the size of the last TCP segment?

num of packets = 9990/1000 = 10

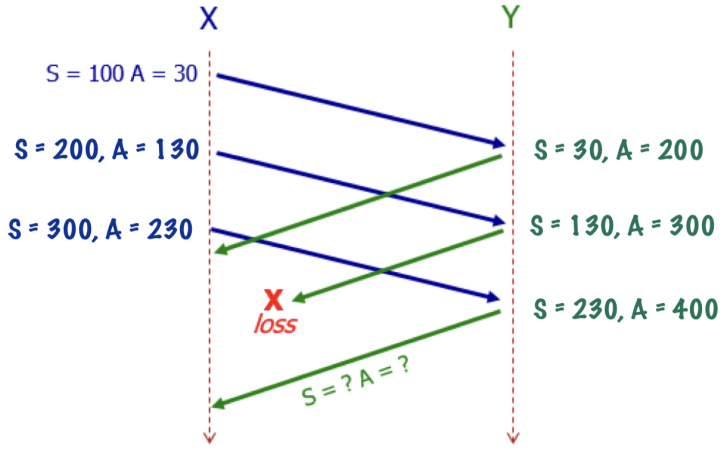
size of last segment = 9990 - 9*1000 + 20 = 1010

ACK

- seq # of the next byte expected
- cumulative ACK



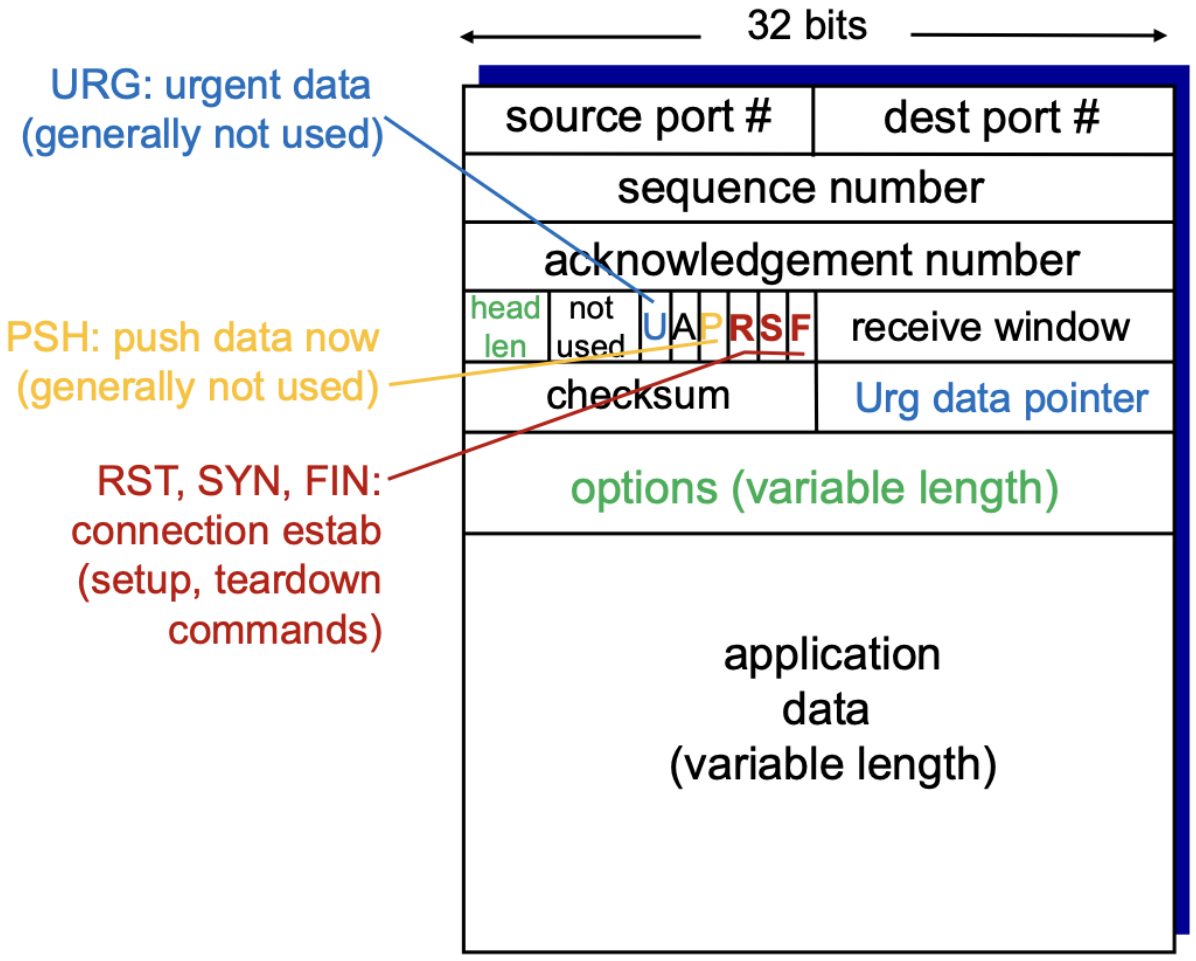
The following diagram shows two hosts **X** and **Y** communicating over a channel using TCP. **X** and **Y** are sending data to each other (recall that TCP supports bi-directional communications). Each segment contains 100 bytes of data. None of the segments shown in the figure are retransmitted packets, nor are they corrupted. However, the second segment send by **Y** is lost.



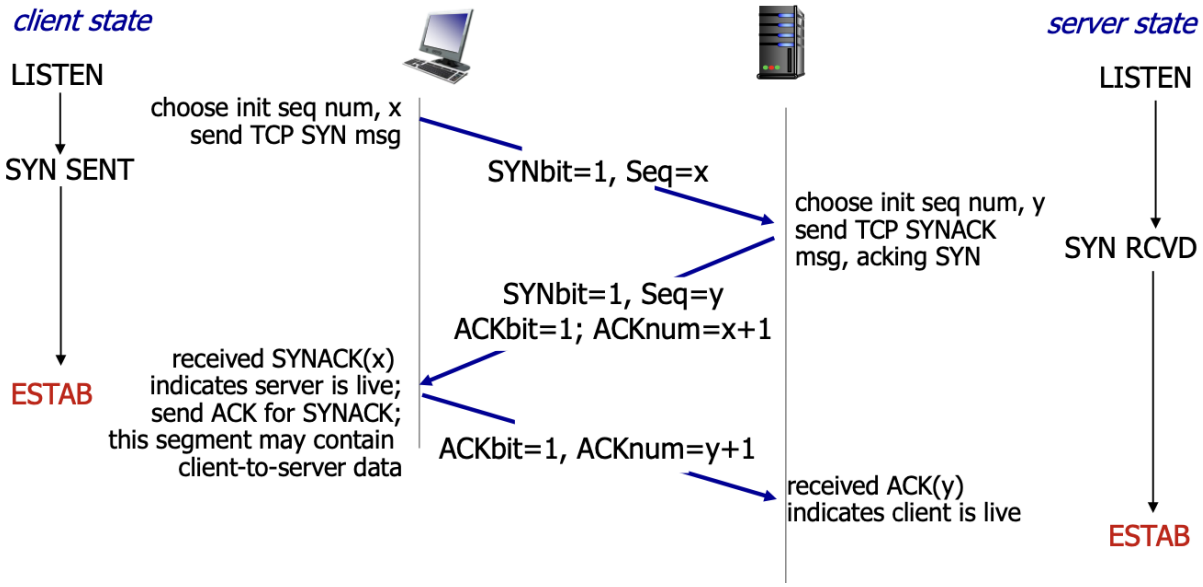
Flow Control

- receiver controls sender so sender wont overflow receiver's buffer by transmitting too much, too fast
- receiver includes receive window `rwnd` value in TCP header to signal the amount of available buffer space

TCP header (20 bytes)

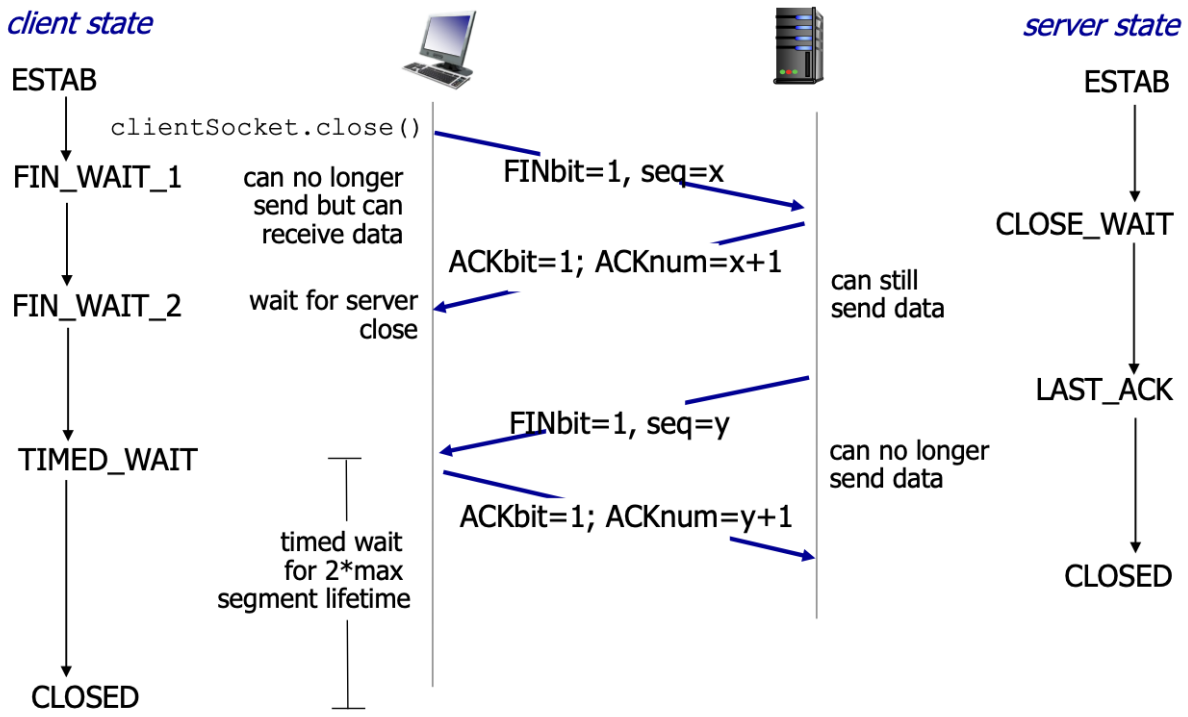


RST = reset (unexpected pkt arrives);
SYN = initiate and establish connection, synchronise seq number between devices;
FIN bit = close connection



3 way handshake

- **Step 1:** In the first step, the client establishes a connection with a server. It sends a segment with SYN and informs the server about the client should start communication, and with what should be its sequence number x.
- **Step 2:** In this step **s**erver responds to the client request with SYN-ACK signal set. ACK helps you to signify the response of segment that is received and SYN signifies what sequence number it should able to start with the segments.
- **Step 3:** In this final step, the client acknowledges the response of the Server, and they both create a stable connection will begin the actual data transfer process.



close connection

```
NextSeqNum = InitialSeqNum
SendBase = InitialSeqNum

loop (forever) {
  switch (event):
    event: data received from application abovecreate TCP segment with seq num NextSeq
Num
      if (timer currently not running) start timer // sender keeps only one timer
      pass segment to IP
      NextSeqNum += len(data)
      break;

  event: timer timeout
    retransmit not-yet-acked segment with smallest seq num
    start timer
    break;

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

}
```

ACK generation

event at receiver	TCP receiver action
arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK (hope to receive the next seq immediately to save 1 ack)
arrival of in-order segment with expected seq #. One other segment has ACK pending	immediately send single cumulative ACK, ACKing both in-order segments (only wait for one additional pkt)
arrival of out-of-order segment higher-than-expect seq. # . Gap detected	immediately send *duplicate ACK*, indicating seq. # of next expected byte
arrival of segment that partially or completely fills gap	immediate send ACK, provided that segment starts at lower end of gap

TCP Timeout value

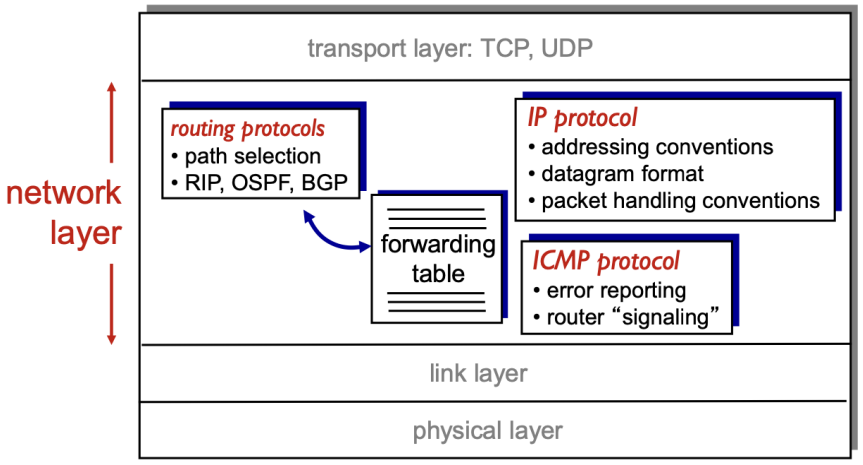
- longer than RTT
- **EstimatedRTT = (1-α)*EstimatedRTT + α*SampleRTT**
  - typical α = 1/8
- **DevRTT = (1-β)*DevRTT + β*|SampleRTT-EstimatedRTT|**
  - typically, β = 1/4
- **TimeoutInterval = EstimatedRTT + 4*DevRTT (safety margin)**

TCP Fast Retransmit

- if sender receives 4 ACKs for same data (“triple duplicate ACKs”), resend unacked segment with smallest seq #
  - likely that unacked segment lost, so don’t wait for timeout

L6: Network Layer

- host-to-host communication



Data Plane

- local, per-router function
- determines how datagram arriving on router input port is forwarded to router output port

Control Plane

- network-wide logic
- determines how datagram is routed among routers along end-end path from source host to destination host
- two approaches
  - transitional routing algo: implemented in router
  - software-defined networking (SDN): implemented in (remote) servers

IP addressing

- 32-bit identifier for host, router interface
- interface: connection between host/router and physical link
- how does a host get IP address?
  - hard-coded by system admin in a file
  - Dynamic Host Configuration Protocol: dynamically get address from server

Subnets

- **a network formed by a group of "directly" interconnected hosts.**

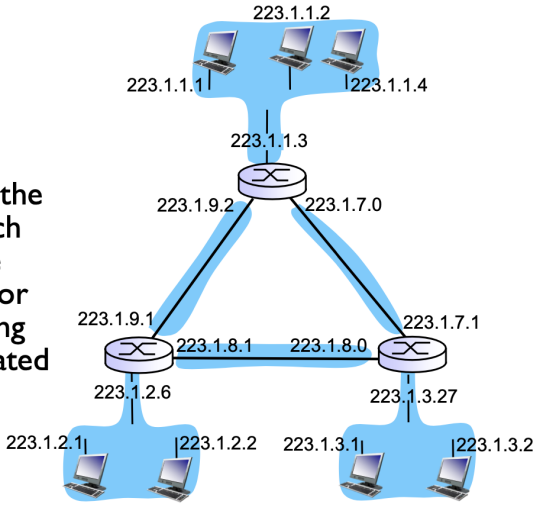
- Hosts in the same subnet can physically reach each other without intervening router.
  - They connect to the outside world through a router.
  - Hosts in the same subnet have the same network prefix of IP address.

Subnets

how many?

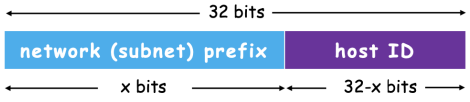
recipe

- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks

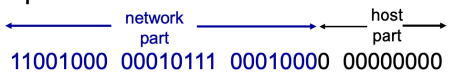


6 subnets

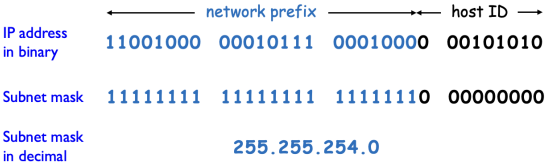
- An IP address logically comprises two parts:



- CIDR: Classless InterDomain Routing
  - arbitrary length for the subnet portion (network prefix)
  - address format: a.b.c.d/x, where x is # bits in subnet portion of address
- Example 200.23.16.0/23



- example: for IP address 200.23.16.42/23:

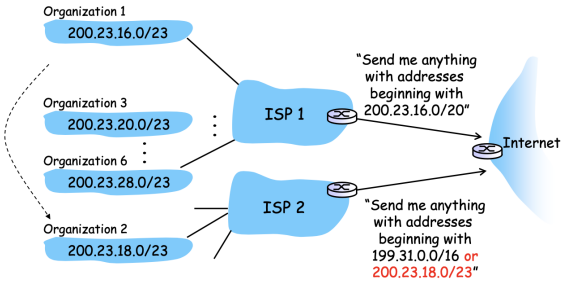


- used to determine which network an IP address belongs to (use bitwise AND operation)

- Subnet mask is made by setting all network prefix bits to "1"s and host ID bits to "0"s.

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.
10.0.0.0/8 172.16.0.0/12 192.168.0.0/16	Private addresses, can be used without any coordination with an Internet registry.
255.255.255.255/32	Broadcast address. All hosts on the same subnet receive a datagram with such a destination address.

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



Longest Prefix Matching

- 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)

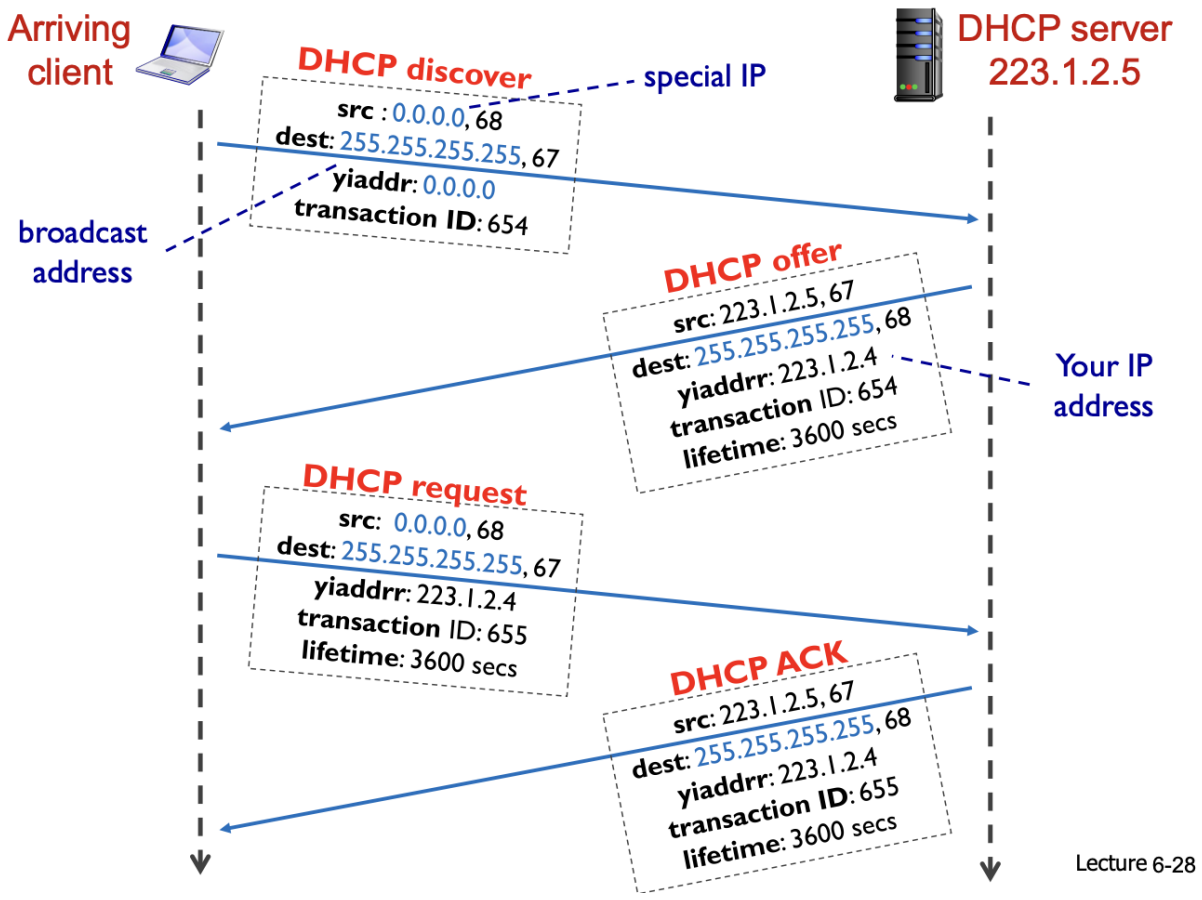
match the longest prefix

Forwarding Table at R3

Destination Address Range	Destination Address Range in Binary	Next Hop
200.23.16.0/20	11001000 00010111 00010000 00000000	R1
200.23.18.0/23	11001000 00010111 00010010 00000000	R2
199.31.0.0/16	11000111 00011111 00000000 00000000	R2

Dynamic Host Configuration Protocol (DHCP)

- allow host to dynamically obtain its IP address from network server when it joins network
  - only hold address when connected ⇒ allow reuse of addresses
  - can renew its lease on address in use
- host broadcasts "DHCP discover" [optional]
- server responds with "DHCP offer" [optional]
- host requests IP address "DHCP request"
- server sends address "DHCP ack"
- server port num: 67
- client port num: 68

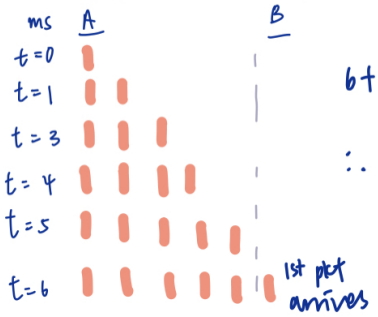


broadcast address in discover as it doesnt know the server

PYP

1.3 10 packets are continuously sent over a 1 Mbps link. Each packet is of 1,000 bits long and RTT is 10 ms. What is the throughput of the link (i.e. on average, how many bits transmitted per second)?

A. 511.856 bps
B. 511.856 Kbps
C. 500 bps
D. 500 Kbps
E. 666.667 Kbps



6+9=15ms for all 10pkt to arrive

$\therefore \frac{10 \times 1000}{15 \times 10^{-3}} = 667$ Kbps

Two hosts A and B are connected via a router. The link rate is 1 Mbps and propagation delay is 40 ms per link. The maximum size of a packet is 1 Kb and packet header is 80 bits. Suppose sender sends as much data as possible in a packet, packets are sent continuously and no packet is corrupted or lost during transmission.

How long (in milliseconds) does it take to send a 400 Kb file from A to B (from when the first bit of the first packet leaves A to when last bit of the last packet arrives at B)?

$\#pkt = \frac{400 \times 10^3}{1000 - 80} = 435$ pkt

total # of bits $= 435 \times 80 + 400 \times 10^3 = 434800$ bits

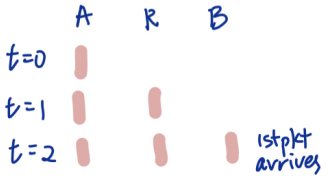
$\Rightarrow$ 434 pkt of max size

$\Rightarrow$ last pkt size = 800 bits

$\frac{1Kb}{1Mbps} = 1ms$ to send a pkt of 1Kb across a link (A to R / R to B)

Total time $= \frac{1000}{10^6} + \frac{434800}{10^6} + 40 \times 10^{-3} \times 2 = 0.5158\ s$

$= 515.8$ ms



2+434 = 436ms for all to arrive if all 1Kb

- t = 41ms: first pkt arrives at R
- t = 82ms: first pkt arrives at B
- subsequently, each pkt arrives at 1ms interval ⇒ 82 + 434 = 516ms

A host uses a variety of protocols to discover information about the network it is connected to. Which of the following statements is false?

- To perform a DNS lookup, a host must first discover the IP address of its local DNS server using DHCP.
- To send a packet outside the host's subnet, the host must first discover the IP address of its first-hop router using DHCP.
- To send a packet outside the host's subnet, a host must first discover the IP address of the destination host using DNS.
- To get an IP address assigned, a host must first discover the IP address of its DHCP server using DNS.
- To send a packet to another host in the same subnet, a host must first discover the MAC address of the destination host using ARP.

Which of the following statement about IP datagram is FALSE?

- Routing protocols determine the routes that datagrams take between sources and destinations

- TTL field of IP header prevents a datagram from circulating in the network forever.
- When a big datagram is fragmented into a series of smaller fragments, transport layer header will be replicated in each fragment.
    - IP (network) header will be replicated
- On the internet, datagrams from the same source may take different routes towards the destination.
- MTU of the link-layer protocol places a limit on the length of a datagram.

Cannot access the web page hosted at www.example.com. Which of the following is NOT the correct use of the corresponding tool?

- run `telnet` to check if www.example.com is running on port 80
- run `traceroute` to check if there is a route from the laptop to www.example.com
    - contact every router along the way to end host
- run `dig` to check if DNS is able to resolve the IP address of the host name www.example.com
- run `ping` to check if you can establish a TCP connection to www.example.com
    - ping checks if a host is alive using ICMP.
    - Ping is the name of a standard software utility, used to test connectivity either between client-client or client-server.
    - directly contacts the host (www.example.com)
- run `curl` to check if www.example.com is responding to a HTTP request correctly.

Telnet protocol allows a user to establish a TCP connection to a remote server. Consider the following command `telnet www.nus.edu.sg 80`.

Which of the following statement is TRUE?

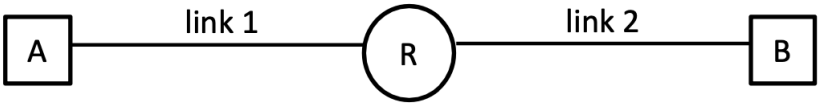
1. The command causes a DNS lookup for the IP address of **www.nus.edu.sg**.
2. The command causes a SYN packet to be sent to **www.nus.edu.sg**.
3. The command causes a HTTP request to be sent to **www.nus.edu.sg**.

4. The command causes host **www.nus.edu.sg** to open port 80 and listen for incoming connections.

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 sending host to the receiving host. Each packet is 1000 bytes long. RTT is 24 milliseconds. No packet is lost or corrupted during transmission and ACK packets are of negligible size. What is the throughput of the transmission (in Mbps)?

- Stop-and-wait protocol: sender waits for ack from the prev pkt before sending the next one
- 1000 bytes = 8000 bits
- sender takes 8ms to send 1 pkt
- total time taken = 10*(24+8) = 10*24 + 10*8000/10^6 * 10^-3= 320ms
- throughput = $\frac{80000}{320\times10^{-3}} = 0.25Mbps$

Two hosts **A** and **B** are connected by a router **R** as shown in the following diagram.



For link 1, link transmission rate is 1 Kbps and propagation delay is 100 milliseconds. For link 2, link transmission rate is 250 bps and propagation delay is 150 milliseconds. Suppose Host **A** sends 2000 packets to Host **B** continuously and each packet is 500 bits long. Host **A** starts sending the 1st packet at time t = 0.

When (**in seconds**) will Host **B** receive the k^{th} packet ($1 \le k \le 2000$)?

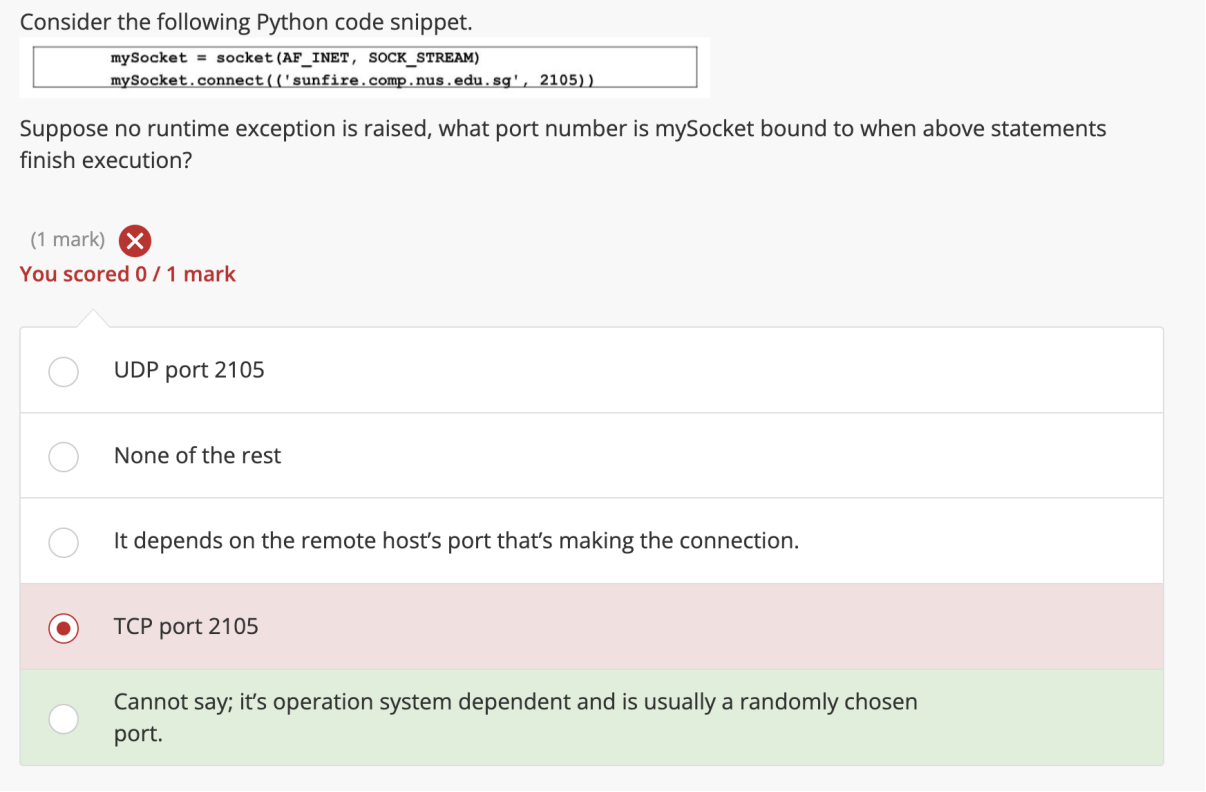
- link 1 takes 0.5s to send a pkt
- link 2 takes 2s to send a pkt
- t = 0.6s: 1st pkt arrives at R
- t = 1.1s: 2nd pkt arrives at R (cannot be sent out by R immediately as R is still sending the 1st pkt)
- t = 2.6s: R sent 1st pkt

- t = 2.75s: 1st pkt arrives at B
- t = 4.6s: R sent 2nd pkt
- t = 4.75s: 2nd pkt arrives at B
- B receives the kth pkt at 0.75 + 2k

GBN sender has ONE timer(s), SR sender has MULTIPLE timer(s) and TCP sender has ONE timer(s).

- GBN/TCP keeps one timer for the oldest unacked pkt
- SR keeps one timer for each pkt

Suppose there are multiple unacknowledged packets. Upon a timeout event, GBN sender retransmits MULTIPLE packet(s), SR sender retransmits ONE packet(s) and TCP sender retransmits ONE packet (s).



- port 2105 is on server side
- local port depends on OS

```
mySocket = socket(AF_INET, SOCK_STREAM)
mySocket.connect(("www.example.org", 12345))
```

Suppose the above code snippet is executed with no error, which of the following protocols is NOT invoked?

- Protocol evoked
  - DNS
  - TCP
  - UDP
- Protocol not evoked
  - HTTP

Which of the following statement is false?

- Local DNS record is always up-to-date
- DNS records are stored in distributed databases on many name servers
- DNS caching can reduce the delay in receiving a DNS response
- DNS root servers do not store all IP to hostname mappings in the Internet
- In a resource record, TTL stands for time to live.

Which of the following statement about HTTP is false?

- in an HTTP response message, state code will be 403 Forbidden, if access to the Web object requested by the client is denied.
- In an HTTP/1.0, an HTTP request message and its corresponding response message are sent through different TCP connections. (Same connection)
- More than one persistent connection may be launched to download an HTML file and other Web objects referenced by the HTML file
- an HTTP response message may contain status line and header lines only and does not carry any data
- a cookie will not be sent by the browser to the Web server if an expiration date has been set and passed.

Two hosts A&B are communicating over a link using a sliding window protocol. Which of the following condition would increase the utilisation of the sender?

(i) decreasing the transmission rate (spend more time sending data, increase utilisation rate)

(ii) decreasing the size of data packet

(iii) decreasing the value of retransmission timeout (spend more time sending data than waiting)

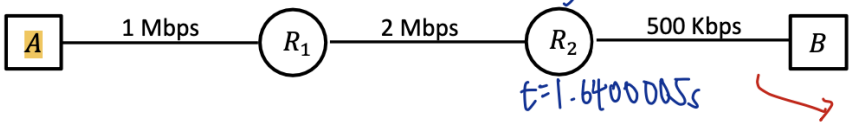
(iv) increasing the distance between A&B (increase d_prop, increase RTT)

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.

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

- Sender must attach a sequence number to every data packet.
- If sender set the timer properly, receiver definitely won't receive duplicate packets.
- Receiver should discard corrupted data packet but must acknowledge it.
- In a feedback packet, receiver must explicitly include the sequence number of the data packet being acknowledged.
- None of the above

14. As shown in the following diagram, the path from host *A* to host *B* comprises three links of rates 1Mbps, 2 Mbps and 500 Kbps respectively. Propagation delay is 500 ms per link. Two network devices R_1 and R_2 each acts on individual bits and forwards every bit to the next hop once it is received. You may assume both R_1 and R_2 have infinite memory.

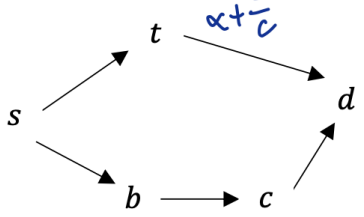


Suppose *A* sends 80,000 bytes to *B*. What is the throughput of transmission?

A. 45 Kbps
B. 48 Kbps
C. 171 Kbps
D. 230 Kbps
E. 360 Kbps

D

15. Consider the following network.

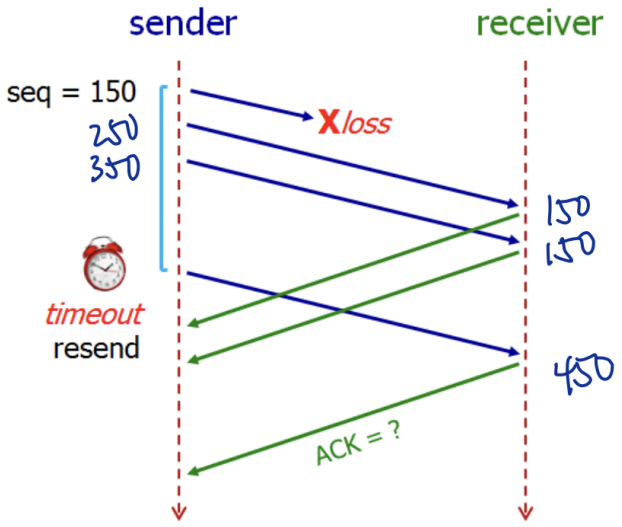


in which the source s generates traffic for destination d in equal sized packets of size L bits. The capacity of each link is c bps. The source s chooses links $s \to t$ with probability p and link $s \to b$ with probability $1 - p$. Ignore the queueing delays in the nodes t, b, c. If the propagation delay on each link is α seconds, what is the average packet delay from $s \to d$? Packet delay is defined as the time it takes from when the first bit of the packet leaves s until the time the last bit of the packet arrives at d.

A. $2.5(\alpha + \frac{L}{c})$
B. $\alpha(3-p) + (2-p)\frac{L}{c}$
C. $(3-p)(\alpha + \frac{L}{c})$
D. $\left(2\alpha + \frac{L}{c}\right)p + \left(3\alpha + \frac{L}{c}\right)(1-p)$
E. None of the above

C

11. Consider the following diagram in which sender sends 3 TCP segments to receiver. Before the first segment is sent out, SendBase is 149. MSS is 100 bytes. The first segment is lost and retransmitted when timer expires. Receiver buffers out-of-order segments for eventual in-order delivery to application.



What is the smallest and largest possible ACK numbers in the last acknowledgement packet?

A. 350; 450

B. 150; 450

C. 50; 350

D. 150; 350

E. None of the above

1314s2

1. A Web server acts as an application gateway to CGI scripts. The server just receives an HTTP
request. The first three lines of the request are shown below.

```
POST /news.pl HTTP/1.1
Content-Length: 310
Content-Type: application/x-www-form-urlencoded
```

Which of the following is TRUE?

- The request does not contain a query string.

    - The query string is part of the HTTP request body
- The environment variable REQUEST METHOD should be set to `news.pl`
    - REQUEST METHOD is POST
- The total size of the HTTP request (including both header and body) is 310 bytes.
    - exclude 310 bytes
- The application named `x-www-form-urlencoded` will be invoked to decode the URL.
    - `news.pl` will be invoked
- The server must close the TCP connection immediately after sending the response.
    - HTTP/1.1 is persistent
- none true

Consider a Java implementation of the stop-and-wait (RDT 3.0) (timeout for ACK available) protocol, using classes and methods similar to your Assignment 2. A student has implemented the sender correctly ac- cording to the state diagram of the protocol. In the receiver, the student implemented the following:

```
byte[] recv() throws IOException, ClassNotFoundException {
      DataPacket p = udt.recv();
      while (p.isCorrupted || p.seq != seq)
          p = udt.recv();

      udt.send(new AckPacket(p.seq));
      seq = 1 - seq;
      return deliverData(p);
}
```

Here, `seq` is the expected sequence number (either 0 or 1) at the receiver, and deliverData is a method that extracts and returns the payload from packet p.

The protocol is used over a channel that may lose or corrupt a packet, but always delivers packets in the order that they are sent.

We say that the receiver waits forever, if it is blocked at the call `udt.recv()`, waiting to receive a packet that will never be sent. We say that the sender loops forever, if it repeatedly retransmits the same packet over and over again.

Which of the following statement CORRECTLY describes the behaviour of the protocol implemented above?

- A single corrupted data packet is sufficient to cause the sender to loop forever.
- A single corrupted data packet is sufficient to cause the receiver to wait forever.
  - if corrupted, no ACK sent → timeout → server resend pkt → will not wait forever (A & B are wrong)
- A single loss ACK packet is sufficient to cause the sender to loop forever.
  - A loss ACK would cause the sender to send a duplicate packet to the receiver after timeout. Since the receiver does not send an ACK on duplicate packet, the sender keeps timing out and loops forever.
- A single loss ACK packet is sufficient to cause the receiver to wait forever.
- A single premature timeout is sufficient to cause the sender to loop forever.
  - A premature timeout would cause the sender to send a duplicate packet as well, but eventually ACK is received by the sender, so the sender won't loop forever.

A and B are communicating over a link with a sliding window protocol. Which of the following condition would reduce the utilisation of a link?

- increasing the transmission rate
- increasing the size of the packet
- increasing the value of retransmission timeout
  - when a packet is loss, the sender spend more time idling waiting for ACK.
- increasing the size of the sending window
- reducing the distance between A and B

equation of utilisation: $U = \frac{NL/R}{L/R+RTT} = \frac{NL}{L+R \cdot RTT}$

- R: transmission rate
- L: pkt size
- N: num pkts

14s2

On **Sunfire** server, we type the command `dig -t a www.duke.edu +trace` and observe the following outputs:

```
; <<>> DiG 9.6-ESV-R8 <<>> www.duke.edu +trace
;; global options: +cmd
.                   155852  IN      NS      b.root-servers.net.
.                   155852  IN      NS      c.root-servers.net.
.                   155852  IN      NS      d.root-servers.net.
.                   155852  IN      NS      e.root-servers.net.
.                   155852  IN      NS      f.root-servers.net.
.                   155852  IN      NS      g.root-servers.net.
.                   155852  IN      NS      h.root-servers.net.
.                   155852  IN      NS      i.root-servers.net.
.                   155852  IN      NS      j.root-servers.net.
.                   155852  IN      NS      k.root-servers.net.
.                   155852  IN      NS      l.root-servers.net.
.                   155852  IN      NS      m.root-servers.net.
.                   155852  IN      NS      a.root-servers.net.
;; Received 492 bytes from 137.132.85.2#53(137.132.85.2) in 13 ms

edu.                172800  IN      NS      a.edu-servers.net.
edu.                172800  IN      NS      f.edu-servers.net.
edu.                172800  IN      NS      d.edu-servers.net.
edu.                172800  IN      NS      c.edu-servers.net.
edu.                172800  IN      NS      g.edu-servers.net.
edu.                172800  IN      NS      l.edu-servers.net.
;; Received 265 bytes from 192.33.4.12#53(192.33.4.12) in 182 ms

duke.edu.           172800  IN      NS      avallone.stanford.edu.
duke.edu.           172800  IN      NS      dns-auth-01.oit.duke.edu.
duke.edu.           172800  IN      NS      dns-auth-02.oit.duke.edu.
;; Received 194 bytes from 192.31.80.30#53(192.31.80.30) in 259 ms
```

from Local

from Root

from TLD

```
www.duke.edu.      21600   IN      CNAME   duke.edu.
duke.edu.          21600   IN      A       54.191.241.8
duke.edu.          21600   IN      A       54.68.155.51
duke.edu.          21600   IN      NS      dns-auth-02.oit.duke.edu.
duke.edu.          21600   IN      NS      dns-auth-01.oit.duke.edu.
;; Received 164 bytes from 152.3.105.232#53(152.3.105.232) in 270 ms
```

from Authoritative

- Address Mapping record (A Record)—also known as a DNS host record, stores a hostname and its corresponding IPv4 address.
- CNAME records must point to a domain and not an IP address. A domain with a CNAME record can either point to another domain with a CNAME record or to a domain with an A record.
    - When a DNS client requests a record that contains a CNAME, which points to another hostname, the DNS resolution process is repeated with the new hostname.
- Name Server records (NS Record)—specifies that a DNS Zone, such as “example.com” is delegated to a specific Authoritative Name Server, and provides the address of the name server.

▼ **Write down one IP address of a local DNS server**

137.132.85.2

▼ **Write down one IP address of a root DNS server**

192.33.4.12

▼ **Write down one IP address of a top-level domain DNS server**

192.31.80.30

▼ **Write down one IP address of `www.duke.edu`**

54.191.241.8

▼ **What is the canonical name of `www.duke.edu`?**

duke.edu

▼ **What port number does a DNS server listen to?**

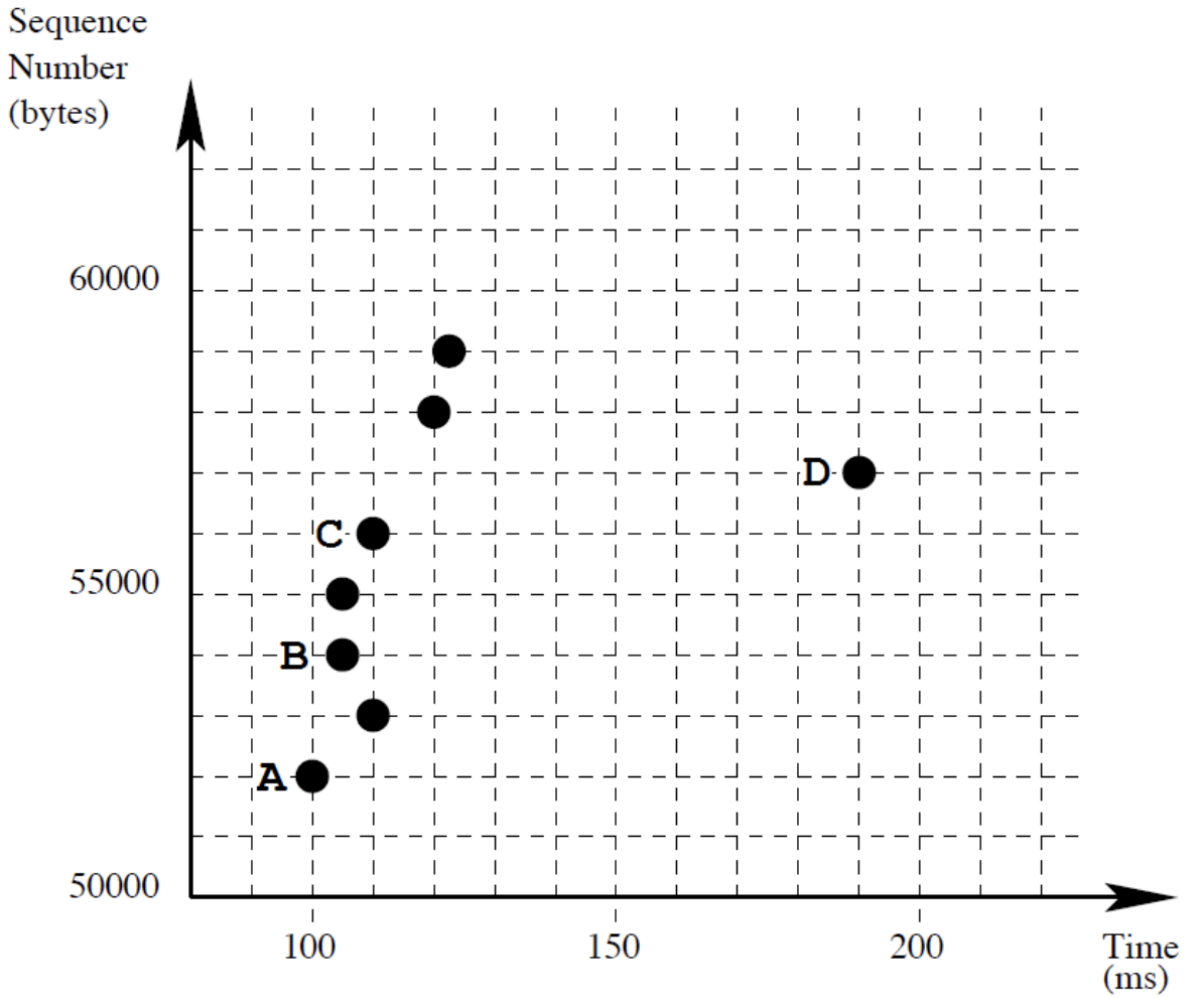
53

Host A sends 5 data packets to host B using TCP protocol. Each data packet contains 10 bytes of application data. Within 100 ms duration, B receives 5 in-order data packets and accepts all of them. How many ACK packets will B send out?

- [TCP ACK Generation] generate 1 ACK for at most 2 packets received within 500ms → 3 ACK generated

15s2

The following graph shows the time sequence graph for a TCP connection between host X and host Y. Each dot represents a TCP segment received at host Y, plotting the sequence number of the segment, versus the time at which it is received. A set of dots stacked above each other represents a series of packets that are received back-to-back by the receiver. The packet labelled with A is the first data packet sent by X. The packet labelled with D is a re- transmitted packet.



Does Y buffer out-of-order packets or discard them?

- Yes, Y buffers out-of-order packets.
- The packet with sequence number 53000 is an out-of-order packet.
- If it were discarded by receiver, X will not retransmit D before this packet is retransmitted (and acknowledged).
- This is because TCP sender only maintains one timer and resends the oldest unacknowledged packet upon timeout.

Suppose X sets a reasonable timeout value for this TCP connection. Retransmission is always successful. Estimate the timeout value that X chooses. State your assumptions.

- Assumption: packets may be lost or corrupted but will not be reordered by the network.

- The previous packet C is received at 110 ms. Once corresponding ACK reaches X, X will start a timer for packet D. When timer expires, D will be resent and received by Y at 190 ms.
- Assume propagation delay is d ms. ACK of packet C take d to reach X. Timeout period is (slightly greater than) $2d$. Retransmission takes another d.
- Therefore $4d = 190 - 110$.
- Timeout value chosen by X is $2d$ which is 40 ms.

Mock Paper 2

Two hosts *A* and *B* are 2,000 km apart and are connected directly using a link with propagation delay of 800 bit times and propagation speed of 2.5 * 108 m/s. *A* is sending a sequence of packets, each is 100 bytes in size, to *B*.

How long does it take for *B* to receive a packet?

$d_{prop} = 8ms = 800\ bit\ times$

link rate = 1s ⇒ 800/8*1000 = 100Kbps

B takes $d_{prop} + d_{trans} = 8 + 8 = 16ms$

***A* is using a sliding window protocol to communicate with *B*. What is the minimum window size *A* should use for the link to be fully utilised?** 3

Time for B to receive one packet + d_prop (for ACK to be returned, assuming ACK is of negligible size)

= 0.016s + 0.008s

= 0.024s

To fully utilize the link, it should be transmitting throughout the whole RTT.

Time to transmit one packet = 0.008s (calculated in part a)

Hence minimum sliding window size = RTT / d_trans per packet = 3

Figure 1 shows the finite state machine of a protocol designed to run over a channel with the following properties: (P1) can corrupt packet, (P2) can lose packets, and (P3) has an unknown round trip time.

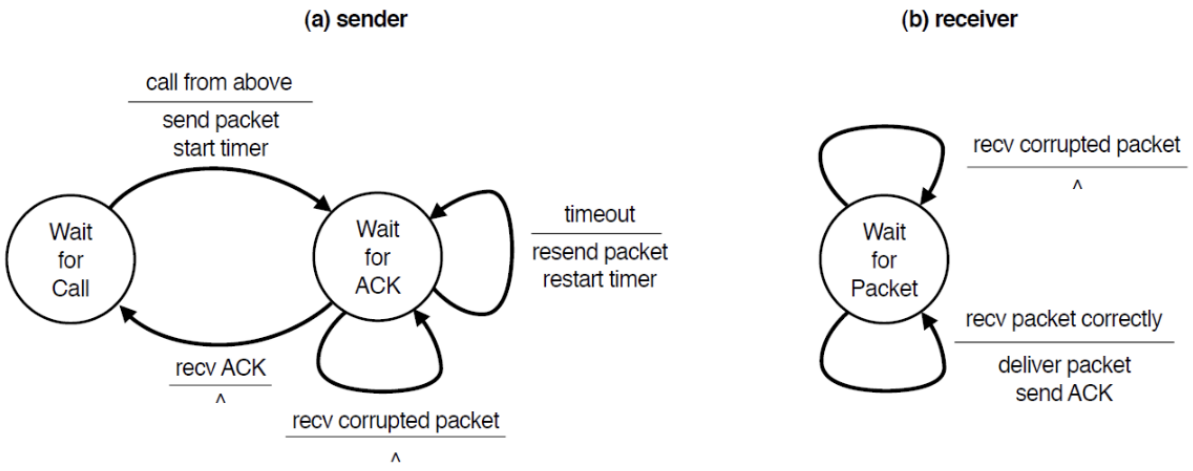


Figure 1: Finite State Machine of a Protocol

(a) Is it possible for this protocol to deliver the same packet twice to the application? Either give an example where the same packet is delivered twice by drawing a timing diagram, or argue why every packet will only be delivered once.

Yes. Premature timeout and retransmitted packet delivered to application.

(b) Is it possible for this protocol to not detect a lost data packet? Either give an example where a lost packet is not detected by drawing a timing diagram or argue why a packet loss is always detected.

Yes. Premature timeout and retransmission. The next new packet is lost but undetected because the sender treats the duplicate ACK as the acknowledgement for the next packet.

(c) Can we remove only one of the network properties P1, P2, P3 so that the protocol works as intended without modification? Justify your answer. P3 (or No?)

2122s2 midterm

Which of the following statement regarding rdt 3.0 is FALSE?

- None of the other choices.
- Receiver may ignore duplicate packet and does not send feedback.
- Sender can simply ignore corrupted ACKs.
- Receiver may discard corrupted packet and does not send feedback.
- Sender can simply ignore duplicate ACKs.

Which of the following is NOT a possible behaviour of a TCP receiver?

- When receiving an out-of-order segment, the receiver drops the segment immediately.
- When receiving an out-of-order segment, the receiver does not send an ACK immediately.
- When receiving an out-of-order segment, the receiver saves it in its receiving buffer.
- When receiving an in-order segment, the receiver sends an ACK immediately.
- When receiving an in-order segment, the receiver does not send an ACK immediately.

Which of the following implementation is WRONG for Selective Repeat (SR)?

- None of the other options.
- When a sender receives an ACK for a packet, it will mark the packet as received only if the ACK number is within its current sliding window.
- When a receiver receives a packet, it will buffer the packet only if the packet is within its current sliding window.
- Exactly one packet will be sent upon a timeout event.
- When a receiver receives a packet, it will send an ACK of the packet only if the packet is within its current sliding window.
    - will send ACK for previous packets also (e.g. duplicate pkt due to early retransmission)

Suppose you have created a website *www.mywebsite.com*. To make your website publicly accessible to the Internet, you need a/an AUTHORITATIVE DNS server to save the mapping between the URL of your website and the IP address of your web server. Furthermore, any user who wants to access your website needs to contact a/an LOCAL DNS server to obtain the mapping.

cause DNS query

- dig
- nslookup
- telnet
- curl
- traceroute