Written by Luka Kerr on November 24, 2019

Introduction

What Is The Internet?

The internet is a large interconnection of different computer networks and devices.

It contains

- Millions of connected computing devices
  - Hosts, running network applications
- Communication links
  - Fiber, copper, radio, satellite
- Packet switches
  - Routers, switches
- Protocols
  - TCP, IP, HTTP, UDP...
- Standards
  - RFC, IETF

The internet can also be viewed as a *service* that provides functionality and programming interfaces to applications.

Protocols

A network protocol defines a format, order of messages sent and received among network entities, and actions taken on message transmission.

Network Edge

On the edge of networks are client and server hosts and wired and wireless communication links.

DSL

Digital Subscriber Line (DSL) uses existing telephone lines connected to a central DSLAM office to transmit data over DSL lines. The typical transmission rate for DSL is ~1Mbps upstream and ~10Mbps downstream, with rates decreasing the further away you get physically form the DSLAM.

Cable

Cable uses *frequency division multiplexing*, in which different channels are transmitted in different frequency bands. The typical transmission rate for cable is ~2Mbps upstram and ~30Mbps downstream.

FTT[NPC]

FTT[NPC] uses optical fiber cables all the way to the node, premises or curb. Typical transmission rate is ~30Mbps to ~1Gbps.

WAN

Wireless Access Networks connect end users to a router via a base station. There are two main types; wireless LANs and wide area wireless access. Wireless LANs are located locally and use a wifi protocol (802.11b/g/n/ac/ax), whereas wide area wireless access points are provided by telecommunication companies and provide internet to a large area.

Network Core

At the core of networks are a mesh of interconnected routers and switches.

There are two forms of switches networks

- **Circuit switching**: used in legacy telephone networks
- **Packet switching**: used in the internet

Circuit Switching

In circuit switching, the end to end resources are allocated to and are reserved for the transmission between the source and destination.

In circuit switching resources are dedicated and there is no sharing.

Circuit switching is not feasable as it

- **Is inefficient**
  - A dedicated circuit cannot be used or shared in periods of silence
  - It cannot adapt to network dynamics
- **Has a fixed data rate**
- **Requires maintenance of connection state**
  - Per communication state needs to be maintained which is a considerable overhead
  - This is not scalable

Packet Switching

Packet Switching

In packet switching, data is sent as chunks of formatted bits (packets). Packets consists of a *header* and *payload*. The header holds instructions for how to handle the packet, and the payload is the data being carried.

Network switches forward packets based on their headers, and each packet travels independently. No link resources are reserved in advance, instead packet switching leverages *statistical multiplexing*. Statistical multiplexing relies on the assumption that not all flows burst at the same time.

Internet Structure

- End systems connect to Internet via access ISPs
- Access ISPs in turn must be interconnected
- ISPs connect via Internet Exchange Points (IXPs)
- Regional networks may connect access ISPs to ISPS
- Content provider networks may run their own network to bring services and content close to end users

Delay, Loss, Throughput

Delay and loss occur when packets queue in router buffers. This occurs when the packet arrival rate to a link (temporarily) exceeds the output link capacity. There are four sources of packet delay, that form part of the overall formula for the nodal delay:

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

where

- d_{proc} is the nodal processing
  - Involves checking bit errors, determining output link
- d_{queue} is the queueing delay
  - The time waiting at the output link for transmission
  - $A =$ packet arrival rate (A packets/sec)
  - $L =$ packet length (bits)
  - $R =$ link bandwidth (bits/sec)
  - If $LA/R \simeq 0$ then the average queueing delay is small
  - As $LA/R \to 1$ then the queueing delay becomes large
  - If $LA/R > 1$ then the average queueing delay is infinite
- d_{trans} is the transmission delay
  - $L =$ packet length (bits)
  - $R =$ link bandwidth (bps)

    - Then $d_{trans} = L/R$
- d_{prop} is the propagation delay
    - $D =$ length of physical link
    - $S =$ propagation speed in medium
    - Then $d_{prop} = D/S$

The end to end (E2E) delay between a source and destination is the sum of all d_{nodal} along the path.

Circuit & Packet Switching E2E Delay

Making the assumption that the network has no packets currently in it ($d_{queue} = 0$), for **circuit switching** we can calculate the end to end delay using a general formula

$$E2E = t_{setup} + N(d_{prop} + d_{proc}) + d_{trans}$$

and similarly for **packet switching** we can calculate the end to end delay using a general formula

$$E2E = t_{setup} + N(d_{proc} + d_{trans} + d_{prop}) + (k - 1)(d_{trans})$$

where for both

- t_{setup} is the setup time of the network
- N is the number of hops (with $N - 1$ intermediate routers)
- d_{proc} is the processing delay
- d_{prop} is the propagation delay
- d_{trans} is the transmission delay
- k is the number of packets to be sent

`traceroute`

The program `traceroute` provides delay measurements from a source to a router along a path via the internet towards a destination.

For all i, `traceroute` will

- Send three packets that will reach router i on path towards destination
- Router i will return packets to sender
- Sender times interval between transmission and reply

Packet Loss

The queue preceding the link has a finite capacity. Packets arriving to a full queue will get

dropped, and may get retransmitted.

Throughput

Throughput is the rate that bits get transferred between a sender and received. Between a sender S and receiver R there are various other bottlenecks $B_0, B_1, \ldots, B_n$. The end to end throughput between S and R is $\min(S, B_0, B_1, \ldots, B_n, R)$. Usually, S or R is the bottleneck.

Protocol Layers

Internet Protocol Stack

- **Application**: supporting network applications
  - FTP, SMTP, HTTP...
- **Transport**: process to process data transfer
  - TCP, UDP
- **Network**: routing of datagrams from source to destination
  - IP, routing protocols
- **Link**: data transfer between neighboring network elements
  - Ethernet, 802.111, PPP
- **Physical**: bits 'on the wire'
  - Fiber, Copper

Each layer depends on the layer below, supports the layer above and is independent. Each layer's protocols have multiple versions, except for IP in which there is one unifying protocol.

The benefit of layering is that each layer provides a common abstraction for various network technologies.

In host machines, every layer is required, whereas on routers only the network layer and below are required.

Application Layer

Principles Of Network Applications

Interprocess Communication (IPC)

- Processes 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 different machines.

To receive messages a process must have an *identifier* which includes both the IP address and port numbers associated with the process on the host machine.

Client Server Architecture

The server

- Exports a well defined request/response interface
- Is a long lived process that waits for requests
- Upon receiving request, carries it out
- Has a permanent IP address

The client

- Is a short lived process that makes requests
- Is the 'user side' of the application
- Initiates the communication
- May have dynamic IP addresses
- Does not communicate directly with other clients

P2P Architecture

In P2P architecture there is no always on server, rather arbitrary end systems (peers) directly communicate.

P2P architecture is beneficial as it is self scalable, fast, reliable and distributed. It does have flaws, including the fact that it has state and action uncertainty.

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)

HTTP is the web's application layer. It is based on a client/server architecture, uses TCP and is stateless.

HTTP is all text, which makes the protocol simple to read, although is not the most efficient.

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 ...
```

A HTTP response is made up of a status line, various header lines and the data requested.

Status Codes

Status code appears in first line in server-to-client response message. They indicate the status of the response. For example a `200` status code means that the request succeeded, whereas a `404` status code means the resource was not found.

Verbs

HTTP consists of multiple verbs indicating to the server what type of request was sent. These include `GET`, `POST`, `HEAD`, `PUT`, `DELETE`, `TRACE`, `OPTIONS`, `CONNECT` and `PATCH`.

HTTP/1.0

- Non-persistent: one TCP connection to fetch one web resource
- Fairly poor page load time

Email

In email there are three major components: user agents, mail servers and simple mail transfer protocol (SMTP).

A **user agent** can be thought of as the mail client. It is responsible for composing, editing and reading mail messages.

A **mail server** is a mailbox containing incoming messages for a user, and a queue of outgoing messages.

The **SMTP** protocol is used between mail servers to send email messages.

SMTP

SMTP uses TCP to reliably transfer messages. It runs on port 25.

There are three phases of transfer:

1. Handshaking
2. Transfer of messages
3. Closure

All SMTP messages must be in 7 bit ASCII.

SMTP uses persistent connections and uses `CRLF.CRLF` to determine the end of a message.

There are also other email protocols including `POP`, `IMAP` and `HTTP(S)`.

DNS

Domain Name System (DNS) is a distributed database implemented in a hierarchy of many name servers. Hosts and name servers communicate to resolve names.

DNS provides many services

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

- Load distribution

Hierarchy

DNS is a heirarchy of many name servers. At the top of the tree are the **root** name servers, which are distributed geographically around the world.

- Top level domain servers are under the root names servers
- Domains are subtrees in a leaf-to-root path
- Each domain is responsible for its immediate level below

Authoritative DNS Servers

Authoritative DNS servers are an organisation's own DNS server(s), providing authoritative hostname to IP mappings for an organisation's named hosts. The can be maintained by an organisation or service provider.

Local DNS Servers

Local DNS name servers do not strictly belong to a hierarchy, rather they belong to ISPs and are configured with a local DNS server address or learn servers via a host configuration protocol.

When a host makes a DNS query it is sent to its local DNS server which holds a cache of recent name-to-address translation pairs. The local server acts as a proxy that forwards the query into the heirarchy.

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 name server
- `MX`

- `value` is the name of the mailserver associated with `name`

P2P Applications

P2P architecture involves end systems communicating with other end systems directly. There is no always-on server, and peers are intermittently connected and change IP addresses.

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 chunks and distributed via 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 chunks and their cryptographic hashes.

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

Requesting Chunks

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

Sending Chunks

Sending Chunks

A peer p sends chunks to four other peers that are sending chunks at the highest rate to p. Every 30 seconds another peer is randomly selected to send chunks to, known as 'optimistically unchoking'.

Distributed Hash Table (DHT)

A DHT is a distributed P2P database containing key value pairs representing tracker information. This removes the reliance on a centralised tracker.

Video Streaming & CDNs

Video

A video is a sequence of images displayed at constant rate, where each image 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 to stream multimedia from a server to a client.

The server

- Divides video files into multiple 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 reques the chunk, *what* encoding rate to request and *where* to request the chunk from

CDNs

To stream content to hundreds of thousands of simultaneous users, content cistribution networks (CDNs) are used to store copies of multimedia at various different geographical locations.

Sockets With UDP & TCP

Sockets With UDP

In UDP there is no 'connection' between the client and server. The client explicitly attaches the IP destination address and port number to each packet, and the server extracts the clients IP address and port number from the received packet.

In UDP, transmitted data may be lost or received out of order.

```
# Client pseudocode
socket = createSocket();

while True:
  data = {...};
  send(data, SERVER_PORT_NUMBER);
  response = receive(SERVER_PORT_NUMBER);

socket.close();

# Server pseudocode
socket = createSocket();

socket.bind(SERVER_PORT_NUMBER);

while True:
  request = receive(CLIENT, SERVER_PORT_NUMBER);
  data = {...};
  send(CLIENT, data);

socket.close();
```

Sockets With TCP

In TCP, the client must contact a server that is already running and waiting for a connection. The client contacts the server by creating a TCP socket, specifying the IP address and port number of the server process.

```
# Client pseudocode
socket = createSocket();

socket.connect(SERVER_IP, SERVER_PORT_NUMBER);

data = {...};
socket.send(data, SERVER_PORT_NUMBER);
```

```
response = socket.receive(SERVER_PORT_NUMBER);

socket.close();


# Server pseudocode
socket = createSocket();

socket.bind(SERVER_PORT_NUMBER);
os.register(socket);

while True:
  connectionSocket = socket.acceptConnection();

  request = socket.read();

  data = {...};
  socket.write(data);

  connectionSocket.close();

socket.close();
```

Transport Layer

Services

Transport layer services and protocols provide logical communication between app processes running on different hosts. Transport layer protocols run in end systems where

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

Multiplexing & 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 segments to the socket with that port number.

IP datagrams with same destination port number, but different source IP addresses and/or source port numbers will be directed to the same socket at the destination.

Connection-oriented Demultiplexing

A 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 identified by its own 4-tuple.

UDP

UDP is a 'bare bones' internet 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 UDP segment is handled independently of others.

A UDP segment header consists of

- Source port number
- Destination port number
- Length (in bytes, including header)
- Checksum (optional)
- Payload

A UDP segment header is 8 bytes in size.

Checksum

The goal of a checksum in UDP is to detect errors in transmitted segments.

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

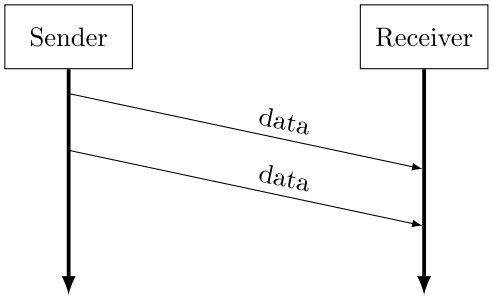
- Adds all the received contents together as 16-bit integers
- Adds the value computed to the checksum
- If the result is not `1111 1111 1111 1111`, there are errors

Principles Of Reliable Data Transfer

Principles Of Reliable Data Transfer

RDT 1.0

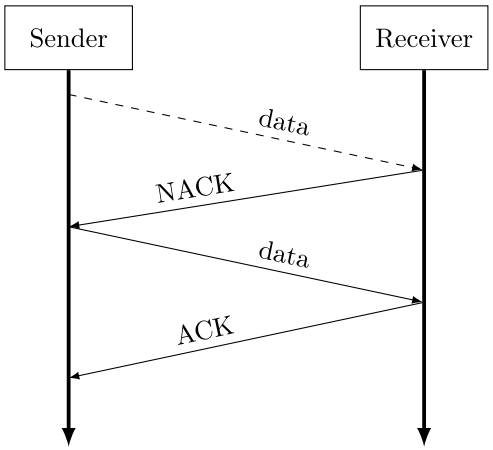
In RDT 1.0, there is an underlying channel that is perfectly reliable, it has no bit errors and no loss of packets. The transport layer does nothing.



RDT 2.0

In RDT 2.0, the underlying channel may flip bits in packets. To recover from errors the receiver sends **ACKs**, acknowledging that the packet was received ok, or **NACKs** to communicate that the packet had errors. If a **NACK** is received by the sender, the packet is retransmitted.

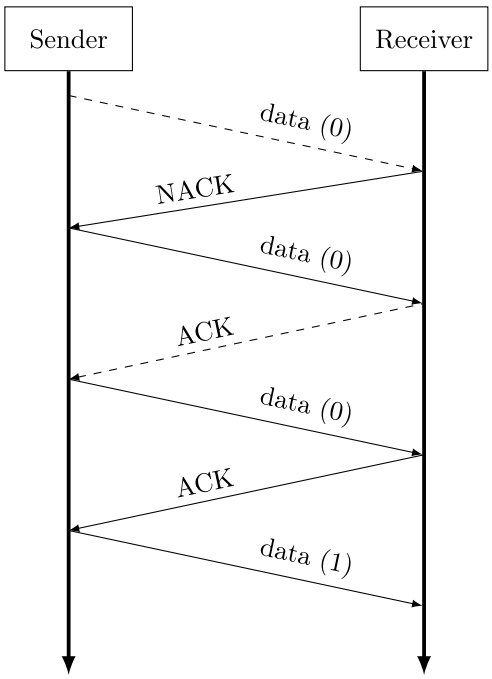
Compared to RDT 1.0, RDT 2.0 has *error detection*, *feedback* via (N)ACKs and *retransmission*.



If a (N)ACK is corrupted, then the sender doesn't know what happened at receiver and can't just retransmit the packet.

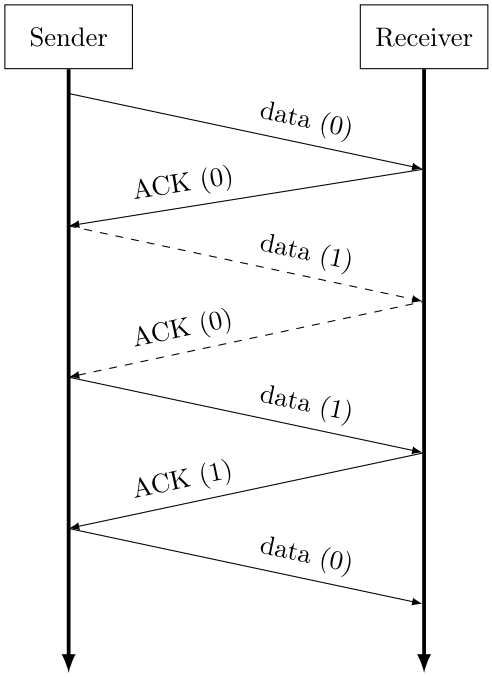
RDT 2.1

In RDT 2.1, the sender adds a sequence number to the packet. Two sequence numbers are used throughout transmittion, 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 the packets sequence numbers.



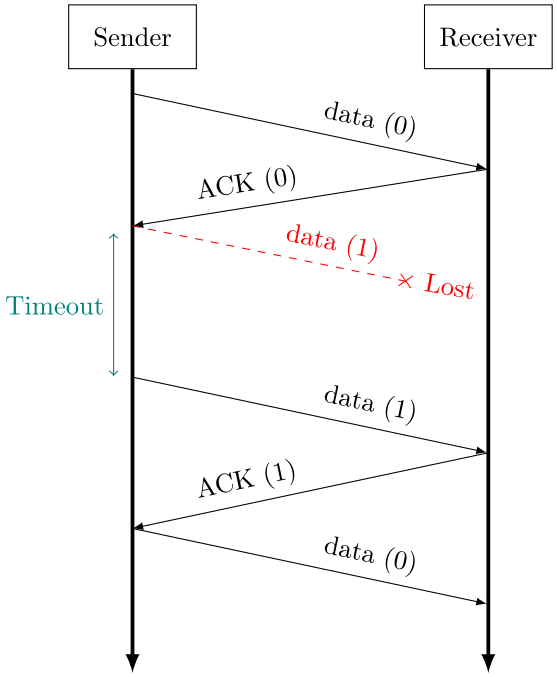
RDT 2.2

In RDT 2.2, the functionality is the same as RDT 2.1, except there are no NACKs. Instead of a NACK, the receiver sends an ACK for last packet received ok. The receiver must explicitly include the sequence number of the packet being ACKed. A duplicate ACK at the sender results in the same action as NACK - retransmit the packet.



RDT 3.0

In RDT 3.0, we assume that the underlying channel can also lose packets, which means ACKs and data can be lost. To combat this, the sender waits a 'reasonable' amount of time for the receivers ACK. If no ACK is received in time, the sender retransmits the packet. If the packet is just delayed, the retransmission will be a duplicate, but sequence numbers will handle this.



Pipelined Protocols

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.

There are two generic forms of pipelined protocols: **go-back-N** and **selective repeat**.

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

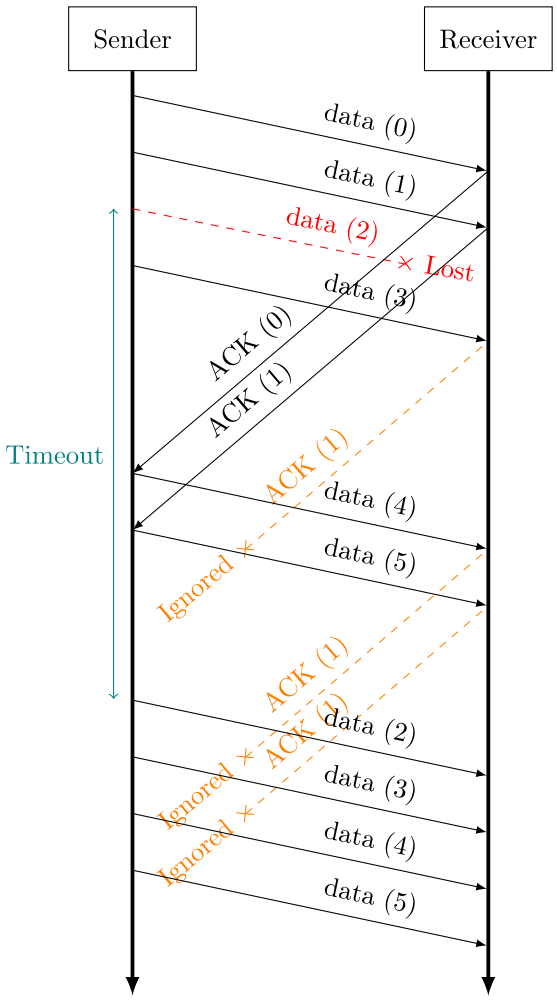
$$U = N\frac{L/R}{L/R + RTT}$$

Go Back N

- The sender can have upto N unACKed packets in the pipeline
- The sender has single timer for the oldest unACKed packet, and when the timer expires, the sender will retransmit all unACKed 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

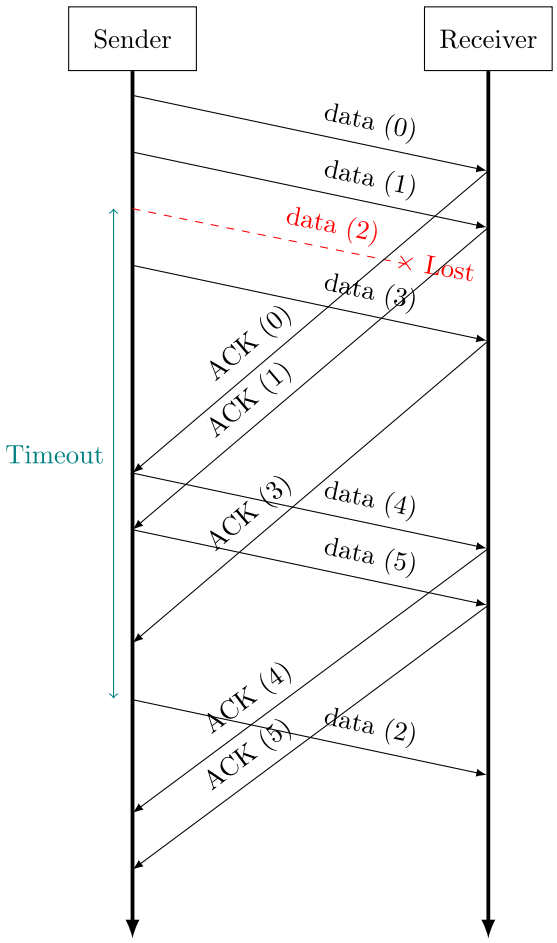
gap

- Sender window size $< 2^n$ where n is the number of bits in the sequence number space



Selective Repeat

- The sender can have up to N unACKed packets in pipeline
- The sender maintains a timer for each unACKed packet, and when the timer expires, the sender will retransmit only that unACKed 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 $\leq 1/2$ of the sequence number space



TCP

TCP is

- Point to point
  - 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
- Connection oriented
  - Handshaking initiates sender and receiver state before data exchange
- Flow controlled
  - Sender will not overwhelm receiver

A TCP segment header consists of

- Source port number
- Destination port number
- Sequence number

- Acknowledgement number
- Receive window
- Header length
- Checksum
- Options
- Data

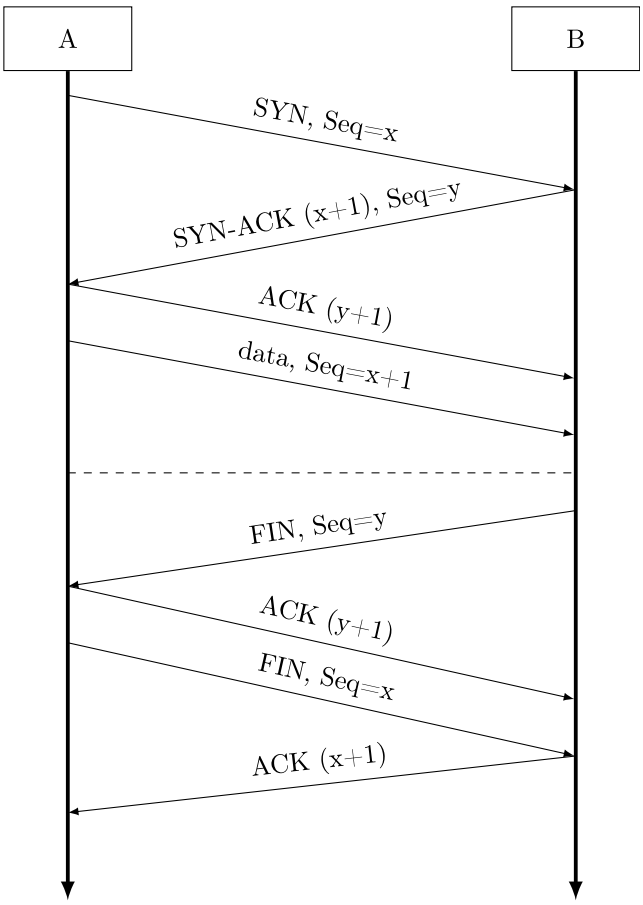
A TCP segment header is 20 bytes in size. An entire TCP segment is no more than the **Maximum Segment Size** (MSS).

TCP Timeout

- $TimeoutInterval = EstimatedRTT + 4 \cdot DevRTT$
- $EstimatedRTT = (1 - \alpha) \cdot EstimatedRTT + \alpha \cdot SampleRTT$
  - Exponential weighted moving average
  - Influence of past sample decreases exponentially fast
  - Typically $\alpha = 0.125$
- $SampleRTT$ is the measured time from segment transmission until ACK receipt
- $DevRTT = (1 - \beta) \cdot DevRTT + \beta \cdot |SampleRTT - EstimatedRTT|$
  - Typically $\beta = 0.25$

Connection Management

1. **A** sends a `SYN` packet to tell **B** it wants to open a connection
   - Starts from some ISN
2. **B** sends a `SYN-ACK` packet to tell **A** it accepts and is ready to receive the next byte
   - The `SYN-ACK` contains 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
   - The `ACK` contains an 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**
5. When finished, **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



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 delivery latency
- Increases loss rate
- Increases retransmissions, many being unnecessary
- Increases congestion even more

There are two approaches to managing congestion

- **End-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 with TCP
- **Network-assisted congestion control**

- **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

$$rate \simeq \frac{cwnd}{RTT} \text{ bytes/sec}$$

where

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

To control congestion, the window size is varied which impacts the sending rate.

TCP Slow Start

When a TCP connection begins, the value of $cwnd = 1$ MSS. Every time a transmitted segment is first acknowledged (i.e. every RTT), $cwnd$ is doubled. In the slow start phase, the sending rate is increased exponentially until the first loss is experienced. The initial sending rate is slow, but increases rapidly.

If there is a loss event indicated by a *timeout*, $cwnd$ is set back to 1 MSS, and the slow start phase starts again. A variable $ssthresh$ is also set to $cwnd/2$.

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

Congestion Avoidance

On entry to the congestion avoidance phase, the value of $cwnd$ is approximately half its value when congestion was last encountered. Now, rather than doubling the value of $cwnd$ every RTT, the value of $cwnd$ only increases by 1 MSS.

The congestion avoidance phase behaves the same way as the slow start phase when a timeout occurs, that is it sets $cwnd$ to 1 MSS and the value of $ssthresh$ is set to $cwnd/2$.

In the case of a loss event indicated by a *triple duplicate ACK*, the value of $cwnd$ is halved and $ssthresh$ is set to $cwnd/2$. In this case, the congestion avoidance phase ends and the fast recovery phase begins.

fast recovery phase begins.

Fast Recovery

In fast recovery, the value of $cwnd$ is increased by 1 MSS for every duplicate ACK received for the missing segment that caused TCP to enter the fast recovery phase. Eventually when an ACK arrives for the missing segment, TCP enters the congestion avoidance phase after deflating $cwnd$.

If a timeout occurs, TCP enters the 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 event occurs.

Additive increase refers to increasing the $cwnd$ by 1 MSS every RTT until a loss is detected.

Multiplicative decrease refers to cutting $cwnd$ in half after a loss.

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

TCP Flavours

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

Network Layer

The network layer

- Transports segments from sending to receiving hosts
- Encapsulates segments into datagrams on the sending side
- Delivers segments to the transport layer on the receiving side
- Has network layer protocols in every host and router
- Routers examine header fields in all IP datagrams passing through them

There are two key network layer functions

- **Forwarding**: moving packets from router to router
  - A forwarding table is used to determine local forwarding at a given router
- **Routing**: determining route taken by packets from source to destination
  - Routing algorithms are used to determine the end to end path through a network

The **data plane** is a local, per router function that determines how a datagram arriving at a router is forwarded to another router.

The **control plane** is network wide logic that determines how a datagram is routed among routers along an end to end path from source host to destination host. There are two control plane approaches; traditional routing algorithms implemented in routers, or software defined networking implemented in a centralised server.

IP

Datagram Format

- Version (3 bits)
  - IP protocol version number
- Header length (5 bits)
  - Header length (bytes)
- Type of service (8 bits)
  - "Type" of data
- Length (16 bits)
  - Total datagram length (bytes)
- 16-bit identifier (16 bits)
- Flags (3 bits)
  - Flags for fragmentation and reassembly
- Fragment offset (13 bits)
- TTL (8 bits)
  - Max number remaining hops
- Upper layer (8 bits)
  - Upper layer protocol to deliver payload to
- Header checksum (16 bits)
- 32 bit source IP address (32 bits)
- 32 bit destination IP address (32 bits)
- Options (if any) (32 bits)
  - For example, timestamp, whether to record route taken, specific list of routers to visit

- Data (variable length, typically a TCP or UDP segment)

TTL

Every packet being forwarded from router to router has a TTL. After each hop, this TTL is decremented, and if it reaches 0 then the packet is discarded and a 'time exceeded' message is sent to the source. This mechanism helps to prevent loops in a network.

Fragmentation

Network links have a maximum transfer unit (MTU) size which represents the largest possible link-level frame. A large IP datagram is divided, or *fragmented*, with the network if it is larger than the MTU. One datagram becomes several, and the IP header bits are used to reassembled it at the destination.

The IPv4 fragmentation procedure is as follows

1. Router breaks up datagram in sizes that the output link can support
2. Router copies IP header to pieces
3. Router adjusts length of pieces
4. Router sets offset of pieces to indicate position
5. Router sets `MF flag=1` on all pieces except the last to indate "more fragments"
6. Router re-computes the checksum
7. The receiving host uses the `Id` field with the `MF flag` field and offset to reassemble the datagram

Fragmentation of fragments is also supported.

An example is given below with a 4000 byte datagram (`Packet`) and a MTU of 1500 bytes. The datagram is split into three smaller datagrams (`P1` , `P2` , `P3`).

```
MTU = 1500
Packet = [length=4000 | Id=x | MF flag=0 | Offset=0]

Data = 4000 - 20 (IP Header)
     = 3980

F1 = 1480
F2 = 1480
F3 = 1020

Offset = 1480/8

P1 = [length=1500 | Id=x | MF flag=1 | Offset=0]
P2 = [length=1500 | Id=x | MF flag=1 | Offset=185]
P3 = [length=1040 | Id=x | MF flag=0 | Offset=370]
```

```
P3 = [length=1040 | id=x | MF flag=0 | offset=370]
```

Path MTU Discovery

- Host
  - Sends a large packet to test whether all routers in path to the destination can support it or not
  - Sets `DF` (do not fragment) flag
- Routers
  - Drops the packet if it is too large
  - Provides feedback to the host with an ICMP message telling them the MTU

IPv4 Addressing

An **IP address** is a 32 bit identifier for hosts and router interfaces. An **interface** is a connection between a host/router and a physical link. Each interface has an associated IP address.

An IP address has a network part (high order bits) and a host part (low order bits).

Masking

A mask is used in conjunction with the network address to indicate how many higher order bits are used for the network part of the address. A bitwise AND is performed on the IP address and mask.

A **broadcast address** an address where the host part is all 1's, i.e. `223.1.1.255`. A **network address** is an address where the host part is all 0's, i.e. `223.1.1.0`. Both of these addresses are not assigned to any host.

Classes

- If the first bit is a 0
  - The address belongs to class A
- Else if the second bit is a 0
  - The address belongs to class B
- Else if the third bit is a 0
  - The address belongs to class C
- Else if the fourth bit is a 0
  - The address belongs to class D
- Else
  - The address belongs to class E

Subnetting

Subnetting

Subnetting is the process of dividing the class A, B or C networks into more manageable chunks that are suited to a network's size and structure. Subnetting allows 3 levels of hierarchy, netid, subnetid and hostid. The original netid remains the same and designates the site. Subnetting remains transparent outside of the site.

The process of subnetting simply extends the point where the 1's of the mask stop and 0's start. Some host ID bits are sacrificed to gain network ID bits.

DHCP

DHCP allows hosts to dynamically obtain its IP address from a network server when it joins a network. The host can renew its lease on an address in use.

The process of obtaining an IP address with DHCP is

1. The host broadcasts a "DHCP discover" message
2. The DHCP server responds with a "DHCP offer" message
3. The host requests an IP address with a "DHCP request" message
4. The DHCP server sends an address with a "DHCP ack" message

Longest Prefix Matching

When looking for a forwarding table entry for a given destination address, use the longest address prefix that matches the destination address.

Network Address Translation

A NAT router must

- Replace the source IP address and port number of every outgoing datagram with the NAT IP address and a new port number
- Remember, in a NAT translation table, every source IP and port number to NAT IP address and new port number translation pair
- Replace the NAT IP address and port number of every incoming datagram with the corresponding source IP address and port number stored in the NAT translation table

Pros	Cons
A local network uses just one IP address as far as the outside world is concerned	It violates the architectural model of IP
A range of IP addresses is not needed from an ISP, rather just one IP address for all devices	It changes the internet from connectionless to a kind of connection oriented network

Pros	Cons
Can change addresses of devices in a local network without notifying the outside world	
Can change ISP without changing addresses of devices in local network	

IPv6 Addressing

The initial motiviation to introduce IPv6 addressing was that the 32 bit IPv4 address space is soon to be complete allocated.

The IPv6 datagram format has a fixed length 40 byte header, and no fragmentation is allowed. In IPv6, the checksum in the head is completely removed to reduce processing time at each hop.

Routing Protocols

A **routing algorithm** is an algorithm that finds the least cost path between some two nodes n_i and n_j in a weighted graph G.

Link State

In link state routing, each node maintains it's local "link state", i.e. a list of its directly attached links and their costs. Each node floods its local link state. Eventually, each node learns the entire network topology.

The time complexity for link state routing is

- $\mathcal{O}(nE)$ for message complexity with n nodes and E links
- $\mathcal{O}(n^2)$ for speed of convergence

Dijkstra's Algorithm

In a network topology, each link's cost is known to all nodes. Dijkstra's algorithm computes the least cost paths from one node (the 'source') to all other nodes. After k iterations, the least cost paths to k destinations are known.

Let

Let

- $c(x, y)$ be the link cost from node x to y, ∞ if not direct neighbors
- $D(v)$ be the current value of the cost of path from a source to the destination v
- $p(v)$ be the predecessor node along a path from a source to v
- N be the set of nodes whose least cost path is definitively known

Then, Dijkstra's algorithm is given by

```
N = {u}

for all nodes v:
 if v adjacent to u:
   D(v) = c(u, v)
 else:
   D(v) = INFTY

while all nodes not in N:
  find w not in N such that D(w) is a minimum
  add w to N

  for all v adjacent to w and not in N:
    D(v) = min(D(v), D(w) + c(w, v))
```

Distance Vector

Let

- $d_x(y)$ be the cost of the least cost path from x to y
- $c(x, y)$ be the cost from x to neighbor y
- $d_v(y)$ be the cost from neighbor v to destination y

then, $d_x(y) = \min(c(x, v) + d_v(y))$ for all $v \in \{neighboring(x)\}$.

In distance vector routing, each router maintains its shortest distance to every destination via each of its neighbours.

First each router initializes its distance table based on its immediate neighbors and link costs. Then each router sends its distance vector (i.e. its 'shortest path' to every other router) to its immediate neighbors, which is then processed. This is repeated and converges to the set of shortest paths.

Poisoned Reverse Rule

The Poisoned Reverse rule is a rule to avoid 'counting to infinity'. If a router A routes via B to get to C, then A tells B its distance to C is infinite

to get to C, then A tells D its distance to C is infinite.

ICMP

The Internet Control Message Protocol is a protocol used by hosts and routers to communicate network level information. It works above the IP layer. An ICMP packet contains a type and a code corresponding to different messages.

Link Layer

The link layer has the responsibility of transferring datagrams from one node to another physically adjacent node over a link.

Datagrams are transferred by different link protocols over different links. Each link protocol provides a different service.

The link layer provides

- Flow control
  - Controlling flow between adjacent sending and receiving nodes
- Error detection and correction
  - Receiver detects and corrects bit errors without resorting to retransmission
- Half duplex and full duplex
  - With half duplex, nodes at both ends of link can transmit, but not at same time

The link layer is implemented in a network interface card in each and every host. It is implemented as a combination of software and hardware.

On the sending side, a datagram is encapsulated in a *frame* and extra information (e.g. error checking bits, flow control) is added.

On the receiving side, the frame is checked for errors, and the datagram is extraced and passed up to the layer above.

Error Detection & Correction

Parity

A method of error detection is to add a parity bit every d bits. On the receiver side, the number of bits in every block of size d is counted and compared with the adjacent parity bit.

This method can only detect when an even number of bits gets flipped, and even then it can only detect errors, not correct them.

The cost of this method is one extra bit for every d bits.

Two Dimensional Parity

Rather than just computing parity every d bits, we can compute a parity bit for every set of n 'columns' too.

This method can detect 1, 2, 3 and some 4 bit errors. It can also correct single bit errors.

Multiple Access Protocols

Given a single shared broadcast channel, if two or more simultaneous transmissions by nodes occur, then interference occurs. If a node receives two or more signals at the same time, then a collision occurs.

A multiple access protocol is a distributed algorithm that determines how nodes share channels, i.e. it determines when nodes can transmit. Communication about channel sharing must use the channel itself.

Given a broadcast channel of rate R bps, we want

- When one node wants to transmit, it can send at rate R
- When M nodes want to transmit, each can send at average rate R/M
- Full decentralisation, i.e. no special node to coordinate transmissions and no synchronisation of clocks
- Simplicity

There are three broad classes of multiple access protocols

- Channel partitioning
  - Divide a channel into smaller pieces (i.e. time slots, frequency, code)
  - Allocate pieces to nodes for exclusive use
- Random access
  - A channel is not divided and allows for collisions
  - A channel can recover from collisions
- Taking turns
  - Nodes take turns, but nodes with more to send can take longer turns

Slotted ALOHA

Slotted ALOHA is a random access multiple access protocol.

Assumptions

- All frames are the same size
- Time is divided into equal size slots

- Nodes can start to transmit only at the beginning of a slot
- Nodes are synchronised

- If two or more nodes transmit in a slot, all nodes detect a collision

When a node obtains a fresh frame, it transmits in the next slot. If there is no collision, the node can send a new frame in the next slot. If there is a collision, the node retransmits the frame in each subsequent slot with probability p until success.

Pros	Cons
A single active node can continuously transmit at the full rate of the channel	Collisions and therefore wasted slots
Highly decentralised	Idle slots
Simple	Nodes may be able to detect collisions in less than the time it takes to transmit a packet
	Clock synchronisation

Slotted ALOHA is not that efficient, as in the long run, only a fraction of slots are successful in transmitting frames. At best, the channel is used for useful transmissions only 37% of the time.

Carrier Sense Multiple Access

Carrier Sense Multiple Access (CSMA) is another multiple access protocol that listens before transmission. If a channel is sensed as idle, the entire frame is transmitted. If a channel is sensed as busy, the transmission is deferred.

In CSMA, collisions can still occur due to propagation delay. Propagation delay means two nodes may not hear each other's transmission. If a collision occurs, the entire frame's transmission time is wasted.

Collision Detection

For ethernet, the collision detection algorithm is as follows

1. The NIC receives datagram from the network layer and creates a frame
2. If the NIC senses a channel is idle, it starts a frame transmission. If the NIC senses channel is busy, it waits until the channel is idle, then transmits

3. If the NIC transmits the entire frame without detecting another transmission, the NIC is done

4. If the NIC detects another transmission while transmitting, it aborts and sends a jam signal
5. After aborting, the NIC enters binary (exponential) backoff
    - After mth collision, the NIC chooses a random k from $\{0, 1, 2, \ldots, 2^m - 1\}$
    - The NIC waits $k \cdot 512$ bit times and returns to step 2
    - The backoff interval increases as the number of collisions increase

Taking Turns

Polling:

- A master node invites slave nodes to transmit in turn
- Typically used with 'dumb' slave devices
- Some concerns include the polling overhead, latency and having a single point of failure (the master node)

Token passing:

- A control token is passed from one node to the next sequentially
- Some concerns include the token overhead, latency and having a single point of failure (the token)

Switched LANs

Addressing

A MAC (or LAN/physical/ethernet) address is a 48 bit address built in to NIC ROM. It is in a hexadecimal format. MAC address allocation is administered by IEEE. The namespace of a MAC address is flat, compared to an IP address which is hierarchical. MAC addresses are used to get packets between interfaces on the same network.

An **ARP table** is a mapping from an IP address to a MAC address with an associated TTL. Each LAN contains an ARP table.

Ethernet

Ethernet is a dominant wired LAN technology. Previously it used a *bus* for transmission, but now more commonly uses a *star* topology which contains an active switch in the center with each 'spoke' running a seperate ethernet protocol.

Ethernet is

- Connectionless
  - There is no handshaking between sending and receiving NICs
- Unreliable
  - A receiving NIC does not send ACKs or NACKs to the sending NIC

Ethernet's MAC protocol is unslotted CSMA/CD with binary backoff.

Switches

A switch is a link layer device that stores and forwards ethernet frames. It examines incoming an frame's MAC address, and selectively forwards it to one or more outgoing links.

Switches are *transparent* in the sense that hosts are unaware of their prescence. Switches also don't need to be configured.

Hosts have a dedicated, direct connection to a switch. Switches buffer packets and runs at full duplex.

Switches are *self learning* in the sense that they learn which hosts can be reached through which interfaces. When a frame is received, the switch learns the location of the sender and records the sender/location pair in it's switch table.

Wireless Networks

Wireless Links

Compared to a wired link, a wireless link has

- Decreased signal strength
- Interference from other sources
- Multipath propagation (i.e. signals bouncing off objects)

The *free space path loss* is calculated as $\left(\frac{4\pi d}{\lambda}\right)^2 \equiv \left(\frac{4\pi df}{c}\right)^2$ where

- d is the distance
- λ is the wavelength
- f is the frequency
- c is the speed of light

WiFi

There are multiple types of WiFi protocols including 802.11b, 802.11a, 802.11g and 802.11n. They all have minor differences including their spectrum/range and bit rate.

802.11 LAN Architecture

A wireless host communicates with a base station or access point (AP).

The 802.11b protocol's 2.4GHz spectrum is divided into 11 channels at different frequencies. The AP admin chooses the frequency for their AP, but this may interfere with neighboring APs.

A host must associate with an AP, which involves scanning channels, selecting an AP, possibly performing authentication and typically running DHCP to get an IP address for the APs subnet.

802.11 MAC Protocol

802.11 uses CSMA with collision avoidance (CSMA/CA) as its random access protocol. This performs the same way as CSMA/CD, except it has collition avoidance rather than collision detection. This means that once a station begins to transmit a frame, it transmits the frame in its entirety.

If more than one station senses that a channel is busy, they all immediately enter random backoff, hopefully choosing different backoff values. The first station to come out of random backoff is the 'winner' and starts transmitting on the channel. All other 'losing' stations now sense that the channel is in use, and refrain from transmitting.

RTS & CTS

To avoid the 'hidden terminal' problem, 802.11 employs a strategy known as Request to Send and Clear to Send (RTS/CTS). When a station wants to send data, it first sends an RTS frame indicating the total time required to transmit the data and the ACK frame. A CTS frame is then broadcasted from the access point indicating that the sender is clear to begin transmission. All other stations will hear this CTS frame and will refrain from transmitting.

Network Security

Cryptography

Let

- m be a plaintext message
- $K_A(m)$ be ciphertext, encrypted with key K_A

Symmetric Key Cryptography

Users share the same (symmetric) key K_S.

DES

Data Encryption Standard (DES) is a US encryption standard that uses a 56 bit symmetric key, and a 64 bit plaintext input.

AES

Advanced Encryption Standard (AES) is a symmetric key NIST standard that replaces DES. It processes data in 128 bit blocks using 128, 192 or 256 bit keys.

Stream Ciphers

Stream ciphers encrypt one bit at time. Each bit of the keystream is combined with a bit of plaintext to get a bit of ciphertext.

We have

- $m(i)$ being the ith bit of the message
- $ks(i)$ being the ith bit of the keystream
- $c(i)$ being the ith bit of the cyphertext

Then

- $c(i) = ks(i) \oplus m(i)$
- $m(i) = ks(i) \oplus c(i)$

Block Ciphers

Block ciphers break plaintext message in to equal sized blocks. Each block is encrypted as a unit. Ciphertext is processed as k bit blocks. A one to one mapping is used to map a k bit block of plaintext to a k bit block of ciphertext.

Public Key Encryption Algorithms

We need K_B^+ and K_B^- such that $K_B^-(K_B^+(m)) \equiv m$. Given a public key K_B^+, it should be impossible to compute the corresponding private key K_B^-.

RSA

To create a public/private key pair

1. Choose two large prime numbers p, q
2. Compute $n = pq$, $z = (p-1)(q-1)$
3. Choose e (with $e < n$) that has no common factors with z (i.e. e and z are relatively prime)
4. Choose d such that $ed - 1$ is exactly divisible by z (i.e. $ed \bmod z = 1$)

At this point the public key K_R^+ is (n, e) and private key K_R^- is (n, d).

To *encrypt* a message m, compute $c = m^e \bmod n$. To *decrypt* a received bit pattern c, compute $m = c^d \bmod n$.

An important property of RSA encryption is that $K_B^-(K_B^+(m)) \equiv m \equiv K_B^+(K_B^-(m))$.

Message Integrity

Confidentiality refers to when a message is kept private and secret. **Integrity** refers to the protection against message tampering. Encryption alone may not guarantee integrity. Both confidentiality and integrity are needed for security.

Digital Signatures

A digital signature is a cryptographic technique analogous to hand written signatures. A message is signed with a private key K_B^- creating a signed message $K_B^-(m)$. A message is verified by applying a public key K_B^+ to m and checking that $K_B^+(K_B^-(m)) \equiv m$.

Message Digests

It is computationally expensive to public key encrypt long messages. A message digest is a fixed length, easy to compute digital footprint. A hash function H is applied to m to get a fixed sized message digest $H(m)$.

The hash function H has the following properties

- Many to one
- Produces a fixed sized message digest
- Given a message digest x, it is computationally infeasible to find m such that $x \equiv H(m)$

Some common hash function algorithms are MD5, SHA-1, SHA-2 and SHA-3.

Securing Email

PGP

PGP is the de facto standard for email encryption. On installation PGP creates a public, private key pair. There is an option to digitally sign the message, encrypt the message or both. PGP uses MD5 or SHA to compute a message digest and RSA for public key encryption.

---

lukakerr

Let me know how you found my site, send me an email!