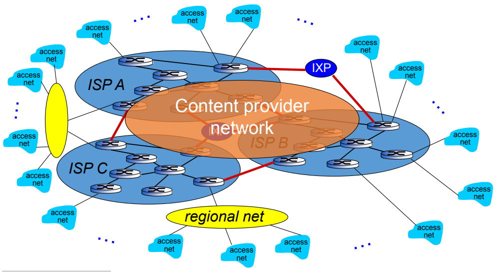
Internet Service Providers (ISPs) - network of packet switches and communication links.

- A variety of types of network access to the end system.
- Internet access to content providers, connecting Web sites and video servers directly to the Internet.
- Each ISP network, whether upper-tier or lower-tier, is managed independently.



Protocols

A protocol defines the format and the order of messages exchanged between two or more communicating entities, as well as the actions taken on the transmission and/or receipt of a message or other event.

Applications communicate using protocol.

Application-layer Protocols

- Types of messages exchanged: request, response
- Message syntax: what fields & how fields are delineated
- Message semantics: meaning of information in fields
- Rules: for when and how applications send & respond to messages
- Open protocols: defined in RFCs, allows for interoperability
- Proprietary protocols

packet switch faster for shorter msg

if packet arrival faster than departure: buffer overflow problem - packet lost

Each packet switch has multiple links attached to it.

Each attached link has an output buffer (output queue)

Processing delay(micro)

Time required to examine the packet's header and determine where to direct the packet.

The time needed to check for bit-level errors in the packet that occurred in transmitting the packet's bits from the upstream node to router A

Queuing delay(micro to milli)

Time the packet waits to be transmitted onto the link - $(nL + (L - x))/R$

Transmission delay(micro to milli)

Time required to push all of the packet's bits into the link - L/R

Propagation delay(milli)

Time required to propagate from the beginning of the link to router B - **d**/s

Traffic intensity: La/R a:packets/sec

Bandwidth delay:

The product of number of bits that can be flowing in the link at one time - $\mathbf{R} \times \mathbf{d}_{prop}$

Throughput - rate - high or low

How many bits can be transmitted per unit time

Instantaneous throughput at any instant of time - rate at which Host B is receiving the file - bits/sec

Throughput is the bottleneck link measured for end-to-end communication

Link capacity(bandwidth): meant for a specific link

TCP

- Full-duplex connection: two processes can send messages to each other over the connection at the same time
- Congestion-control: attempts to limit each TCP connection to its fair share of network bandwidth
- Flow-control:Prevents sender from flooding receiver
- Easily enchanced at the application layer with SSL to provide security services

UDP

Doesn't include congestion-control

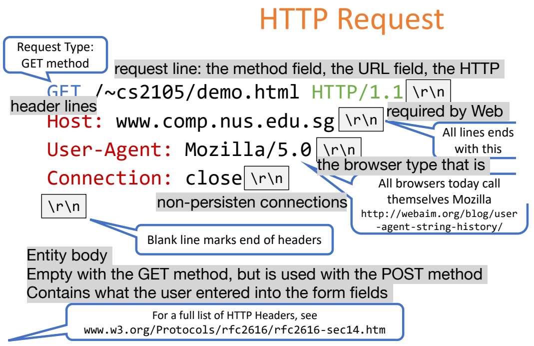
HTTP - stateless protocol

URL = the hostname of the server that houses the object + the object's path name

persistent connection - default

RTT(round-trip time) = packet-propagation delay + packet-queuing delays in intermediate routers and switches + packet-processing delays

For each connections, TCP buffers must be allocated and TCP variables must be kept in both the client and server - can place a significant burden on the Web server



HTTP Request

Request Type: GET method

request line: the method field, the URL field, the HTTP

header lines

```
GET /~cs2105/demo.html HTTP/1.1 \r\n
Host: www.comp.nus.edu.sg \r\n
User-Agent: Mozilla/5.0 \r\n
Connection: close \r\n
\r\n
```

required by Web

All lines ends with this

the browser type that is

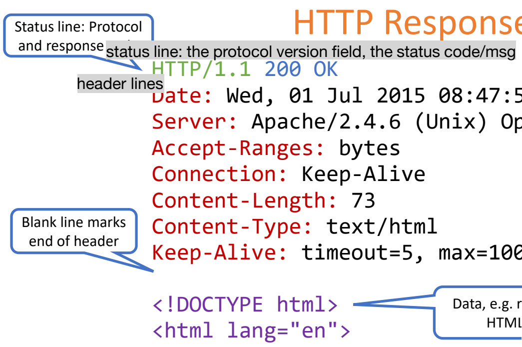
All browsers today call themselves Mozilla http://webaim.org/blog/user-agent-string-history/

non-persisten connections

Blank line marks end of headers

Entity body
Empty with the GET method, but is used with the POST method
Contains what the user entered into the form fields

For a full list of HTTP Headers, see www.w3.org/Protocols/rfc2616/rfc2616-sec14.htm



HTTP Response

Status line: Protocol and response

status line: the protocol version field, the status code/msg

header lines

```
HTTP/1.1 200 OK
Date: Wed, 01 Jul 2015 08:47:52 GMT
Server: Apache/2.4.6 (Unix) OpenSSL/1.0.1m
Accept-Ranges: bytes
Connection: Keep-Alive
Content-Length: 73
Content-Type: text/html
Keep-Alive: timeout=5, max=100

<!DOCTYPE html>
<html lang="en">
```

Blank line marks end of header

Data, e.g. requested HTML

Date: … - the time and date when the HTTP response was created and sent by the server

Server: … - the message was generated by … Web server - analogous the the User-agent:

Last-Modified: … - the time and date when the object was created or last modified - critical for object caching, both in the local client and in network cache servers

Content-Length: … - the number of bytes in the object being sent

Content-Type: … - the object type in the entity body

Cookies - 4 components

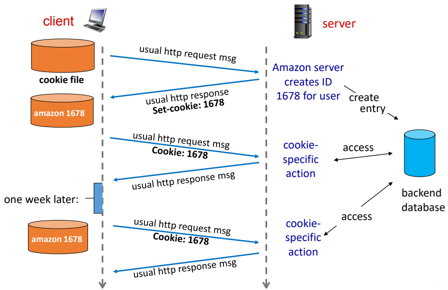
Allow sites to keep track of users

A cookie header line in reponse message

A cookie header line in request message

A cookie file kept on the user's end system and managed by the user's browser

A back-end database at the Web site



Web caching - proxy server

A network entity that satisfies HTTP requests on the behalf of an origin Web server.

Has its own disk storage and keeps copies of recently requested objects in this storage

Both a server and a client

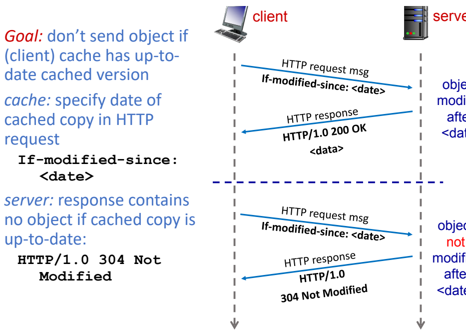
- TCP connection - browser & Web cache
- (optional) TCP connection - Web cache & the origin server

Can substantially reduce the response time for a client request

Can substantially reduce traffic on an institution's access link to the Internet - doesn't have to upgrade bandwidth, reducint costs. Reduce Web traffic in the Internet as a whole, improving performance for all applications.

Conditional GET - GET method + If-Modified-Since: header line

A mechanism that allows a cache to verify that its objects are up to date



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

root DNS servers - over 400 - provide the IP addresses of the TLD servers

top-level domain(TLD) DNS servers - provide the IP addresses of the authoritative DNS servers, responsible for .com, .org, .net, .edu, .uk, .sg

authoritative DNS servers - houses records

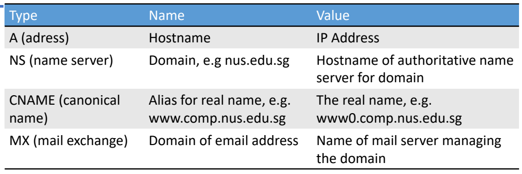
Local DNS Server - each ISP has one

Resource records (RRs)

Hostname-to-IP address mappings

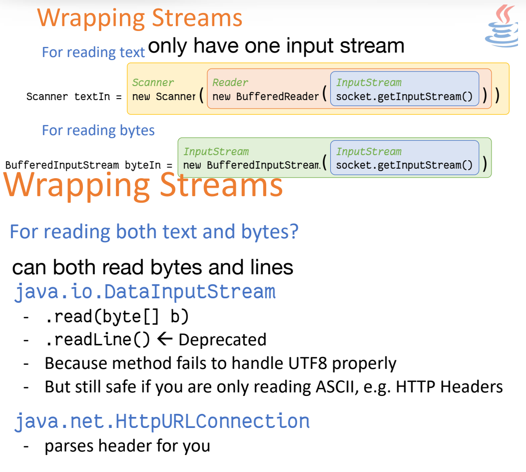
Each DNS reply message carries one or more resource records (Name, Value, Type, TTL)

`RR Format: <name, value, type, TTL>`



Type	Name	Value
A (adress)	Hostname	IP Address
NS (name server)	Domain, e.g nus.edu.sg	Hostname of authoritative name server for domain
CNAME (canonical name)	Alias for real name, e.g. www.comp.nus.edu.sg	The real name, e.g. www0.comp.nus.edu.sg
MX (mail exchange)	Domain of email address	Name of mail server managing the domain

nslookup/dig



Wrapping Streams

For reading text only have one input stream

```
Scanner textIn = new Scanner( new BufferedReader( socket.getInputStream() ) )
```

For reading bytes

```
BufferedInputStream byteIn = new BufferedInputStream( socket.getInputStream() )
```

Wrapping Streams

For reading both text and bytes?

can both read bytes and lines

`java.io.DataInputStream`

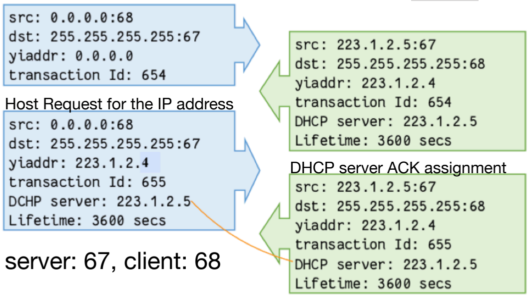
- `.read(byte[] b)`
- `.readLine()` ← Deprecated
- Because method fails to handle UTF8 properly
- But still safe if you are only reading ASCII, e.g. HTTP Headers

`java.net.HttpURLConnection`

- parses header for you

transport layer invisible to client and server

Dynamic Host Configuration Protocol(DHCP)



```
src: 0.0.0.0:68
dst: 255.255.255.255:67
yiaddr: 0.0.0.0
transaction Id: 654
```

```
src: 223.1.2.5:67
dst: 255.255.255.255:68
yiaddr: 223.1.2.4
transaction Id: 654
DHCP server: 223.1.2.5
Lifetime: 3600 secs
```

Host Request for the IP address

```
src: 0.0.0.0:68
dst: 255.255.255.255:67
yiaddr: 223.1.2.4
transaction Id: 655
DCHP server: 223.1.2.5
Lifetime: 3600 secs
```

DHCP server ACK assignment

```
src: 223.1.2.5:67
dst: 255.255.255.255:68
yiaddr: 223.1.2.4
transaction Id: 655
DHCP server: 223.1.2.5
Lifetime: 3600 secs
```

server: 67, client: 68

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

32 bits

1 IP datagram structure 32

<table>
<tr><td>ver</td><td>IHL</td><td>TOS</td><td colspan="2">total length</td></tr>
<tr><td colspan="3">identification</td><td>flags</td><td>offset</td></tr>
<tr><td colspan="2">TTL</td><td>protocol</td><td colspan="2">header checksum</td></tr>
<tr><td colspan="5">source IP address</td></tr>
<tr><td colspan="5">destination IP address</td></tr>
<tr><td colspan="5">Transport segment
(Typically TCP or UDP)</td></tr>
</table>

32 bits

Internet Control Message Protocol(ICMP) datagram structure

<table>
<tr><td>ver</td><td>IHL</td><td>TOS</td><td colspan="2">total length</td></tr>
<tr><td colspan="3">identification</td><td>flags</td><td>offset</td></tr>
<tr><td colspan="2">TTL</td><td>0001</td><td colspan="2">header checksum</td></tr>
<tr><td colspan="5">source IP address</td></tr>
<tr><td colspan="5">destination IP address</td></tr>
<tr><td colspan="2">type</td><td>code</td><td colspan="2">checksum</td></tr>
<tr><td colspan="5">rest of ICMP data</td></tr>
</table>

protocol flag - not actually transport layer data

Type	Code	Description
8	0	echo request
0	0	echo respond
3	1	destination host unreachable
3	3	destination port unreachable
11	0	TTL expired
12	0	bad IP header

ping/traceroute

Link-State Algo

- Routers周期性broadcast link cost to each other
- Compute least cost path locally

Distance Vector Algorithms

Routers only know

- physically connected neighbours
- and link costs to neighbours

✓Decentralized ✓ Self-terminating

✓Iterative✓ Asynchronous

Bellman-Ford Equ: $d_x(y)=\min_i\{c(x, i) + d_i(y)\}$

Routing Information Protocol(RIP)

- Implements DV algorithm - hop count = cost
- Entries in the routing table - all subnet masks
- Exchange routing table per 30s (over UDP #52)
- If no update from a neighbour for 3 mins, assue neighbour died

Frequency Shift Keying is limited by bandwidth, if f1>f0, bandwidth = f1-f0

Symmetric Key Cryptography

DES (Data Encryption Standard)

AES (Advanced Encryption Standard)

Public Key Cryptography

RSA (Rivest, Shamir, Adelson algorithm)

Cryptographic Hash Functions

input m -> fixed size str, msg digest H(m)

MD5 (Message Digest) and SHA-1 (Secure Hash Algorithm)

Message Authentication Code (MAC)

a key is used as part of MD generation

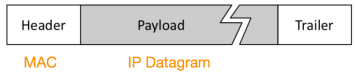
$H(m+S)\oplus m$

HmacMD5, HmacSHA1, HmacSHA256

cannot prove MAC is produced by who

Digital Signature $K_A^-(H(m))\oplus m$

Link Layer



- Framing
- Link Access Control
- Reliable Delivery (often used on error-prone links)
- Error Detection &/ Correction

Error Detection & Correction

ChkSum: TCP/UDP/IP - complex, slow

Parity Checking

if no error, even number of 1s, XOR return 0

Cyclic Redundancy Check(CRC) - link layer

$D2^r + R = kG$ - Generate CRC of r bits

- D: data bits, viewed as a binary number
- G: a chosen generator of $r + 1$ bits
- R: remainder of $D2^r/G$

Receiver knows G, divides $D2^r + R$ by G - 0

Multiple Access Control (MAC)

distribute algo - how nodes share channels

no out-of-band channel signaling

Channel partitioning protocols

- Channel is shared fairly and efficiently if **most** nodes have data to send

Time/Frequency/Code-DMA

Taking turns protocols

Polling from master node/token passing

- Efficient at **both** low and high load
- Single point of failure
- Polling overhead
- (token: Latency)

Random access protocols

- Efficient at **low** load: single node can fully utilize channel
- High collision rate at high load: wasted channel time

Carrier Sense Multiple Access(CSMA)

sense channel before transmission

CSMA/CD

stop talking once collision is detected

Retransmit after a random amount of time

Ethernet - at least 64 bytes

CSMA/CA

Request to send/clear to send (RTS/CTS)

ACK for received frame

Retransmission Algorithm

Slotted ALOHA - the more collision you have, the lower priority you are

CSMA/CD - At nth consecutive collision: $m = \min(n, 10)$, Pick $k \in \{0,1,\ldots,2^{m-1}\}$, wait $512\times k$ bit-time

MAC Address	Interface	TTL
FA:CE:BD:0C:11:FD	1	60
66:23:C6:1D:FE:32	2	60
15:00:2A:F1:CE:A1	3	60

preamble	dst MAC	src MAC	type	data	CRC
	header			IP datagram	trailer

mble:

Preamble:

- 7 bytes of 10101010 followed by 10101011
- Used for hardware clock synchronization

MAC Addresses:

- 6 bytes each

Type:

- Indicate higher level protocol
- Mostly IP, but others include Novell IPX, Apple Talk, etc

CRC:

- Error detection and correction

Router	Switches
Check IP address	Check MAC address
Store-and-forward	Store-and-forward
Compute routes to destination	Forward frame to outgoing link or broadcast

MAC Address - 6 bytes, permanent

- Every adapter (NIC) has a MAC address
- Used to send & receive link layer frames

Address Resolution Protocol (ARP)

Each IP node (host, router) has an ARP table

Stores mapping of IP address to MAC address of other nodes **in the same subnet**

IP address	MAC address	TTL
IP_B	MAC_B	60
IP_C	MAC_C	60
IP_D	MAC_D	60

Sending frame in the same subnet (A to B)

- A knows B's MAC address from its ARP table

Construct frame with B's MAC address as destination address

- A doesn't know B's MAC address

Broadcast ARP query packet with B's IP address (broadcast address: FF:FF:FF:FF:FF:FF); B replies with MAC address; A caches it in ARP table

- Sending fram to different subnet (A to X)

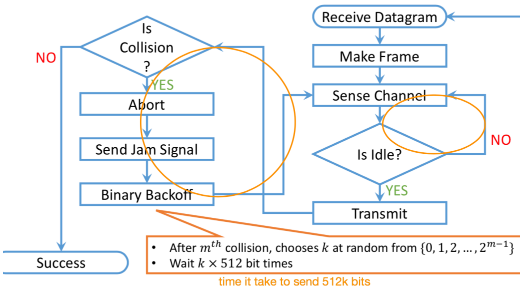
Send with MAC_X; Send to router MAC_{R1} - IP_X; MAC_{R1} -> IP_X; MAC_{R2}

Local Area Network (LAN) - geographical area

Ethernet - wired (dominant)

Data Delivery Service

- Connectionless - no handshaking
- Unreliable - no ACK/NAK sent
- MAC - CSMA/CD



- After m^{th} collision, chooses k at random from $\{0, 1, 2, \ldots, 2^{m-1}\}$
- Wait $k \times 512$ bit times

time it take to send 512k bits

Ethernet Switch

- Store and forward Ethernet frames

Examine incoming frame's MAC address

Selectively forward frame to one-or-more outgoing links

- Transparent to hosts: No IP address
- Star topology, Full duplex, Buffered

Non-Return to Zero (NRZ) (0, 1)
Return to Zero (RZ) (-1, 0, 1) bit slip (x)
Higher bandwidth required
Non-Return to Zero-Level (NRZ-L)
Bit-0: -V Bit-1: +V
Non-Return to Zero-Invert (NRZ-I)
Bit-0: no inversion Bit-1: inversion
Bit-slip: sender has a faster clock than the receiver - may have inversion in the middle, the longer you send all 1s, the more problematic it may be
Manchester bit slip (x)
Bit-0: Bit-1:
Differential Manchester bit slip (x)
Bit-0: no inversion Bit-1: inversion

Nyquist Bit-Rate Formula - ideal **noise-less**

$2B \times \log_2 L$

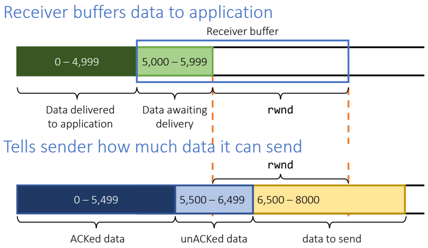
- B is the channel bandwidth
- L is the number of signal levels

Shannon Channel Capacity - **noisy** channel

$B \times \log_2 (1 + \text{SNR})$

- SNR is the measure of the strength of signal over noise

TCP Flow Control



Extending host-to-host delivery to process-to-process delivery is called transport-layer multiplexing and demultiplexing
Integrity checking - by including error-detection fields in their segments' headers

UDP header	source port	destination port
	length (inc. header)	checksum
	Application data (message)	

Length: the number of bytes in the UDP segment = header + data

UDP checksum
At the sender side: 1s complement of the sum of all the 16-bit words in the segment, any overflow during the sum is wrapped around
At the receiver side: all 16-bit words are added + checksum
It all 1 -> no errors

Utilization
The fraction of time the link is actually being used
throughput = $L/(RTT+d_{trans})$
$U_{sender} = d_{trans}/(RTT+d_{trans})$
Pipelining has the following consequences for reliable data transfer protocols:

- The range of seq numbers must be increased
- The sender and receiver sides of the protocols may have to buffer more than one packet - buffer packets that have been transmitted but not yet ACKed
- The range of seq numbers needed and the buffering requirements will depend on GBN or selective repeat

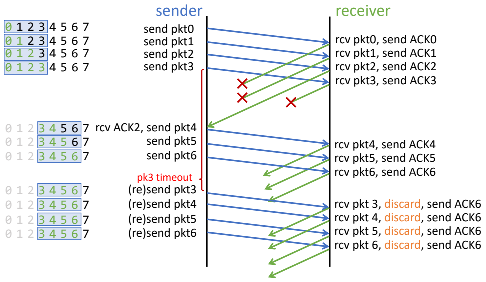
GBN - sliding-window protocol
ACK n means all packets ≤ n have been received
Constrained to have no more than N of unACKed packets in the pipeline
Keep track of N unACKed packets. Timer for oldest unACKed packet. On timeout, retransmit all packets.
GBN Sender

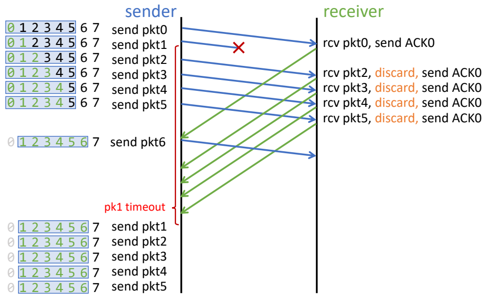
- can have up to N unACKed packets in pipeline.
- insert k-bits sequence number in packet header.
- use a "sliding window" to keep track of unACKed packets.
- keep a timer for the oldest unACKed packet.
- timeout(n): retransmit packet n and all subsequent packets in the window.

GBN Receiver

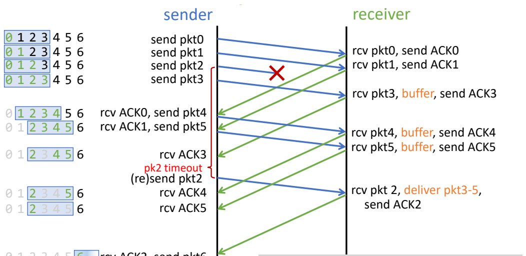
- only ACK packets that arrive in order.
  - simple receiver: need only remember expectedSeqNum
- discard out-of-order packets and ACK the last in-order seq. #.
  - Cumulative ACK: "ACK m" means all packets up to m are received.

Go-back-N in action



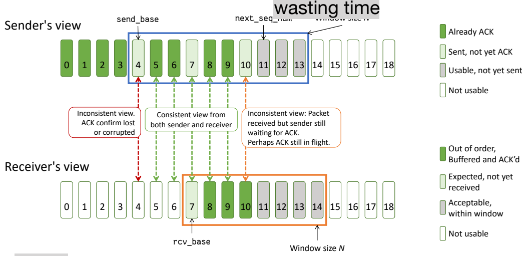


SR
One timer per packet, receiver needs a buffer
A window size of N will be used to limit the number of outstanding, unACKed packets in the pipeline
The sender will have already received ACKs for some of the packets in the window
Sender
data from above:
- if next available seq # in window, send pkt
timeout(n):
- resend pkt n, restart timer
ACK(n) in [sendbase, sendbase+N]
- mark pkt n as received
- if n is smallest unACKed pkt, advance window base to next unACKed seq. #
Receiver
pkt n in [rcvbase, rcvbase+N-1]
- send ACK(n)
- out-of-order: buffer
- in-order: deliver (also deliver buffered, in-order pkts), advance window to next not-yet-received pkt
pkt n in [rcvbase-N, rcvbase-1]
- ACK(n)
otherwise: - ignore



this window size cannot be too big, otherwise receiver cannot buffer the sent packets, which is wasting time

Sender and Receiver W



TCP
not run in intermediate network elements (routers & link-layer switches)
Maximum segment size(MSS): the maximum amount of application-layer data (data field) in the segment, **not including headers**
Maximum transmission unit(MTU): the length of the largest link-layer(which has physical limitation) frame that can be sent by the local sending host (on all links from src to des)

	32 bits		
	1	16 17	32
TCP header	source port		destination port
	sequence number		
	acknowledgement number		
	length	URG ACK PSH RST SYN FIN	receive window
	checksum		
	options		
	Application data (message)		

Receive window: 16-bit, flow control, used to indicate the number of bytes that a receiver is willing to accept
Header length: 4-bit, specifies the length of the TCP header in 32-bit words. If the options field is empty, TCP header is 20 bytes
Options field: optional and variable-length, used when a sender and receiver negotiate the MSS or as a window scaling factor for use in high-speed networks.
Flag field: 6 bits - ACK bit; RST, SYN, FIN bits - connection setup and teardown; CWR, ECE bits - explicit congestion notification; PSH bit - the receiver should pass the data to the upper layer immediately; URG - indicate that there is data in this segment that the sending-side upper-layer entity has marked as 'urgent'
Urgent data field: 16-bit, the location of the last byte of this urgent data
Seq numbers and ACK numbers
TCP views data as an unstructured, but ordered, stream of bytes
The ACK number that Host A puts in its segment is the seq number of the next byte Host A is expecting from Host B
Cumulative acknowledgments

RTT estimation and timeout
exponential weighted moving average(EWMA)
Timeout > RTT
$RTT\varepsilon=(1-\alpha)\cdot RTT\varepsilon+\alpha\cdot RTTs$ ($\alpha = 0.125$)
Setting retransmission time out (RTO)
$RTT\text{dev}=(1-\beta)\cdot RTT\text{dev}+\beta\cdot|RTTs-RTT\varepsilon|$ ($\beta = 0.25$)
RTO Interval is set to $RTT\varepsilon +4 \times RTT\text{dev}$

a) What is the URL of the document requested by this browser?

www.comp.nus.edu.sg/~cs2105/demo.html

d) What is the IP address of the host on which the browser is running?

IP address is not shown in HTTP message. One would be able to get such information from socket.

e) What type of browser initiates this message? Why is the browser type useful in an HTTP request message?

Mozilla. The browser type information sometimes is useful for server to send different versions of the same object to different types of browsers.

---

a) Was the server able to successfully find the document or not?

The status code 200 and the phrase OK indicate that the server was able to locate the document successfully.

b) What time did the server send the HTTP response message?

The HTTP response message was formed on Tuesday, 20 Jan 2015 10:08:12 Greenwich Mean Time.

c) How many bytes are there in the document being returned?

There are 73 bytes in the document being returned.

d) Did the server agree to a persistent connection?

The server agreed to a persistent connection, as indicated by the header field 'Connection: Keep-Alive field'.

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.

Network edge: end hosts, servers
Network core: ISPs, Routers
hosts = client+server
Unshielded twisted pair copper wire - LANs
Coaxial cable - guiede shared medium
Fiber optics - long-distance

---

a) Non-persistent HTTP with no parallel TCP connections?

$3 \times D_{DNS} + (5 + 1) \times 2 \times D_{Web}$

b) Non-persistent HTTP with the browser configured for five parallel connections?

$3 \times D_{DNS} + 2 \times D_{Web} + 2 \times D_{Web}$

Need to fetch HTML file first ($2 \times D_{Web}$). Subsequently the rest 5 objects can be fetched in parallel each using a TCP connection ($2 \times D_{Web}$).

c) Persistent HTTP with pipelining?

$3 \times D_{DNS} + 2 \times D_{Web} + D_{Web}$

Need to fetch HTML file first ($2 \times D_{Web}$). The rest 5 objects can be fetched through the same TCP connection in parallel – no RTT for TCP handshake needed.

---

5. RIP is an application layer problem. How does it implement network-layer functionality?

Ans: RIP uses a transport-layer protocol (UDP) on top of a network layer protocol (IP) to implement network-layer functionality (e.g., a routing algorithm).

6. Two hosts A and B participate in a peer-to-peer
file sharing application and need to connect to each other. Both A and B, however, are behind NATs.
Devise a technique that will allow A to establish a TCP connection with B without application specific NAT configuration, if...

(a) the NAT router uses a simple, predictable, algorithm to allocate a public port number for mapping to the local/private port number. Ans: It is not possible to devise such a technique. In order to establish a direct TCP connection between A and B, either must initiate a

---

There are many nodes in a shared medium network and most nodes are likely to transmit frequently. Which of the following multiple access protocol(s) is (are) suitable? (1) TDMA; (2) CSMA; (3) Token passing.

TDMA and token passing are suitable because there is sufficient work to do to utilize the "fixed" resources allocated. CSMA is not because many nodes competing for the shared channel can result in lots of collision. Utilization will be low.

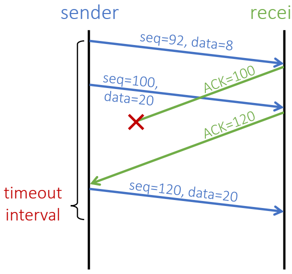
Application Layer - message - end system
- HTTP, SMTP, FTP, POP, REST, BT
Transport Layer - segment - app endpoints(process)
- TCP, UDP
Network Layer(IP layer) - datagrams - host and routers
- IP, ICMP, routing protocols(RIP, OSPF, BGP, Path selection)
<u>determine which output link/the route should follow</u>
Link Layer - frames - node
- Ethernet, WiFi, DOCSIS protocol
Physical Layer - individual bits within the frame - wire/air
- QAM, OFDM, TDM, NRZ, Manchester
IP: 32-bit   port number: 16-bit (1-1023 reserved)

Internet - packet switching network
DHCP -> UDP
RIP -> UDP on top of IP
TCP -> use service provided by IP
HTTP -> TCP as transport protocol
DNS -> UDP #53

- Finer application-level control over what data is sent and when
- No connection establishment
- No connection state: Can support many more active clients when the application runs over UDP rather than TCP
- Small packet header overhead

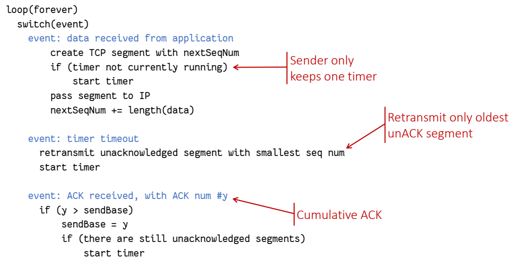
TCP: 20 bytes of header overhead in every segment
UDP: 8 bytes

TCP Timeout/Retransmission



Cumulative ACK prevents rxmt

TCP Sender Events (simplified)



```
loop(forever)
  switch(event)
    event: data received from application
       create TCP segment with nextSeqNum
       if (timer not currently running)
           start timer
       pass segment to IP
       nextSeqNum += length(data)

    event: timer timeout
      retransmit unacknowledged segment with smallest seq num
      start timer

    event: ACK received, with ACK num #y
      if (y > sendBase)
          sendBase = y
          if (there are still unacknowledged segments)
              start timer
```

TCP Receiver Events