COMP3331 topics.

Not asked:

- Derive complex probability computations
- Specifics of access networks
- Networks under attack
- History
- HTTPS
- HTTP/2
- IMAP, POP
- DNSSEC, DNS over TLS. DNS over HTTP
- Socket Programming
- Complex checksum computations
- Congestion control (for final)
- No programming
- No use of tools from lab exercises

Asked:

- Packet Switching
- Delay, loss, throughput
- Protocol layering
- Basic questions on probability

Applications

- Principles
- Web/HTTP
- E-mail (conceptual)
- DNS
- P2P, BitTorrent, DHT
- Video Streaming and CDN (basic eye level)

Transport Layer

- Principles
- Sockets
- UDP (basic)
- RDT (1 through 3) GBN, SR
- TCP

Basic Principles

Packet Switching

Data is sent as chunks of formatted bits (packets). They consist of header and payload (seen through packets shown in wireshark). Header has instructions on how to handle packets. Payload is data carried.

Delay, loss, throughput

Delay and loss occur when packets are queued in router buffers. It happens when the arrival rate exceeds the output link capacity. There are 4 sources of packet delay, nodal processing, queuing, transmission and propagation delay. Together they form the overall formula for nodal delay

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

d_{proc}

- Nodal Processing. Involves checking bit errors within the packet

d_{queue}

- Queue delay. Time taken waiting for transmission at the output such that when,
  A = Arrival rate (packets/sec),
  L = Packet Length (bits),
  R = bandwidth,
  then, $d_{queue} = LA/R$
  where $d_{queue} \simeq 0$ is small, $d_{queue} \rightarrow 1$ is greater and $d_{queue} > 1$ is infinity.

d_{trans}

- Transmission delay. Time taken to push all the packet's bits into the wire for propagation such that,
  L = packet length (bits), (times by 8 to get bits from bytes)
  R = bandwidth(Mbytes/sec),
  then, $d_{trans} = L/R\ (microseconds)$

d_{prop}

- Propagation delay. The time taken for a packet to reach a destination such that when,
  D = length of physical link in metres,
  S = propagation speed in medium,
  $d_{prop} = D/S\ (in\ seconds)$

Protocol layering

'The Stack'

1. Application. Supporting network applications
    - FTP, SMTP, HTTP
2. Transport. Ways to process data transfer and move it across the network.
    - TCP, UDP
3. Network. The actual network and how everything is "addressed and located"
    - IP, Routing protocols
4. Link. How data is transferred within the network and 'neighbours'
    - Ethernet, 802.111, PPP
5. Physical. How the network is physically built. Bits 'on the wire'.
    - Fiber, Copper

Each level is dependent on the level below with everything being built on the 'physical' level.

LAYERS COVERED BY MIDTERM. APPLICATION AND TRANSPORT.

The Application Layer

Principles Of Network Applications

Interprocess Communication

- Process can only talk to each other via IPC
- IPC only on shared memory on a single machine

Sockets

- Sockets allow processes to communicate even if they are on a different machine.
- The end point of a socket consists of an IP address and a port in which the process is run from which allows it to be identified by a transport network like TCP
- To receive messages a process must have an identifier which includes both the IP address and port numbers of the associated process.

Client Server Architecture

The server

- Exports requests/response interface
- Long running process that waits for requests from clients

    - Has a welcome socket open to welcome new requests
- When requests are received, it carries them out
- Has a permanent IP address

The client
- Short running process that sends requests to servers
- Regarded as the ‘user’ side of the application
- Initiates communication
- May have a dynamic IP address
- Does not communicate directly with other clients

P2P Architecture
There is no long running server but rather, clients directly communicate with each other.
Its benefits are that it is scalable, fast, reliable and widely distributed.
Its shortcomings are that it can be uncertain and unstable at times due to clients possibly being unable to sustain reliable connections

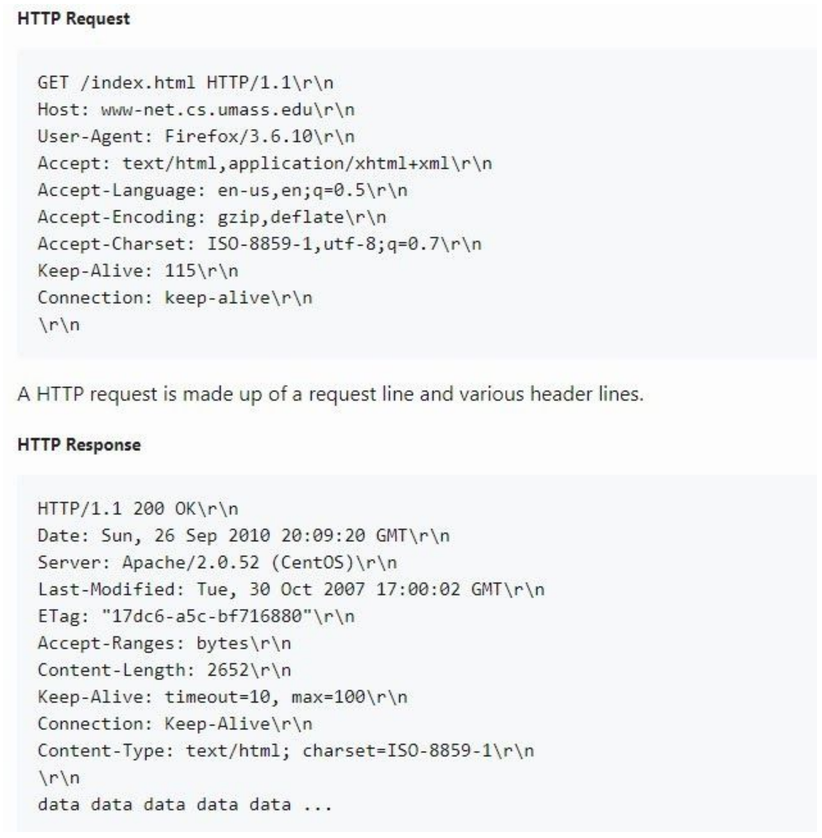
Web & HTTP
Uniform Resource Locator (URL)

protocol://host-name[:port]/directory-path/resource

- protocol: http, ftp, https, smtp etc
- host-name: DNS name, IP address
- port: defaults to protocol’s standard port; e.g. http: 80 https: 443
- directory-path: hierarchical, reflecting file system
- resource: identifies the desired resource

Hypertext Transfer Protocol (HTTP)
- Is the web’s application layer. Based on a client/server architecture, uses TCp and it’s stateless.
- HTTP is all text. This means that a user can decipher its contents using programs such as wireshark. This makes it user friendly, but not efficient. (high level)

Example HTTP request and response



HTTP Request

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

A HTTP request is made up of a request line and various header lines.

HTTP Response

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```

Status codes
Status codes appear at the beginning of responses to communicate statuses with the response. E.g 404 not found, 200 OK

Verbs
HTTP consists of verbs that indicate to the server, the type of response requested. E.g GET, POST, HEAD etc.

HTTP 1.0 has no persistent connections. This means that new requests must open new connections
HTTP 1.1 has pipelining which means that multiple requests can be sent on the same connection
HTTP 2.0 is the same as 1.1 but more elegant

Email
In an email system, there are 3 main components

- User agents. The ‘user client’ which is used to compose, edit and read emails

- Mail servers. The ‘server’ which is used to queue outgoing emails and hold incoming emails.
- Simple Mail Transfer Protocol (SMTP). The “Transport” used to move emails between mail servers. Implements TCP to exchange data between servers.

SMTP

Uses TCp to reliably transfer messages. Utilises port 25

There are 3 stages of a mail transfer using SMTP

1. Handshaking
2. Transfer of Data/Messages
3. Closure.

All SMTP messages are in 7bit ASCII

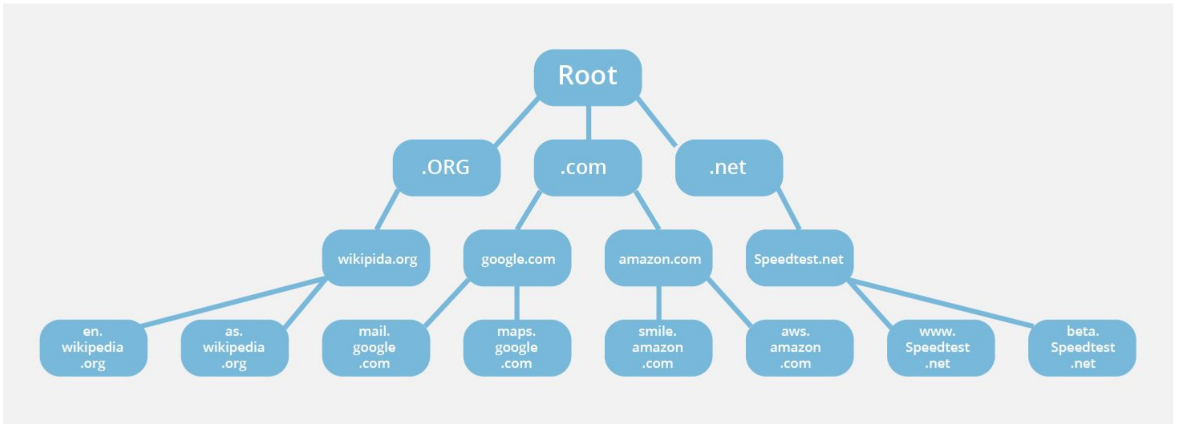
It uses CRLF.CRLF to determine the end of a message

Other protocols such as POP, IMAP and HTTP(S) are used to pull user emails onto the user agent.

DNS and DNS Servers

Domain Name System (DNS) is a distributed database implemented in a hierarchy of many name servers. DNS uses TCP for Zone transfer and UDP for name queries. UDP can be used to exchange small information whereas TCP must be used to exchange information larger than 512 bytes

An example hierarchy:



Hierarchy

- At the top of the tree are the root name servers which are distributed around the world.
- Top level domains (e.g .com .net)
- Domains are in a leaf to root path (wikipedia.org

- Each domain is responsible to its level below. (en.wikipedia.org, as.wikipedia.org)

DNS provides services such as

- Hostname to IP address translation
- Host aliasing
- Mail server aliasing
- Load distribution

Authoritative DNS Servers

Are an organisation's own DNS server(s), providing authoritative hostname to Ip mappings for an organisation's named hosts. They are maintained by the organisation or a provider

Local DNS Servers

Local DNS name servers do not belong to the hierarchy but rather they belong to ISPs and are configured with a local DNS server address or learn servers via Dynamic host configuration protocol (DHCP).

It acts like a middle man or proxy that forwards user requests into the hierarchy.

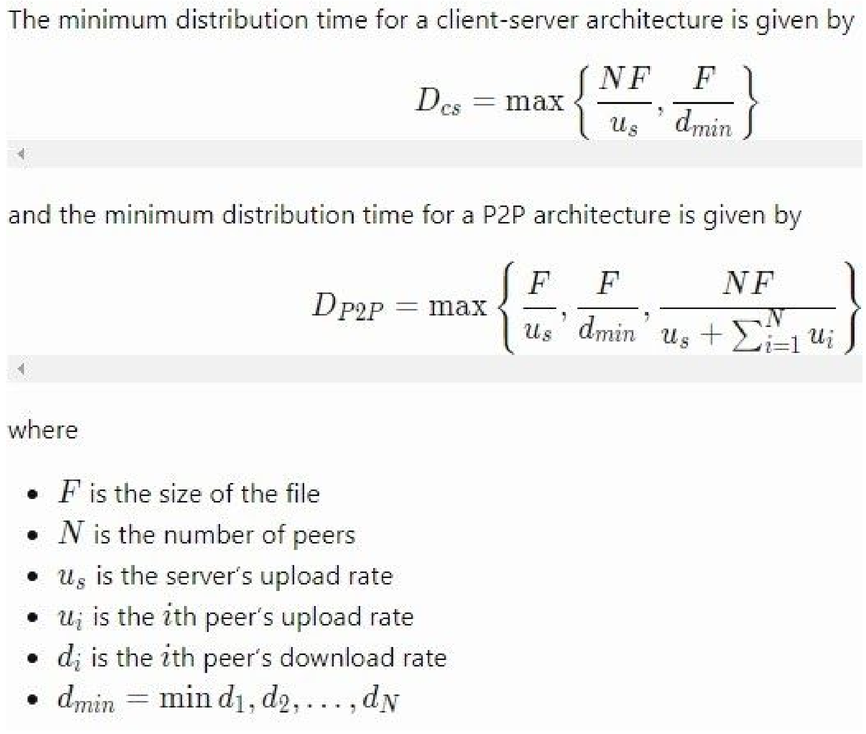
DNS Records

A DNS database contains resource records (RR). THe format of a RR is (name, value, type, ttl)
The type of a RR can be

- A
    - Name is the hostname
    - Value is the IP address
- CNAME
    - Name is an alias for some canonical name
    - Value is the canonical name
- NS
    - Name is the domain
    - Value is the hostname of the authoritative server
- MX
    - Value is the name of the mailserver associated with name

P2P Applications

P2P architecture involves end systems (usually users) communicating with each other directly. There is no ‘always on server’ and peers are intermittently connected and exchange IP addresses. A good example of this application being used is in GTA ONLINE.



The minimum distribution time for a client-server architecture is given by

$$D_{cs} = \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$$

and the minimum distribution time for a P2P architecture is given by

$$D_{P2P} = \max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\}$$

where

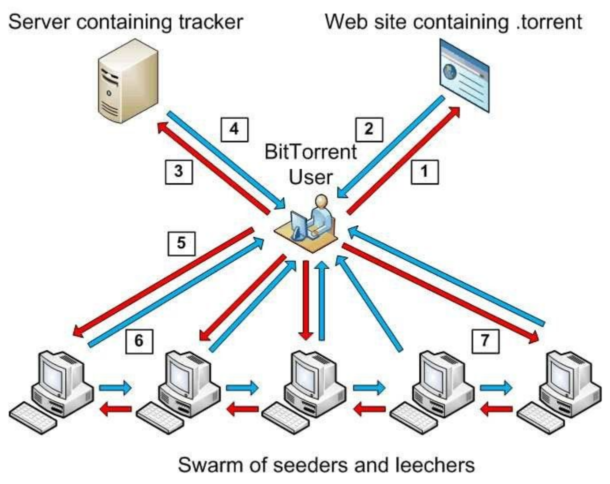
- F is the size of the file
- N is the number of peers
- u_s is the server's upload rate
- u_i is the ith peer's upload rate
- d_i is the ith peer's download rate
- $d_{min} = \min d_1, d_2, \ldots, d_N$

BitTorrent

Bittorrent is a P2P file distribution protocol where files are divided into 256KB chucks and distributed by peers that send/receive said chunks.

A torrent is a group of peers exchanging chunks of a file. A .torrent file contains addresses of trackers for the file it represents, as well as a list of chuks and their cryptographic hashes.

A tracker is a server that tracks peers participating in a torrent.


Server containing tracker
Web site containing .torrent
BitTorrent User
1
2
3
4
5
6
7
Swarm of seeders and leechers

Requesting chunks
At any given time, different peers have different subsets of file chunks. Peers request chunks from other peers in rarest first order.

Sending chunks
Peer *p* sends chunks to 4 other peers (default) that are sending chunks at the highest rate to *p*. Every 30 seconds, another peer is randomly selected to send chunks to known, known as ‘optimistic unchoking’.

Distributed Hash Table (DHT)
A DHT is a P2P database containing key value pairs representing tracker information. This removes the reliance on a centralised tracker.

Video Streaming and CDNs
Video
Is a sequence of images displayed at a constant frame rate where each image (frame) is represented by an array of pixel values.

The constant bit rate (CBR) is the fixed video encoding rate. The variable bit rate (VBR) is determined by video encoding rate changing as an amount of spatial and temporal coding changes.

DASH
Dynamic, Adaptive Streaming over HTTP (DASH) is a method used to stream multimedia from a server to a client.

The server

- Divides the video into chunks
- Each chunk is stored and encoded at different rates
- A manifest file provides URLS for different chunks

The Client

- Periodically measures server-to-client bandwidth
- Consults the manifest and requests one chunk at a time based on current bandwidth.
- Is responsible for choosing when to request the chunk, what encoding rate to request and where to request the chunk from (what URL).

CDNs

Content delivery networks (CNDs) are used to store copies of multimedia at various different geographical server locations. It is mainly used for its efficiency and speed, security and reduced bandwidth costs.

Sockets with UDP & TCP (Not tested in midterm)

(Insert programming here)

Transport Layer

Services

The transport layer provides communication between app processes running on different hosts. They run in end systems where

- The sender breaks messages into segments and passes them into the network layer
- The receiver reassembles segments into messages and passes them into the application layer

Multiplexing and Demultiplexing

Sender multiplexing

- Handles data from multiple sockets
- Adds transport header

Receiver Demultiplexing

- Uses header information to deliver received segments to correct socket\

Connectionless Demultiplexing
When the host receives a UDP segment, it checks the destination port number in the segment and directs the segment to the socket with that port number

Connection oriented Demultiplexing
The TCp socket is identified by a 4-tuple (sourceIP, sourcePort, destIP, destPort). The receiver uses all four values to direct segments to appropriate sockets. The server may support many simultaneous TCP
sockets where each socket is identified by its own 4-tuple.

UDP
Is a ‘bare bones transfer protocol. During transport, UDP segments may get lost or may be delivered out of order. UDP is connectionless meaning there is no handshaking between sender and receiver and that each segment is handled independently of others.

A UDP segment header which is usually 8 bytes in size consists of

- A source port
- Destination port
- Length in bytes with header
- Checksum (optional)
- Payload

Checksum
The goal of a checksum in a UDP is to detect errors in the transmitted arguments

The sender

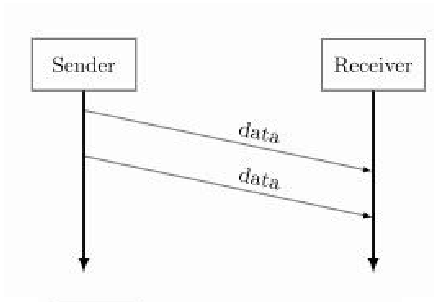
- Treats segment contents including header fields, as a sequence of 16-bit integers
- Computes a checksum using addition (one’s complement sum) of segment contents
- Puts the checksum value into the UDP checksum field

The receiver

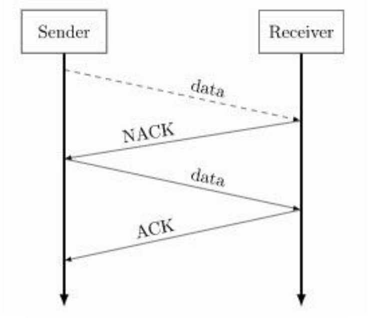
- Adds the received contents together as 16 bit integers
- Adds the value computed to the checksum
- If the result is not 1111 1111 1111 1111, there are errors.

Reliable Data Transfer (RDT)
The problem of being able to transfer data over the internet reliably.

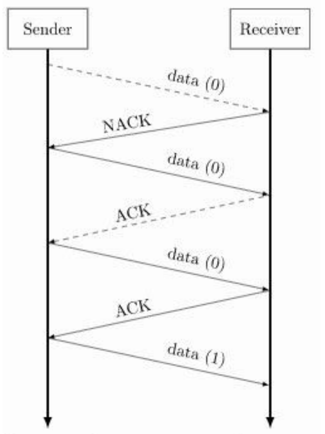
RDT 1.0
There is a channel that is perfectly reliable, it has no bit errors and no loss of packets. The transport layer does nothing.



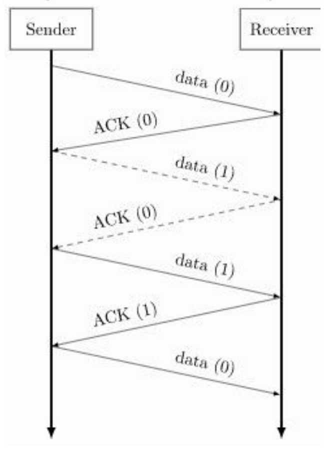
RDT 2.0
The underling channel may flip bits in a packet. To recover from errors the receiver sends ACKs, acknowledging that the packer was received ok, or NACKs to communicate that the packet had errors. If a NACK is received by the sender, the packet is retransmitted. RDT 2.0 has error detection through feedback (ACKs NACKs)



RDT 2.1
The sender adds a sequence to the packet. The two sequence numbers are used throughout transmission which are 0 and 1. The sender must also check if the (N)ACKs received are corrupted. The receiver must check if a received packet is a duplicate using packets sequence numbers.

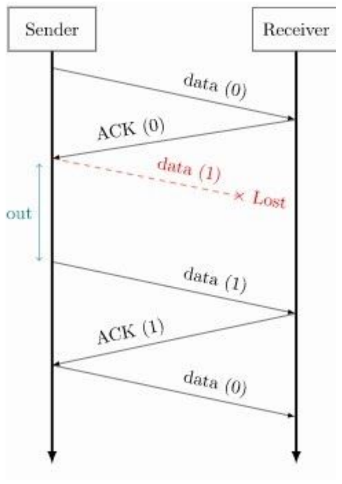


RDT 2.2
Functionally is the same as RDT2.1, except there are no NACKs. Instead the receiver sends an ACK for the last packet received. The receiver must explicitly include the sequence number of the packer being ACKed. A duplicate ACK at the sender results in the same action as NACK, retransmission.



RDT 3.0

We assume that the underlying channel can also lose packets. ACKs and data can be lost. We implement 2.2 but with a 'reasonable' time for timeout so that if no response is given within the time period, a new transmission is sent. If the transmission is just delayed, ACK(x) will be able to handle such events if a duplicate response is sent.



Pipelined Controls (GBN & SN)

Pipelining allows the sender to send multiple yet to be acknowledged packets. To achieve this, the range of sequence numbers must be increased, and there must be buffering at the sender and/or receiver.

The utilisation U of the link when a pipelined protocol with window size N is used is given by

$$U = N \frac{\frac{L}{R}}{\frac{L}{R} + RTT}$$

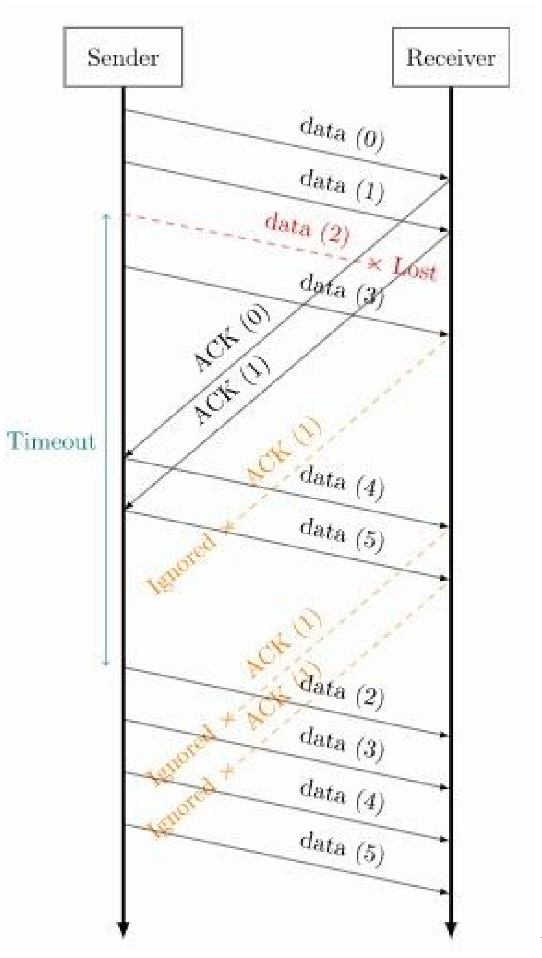
Where:

- L is packet length (bits)
- R is bandwidth
- RTT is real time transmission
- N is number of packets in window

Go-Back-N (GBN)

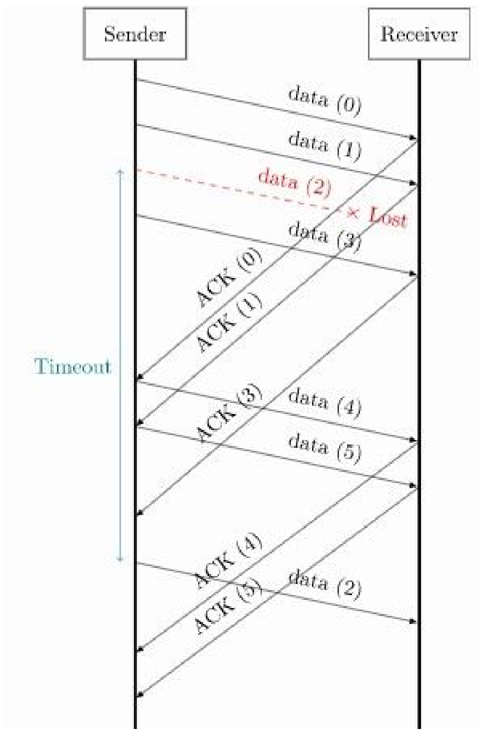
- The sender can have upto N un**ACK**ed packets in the pipeline
- The sender has a single timer for the oldest un**ACK**ed packet, and when the timer expires, the sender will retransmit all un**ACK**ed packets.
- There is no buffer available at the receiver and out of order packets are discarded
- The receiver only sends a cumulative ACK and doesn't ACK the new packet if there's a gap
- Sender window size < 2^n where n is the number of bits in the sequence number space

GBN example

- 

,

Selective repeat



- The sender can have up to N un**ACK**ed packets in the pipeline
- The sender maintains a timer for each un**ACK**ed packet, and when the timer expires, the sender will retransmit only that timed packet
- The receiver has a buffer and can accept out of order packets
- The receiver sends an individual ACK for each packet
- Sender window size ≤ 1/2 of the sequence number space

TCP

Features

- Point to point (NOT P2P)
    - One sender, one receiver
- Reliable in order byte stream
- Pipelined
    - TCP congestion and flow control set window size
- Full duplex
    - Bidirectional data flow in same connection
    - Data can be sent back and forwarth on the same connection
- Connection oriented
    - Handshaking initiates sender and receiver state before data exchange
- Flow controlled
    - Sender will not overwhelm receiver

A TCP segment header, which is 20 bytes in size contains

- Source port
- Destination port
- Sequence number
- Acknowledgement number
- Receive window
- Header length
- Checksum
- Options
- Data

An entire TCP segment is no more than the Maximum Segment Size (MSS).

TCP Timeout

$Timeout\ Interval = Estimated\ RTT + 4 \times DevRTT$

$EstimatedRTT = (1 - \alpha) \times EstimatedRTT + (\alpha \times SampleRTT)$

- Typically $\alpha = 0.125$
- Exponential weighted moving average

- Influence of past sample decreases exponentially fast

$SampleRTT$ is the measured time from segment transmission until ACK receival

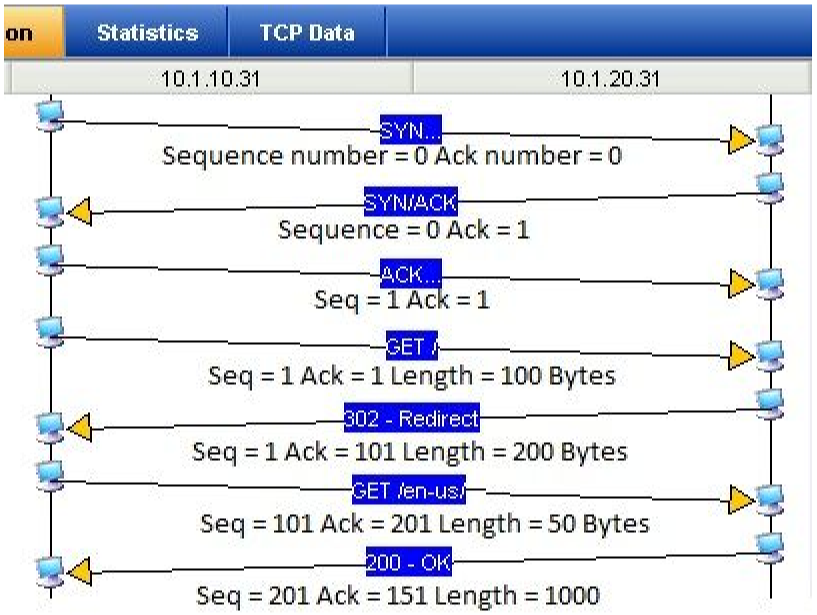
$DevRTT = (1-\beta) \times DevRTT + \beta \times | SampleRTT - EstimatedRTT$

- Typically $\beta = 0.25$

Connection management

1. A send a SYN (synchronise) packet to tell B it wants to open a connection
    a. Initiates a handshake
    b. Starts with some ISN (initial sequence number)
2. B sends a SYN-ACK packet to tell A it is ready to receive
    a. Returns the handshake
    b. Will contain an ACK number with the value of A's ISN plus 1, and an ISN for B
3. A sends an ACK packet to B to acknowledge B's SYN-ACK packet
    a. Shakes back to confirm yes
    b. Contains ACK number with the value of B's ISN plus 1, and a sequence number of A's ISN plus 1
4. **Data is sent between A and B**
    a. Acknowledgement number is length of bytes of TCP segment + sequence number
5. When completed, B sends a FIN packet to A
6. A sends an ACK packet to B with an ACK number of B's sequence number plus 1, and then a FIN packet to B
7. B sends an ACK packet to A with an ACK number of A's sequence number plus 1

An example of data exchange between A and b


on
Statistics
TCP Data
10.1.10.31
10.1.20.31
SYN...
Sequence number = 0 Ack number = 0
SYN/ACK
Sequence = 0 Ack = 1
ACK...
Seq = 1 Ack = 1
GET /
Seq = 1 Ack = 1 Length = 100 Bytes
302 - Redirect
Seq = 1 Ack = 101 Length = 200 Bytes
GET /en-us/
Seq = 101 Ack = 201 Length = 50 Bytes
200 - OK
Seq = 201 Ack = 151 Length = 1000

Principles of congestion control

Congestion can be thought of as "too many sources sending too much data too fast for a network to handle". It is different from flow control. As a result of congestion, packet loss and long delays can occur.

Congestion

- Increases latency
- Increases loss rate
- Increases retransmissions (many unnecessary due to latency)
    - Thus increases congestion even more

There are two approaches to managing congestion

- End to end congestion control
    - No explicit feedback from network
    - Congestion is inferred from the end-system's observed loss and delay
    - This is the approach taken by TCP
- Network assisted congestion control
- Routers provide feedback to end systems

TCP Congestion Control

A TCP connection has a window that controls the number of packets 'in flight'

The TCP sending rate is

$$rate \approx \frac{cwnd}{RTT} bytes/sec$$

Where:

- cwnd is the current window's number of bytes that can be sent without overflowing routers
- RTT is the round trip time

To vary congestion control, $cwnd$ is varied to the rate at which bytes are sent and are 'in flight'

TCP Slow Start

When a TCP connection begins, the value of $cwnd = 1\ mss\ (Maximum\ Segment\ Size)$. Everything a transmitted segment is first acknowledged (i.e every RTT), and **cwnd is doubled**. In the slow start phase, the sending rate increases two fold until the first loss is experienced. The initial sending rate is slow but increases rapidly.

When a loss is experienced/timeout occurs, $cwnd$ is set back to 1 mss and the slow start phase is restarted again. The less packets lost 'in flight', the more 'congestion' is permitted. During this event, ssthresh is also set to previous cwnd/2.

When the value of $cwnd$ reaches *ssthresh*, the slow start phase ends and congestion avoidance begins.

Congestion avoidance

Upon reaching the congestion avoidance phase when $cwnd$ reaches ssthresh (slow start threshold value), $cwnd$ now only increases by 1 MSS instead of doubling.

It then behaves much like the slow start that when a timeout occurs/loss is experienced, $cwnd$ is set back to 1 and the value of ssthresh is set to cwnd/2

In the case that a loss event is indicated by a triple duplicate ACK, the value of $cwnd$ is halved instead of being set to 1. In this case, congestion avoidance stops and fast recovery begins

Fast recovery

- $cwnd$ is increased by 1 MSS every duplicate ACK received
- When an ACK arrives for a missing segment, TCP enters congestion avoidance again after deflating cwnd
- If a timeout occurs, TCP enters slow start phase, $cwnd$ is set to 1 and the value of ssthresh is set to $cwnd/2$

Additive increase multiplicative decrease (AIMD)

In AIMD, the sender increases the transmission rate (window size), probing for usable bandwidth until another congestion even occurs

Additive increase refers to increasing the $cwnd$ by 1 every RTT until a loss is detected
Multiplicative decrease refers to reducing $cwnd$ by half after every loss

A slow start threshold is also introduced and intialised to a large value. When $cwnd = ssthresh$, we convert to additive increase

TCP Flavours

- Tahoe
  - $cwnd = 1$ and enters slow start phase after a timeout or triple ACK
- Reno
  - $cwnd = 1$ on timeout
  - $cwnd = cwnd/2$ on triple ACK
- New Reno
  - TCP reno + improved fast recovery

END OF MIDTERM TESTED MATERIAL