Computer Networking Cheatsheet

Edoardo Riggio

June 5, 2021

Computer Networking - SP. 2021
Computer Science
Università della Svizzera Italiana, Lugano

Contents

1 Computer Networking and the Internet

1.1 Internet

The Internet is both a computer network that connects together billions of computing devices around the world, and an infrastructure that provides services to applications.

All of the devices connected to the Internet are called **hosts** or **end systems**. These hosts and end systems are connected together by a network of **communication links** and **packet switches**. The transmission rate of links is measured in bits/s.

When two end systems need to exchange data, this is divided into **packets** and are sent through the network. When they arrive at destination, then they are reassembled. A **packet switch** takes a packet arriving on one of its incoming links and forwards them to one of its outgoing links. Packet switches can be of different types, such as **routers** and **link-layer switches**.

End systems can access the Internet through **ISPs**. Each ISP in itself is a network of packet switches and communication links.

1.2 Internet Services

Applications on the Internet are said to be **distributed applications**, since they involve multiple end systems that exchange data with each other. In order for a program to instruct the Internet on how to deliver data to another program on another end system, **socket interfaces** are used. These interfaces are provided by hosts attached to the Internet.

1.3 Protocols

All activity in the Internet that involves two or more communicating remote entities is governed by a protocol.

A protocol defines the format and the order of messages exchanged between two or more communicating entities, as well as actions taken on the transmission and/or receipt of a message or other event. Different protocols are used to accomplish different communication tasks.

Internet standards are developed by the **IETF** (Internet Engineering Task Force) by means of **RFCs** (Requests For Comment). These RFCs define protocols such as TCP, IP, HTTP, SMTP...

Other bodies also specify standards for network components, such as the **IEEE** (Institute of Electrical and Electronics Engineers).

1.4 Network Edge

A network edge is composed mainly of two parts:

- **Network Edge**

  Is the part of the network which is composed of applications and end systems. These end systems are also referred to as **hosts**. That is because they run application programs.

- **Access Network**

  It is the network that physically connects an end system to the fist router, on a path from an end system all the way to any other distant end system.

1.5 Physical Media

1.5.1 Guided Media

In the case of guided media, the electromagnetic/optical waves are guided along a solid medium. Some examples of this kind of medium are:

- **Twisted-Pair Copper Wire**

  Most used form is **UTP** (Unshielded Twisted Pair), and it is used for a computer network within a building.

- **Coaxial Cable**

  This medium is made out of copper (core) and has an internal concentric structure.

- **Fiber Optic Cable**

  This type of medium transfers bits as light pulses. It is immune to electromagnetic interference, have a very low signal attenuation up to 100 km and are very hard to tap.

1.5.2 Unguided Media

In the case of unguided media, electromagnetic waves are propagated in the atmosphere and/or in outer space.

- **Terrestrial Radio Channel**

  No physical wire is needed to be installed, waves can penetrate walls and have a long range of action. Terrestrial radio channel can be divided into 3 categories: short range (Bluetooth), medium range (Wi-Fi) and long range (3G, LTE, 4G, 5G ...).

- **Satellite Radio Channel**

  Two types of satellites are used: **geostationary satellites** and **LEO satellites** (Low-Earth Orbiting).

1.6 Packet Switching

In order to send a message, the source breaks a long message into smaller chunks called **packets**. Between the source and the destination, each packet travels through **communication links** and **packet switches**. Packets are transmitted at a rate equal to the full transmission rate of the link. The formula is:

$$\Delta_{TX} = \frac{L}{R}$$

1.6.1 Store-and-Forward

Most packets use the store-and-forward transmission. This means that the packet must be fully received before it can be forwarded to the next packet switch. The end-to-end delay over N routers can be measured as follows:

$$d_{end-to-end} = N \cdot \frac{L}{R}$$

1.7 Forwarding Tables and Routing Algorithms

Each router has a **forwarding table** that maps destination addresses to that router's outbound links. This is used by the router, in combination with the destination IP address in the packet's header, in order to find the next router the packet needs to be sent to.

Routing protocols may, for example, determine the shortest path the packet needs to traverse in order to reach its destination.

1.8 Circuit Switching

In circuit switched networks, the resources needed along the path to provide for communication between the end systems are reserved for the whole duration of the connection. For this duration, also a constant transmission time is reserved and it is guaranteed to be constant. A circuit in a link is implemented using either:

- **FDM (Frequency Division Multiplexing)**

  The frequency spectrum of the link is divided up among the established connections across the link. The link dedicates the frequency of the band for the entire duration of the connection. The bandwidth for telephone networks is 4 kHz, while FM radio stations share the 88-108 MHz frequency spectrum.

- **TDM (Time-Division Multiplexing)**

  In this case time is divided into frames of fixed duration, and each frame is divided into a fixed number of time slots. When the network establishes a connection, the network dedicates one time slot in every frame for this connection.

Circuit switches can be wasteful because the dedicates circuits are idle during **silent periods**. These idle periods are instead used by packet switching networks and used for other connections.

1.9 Structure of the Network

In order to connect end users to each other, a network must be built. This network is hierarchical and is composed of multiple interconnected networks. These are the networks:

- **Network Structure 1**

  This network structure interconnects all of the access ISPs with a **single global transit ISP**. This would be a network of routers and communication links that spans the globe.

- **Network Structure 2**

  This network consists of **many access ISPs** and a **few global transit ISPs**. Global transit ISPs themselves must be interconnected.

- **Network Structure 3**

  In any given region, there may be a **regional ISP** to which each **access ISP** connects. Each regional ISP then connects to **tier-1 ISPs**. These are the global transit ISPs.

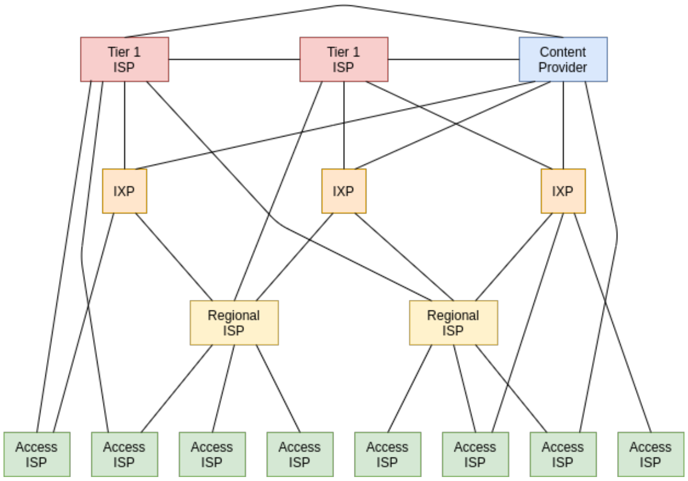
- **Network Structure 4**

  PoPs (Points of Presence) are added in order to group one or more routers at the same location such that they can be connected to provider ISPs. Any ISP, could also decide to **multi-home**. That is, to connect two or more provider ISPs. **Peering** is when two ISPs that are both near and at the same hierarchy, connect their networks together. Finally, a third-party company can create an **IXP** (Internet eXchange Point), which is a meeting point where multiple ISPs can peer together.

- **Network Structure 5**

  Content-Provider Networks are added to the network – such as Google. These networks peer with lower-level ISPs and connect to tier-1 ISPs.

This can be a schematic representation of the modern network of networks:



1.10 Packet Delay

As a packet travels from one node to the subsequent node, the packet suffers from several types of delay at each node.

1.10.1 Processing Delay

The processing delay can be caused by the time required by routers and switches to examine the packet's header and finding out where it needs to go; it can also be caused by the time the router or switch takes in order to find bit-level errors in the packet, that may happen when the packet is transferred from a node to the other.

This type of delay is in the order of μs.

1.10.2 Queuing Delay

The queuing delay is caused by the number of earlier-arriving packets that are waiting for transmission. If the traffic is heavy, then the time that the packets need to wait for them to be transmitted is longer. This kind of delay is calculated with the following formula:

$$d_{queue} = \frac{L \cdot a}{R} \cdot \frac{L}{R} \cdot \left(1 - \frac{L \cdot a}{R}\right)$$

For $\frac{L \cdot a}{R} < 1$. This type of delay is in the order of μs.

1.10.3 Transmission Delay

Assuming that we have a FIFO packet transmission, the packet can be transmitted only if all of the packets that are in front of it have been transmitted. The transmission delay can be calculated as follows:

$$d_{transmission} = \frac{L}{R}$$

This measures the time required to transmit all of the packet's bits onto the link. This type of delay is in the order of $\mu s/ms$.

1.10.4 Propagation Delay

The propagation delay depends on the physical medium of the link connecting the two nodes. The propagation speed can go from $2 \cdot 10^8 m/s$ to $3 \cdot 10^8 m/s$. The delay is calculated as follows:

$$d_{propagation} = \frac{d}{s}$$

This type of delay is in the order of ms.

1.10.5 Nodal Delay

The total nodal delay that happens at every node is:

$$d_{nodal} = d_{processing} + d_{queue} + d_{transmission} + d_{propagation}$$

1.11 Packet Loss

Because the capacity of a router/switch queue is finite, if there are more packets than the buffer can hold, then those extra packets are dropped. Performance at a node must be thus measured also in terms of packet loss other than delay.

1.12 End-to-End Delay

The total delay from source to destination is calculated as:

$$d_{end-to-end} = N \cdot (d_{processing} + d_{transmission} + d_{propagation})$$

This is true if all nodes have the same processing, transmission and propagation queues. If that were not the case, then all the single nodal delays must be added up together.

1.13 End-to-End Throughput

End-to-end throughput is the instantaneous rate at which a host can receive dat from another host. The formula is:

$$t_{end-to-end} = \frac{F}{T}$$

Throughput not only depends on the transmission rates of the links along the path, but also on the intervening traffic.

1.14 Protocol Layers

Each of the 5 layers that compose the **TCP/IP** protocol stack provides its service by performing certain actions within that layer and by using the services of the layer directly below it.

1.14.1 Application Layer

The application layer is where the network applications and their protocols reside.

Some of the most important protocols of this layer are **HTTP**, **SMTP**, **FTP**, **DNS**...

The packets at this layer are referred to as **messages**.

1.14.2 Transport Layer

The transport layer transports application-layer messages between application endpoints.

The two layers of this protocol are **TCP** and **UDP**. TCP provides a connection-oriented service. It guarantees the delivery of messages to the destination and flow control. UDP, on the other hand provides no reliability, no flow control and no congestion control.

The packets at this layer are referred to as **segments**.

1.14.3 Network Layer

The network layer moves packets from one host to the other. It also provides the service of delivering the segment to the transport layer in the destination host.

The protocol provided at this level is the **IP protocol**, which defines how routers and end systems act on the packet's fields. All Internet components that have a network layer must run the IP protocol.

The packets at this layer are referred to as **datagrams**.

1.14.4 Link Layer

The link layer moves a packet from one node to the next node in the route.

The protocols used in this layer are for example **Ethernet**, **Wi-Fi** and **DOCSIS**. A datagram might be handled by a different protocol for each node it traverses.

The packets at this layer are referred to as **frames**.

1.14.5 Physical Layer

The physical layer moves the individual bits within the frame from one node to the next.

The protocols in this layer depend on the link and on the transmission medium.

1.15 Encapsulation

Routers and link switches do not incorporate all of the layers in the TCP/IP protocol stack.

Each time the packet goes from a layer to another, a header or trailer is added to it. These headers and trailers contain information about the layer, such as the IP address and MAC address of the destination end system.

From layers 1 to 3 (application, transport and network layers) headers are added at each layer. Instead in layer 4 (link layer), both a header and a trailer are added to the packet. The trailer is used to verify the integrity of the packet.

In more complex cases, a large message may be divided into multiple transport-layer segments. These segments would have to be reconstructed by the destination host in order to get the full message.

2 Application Layer

2.1 Architectures

2.1.1 Client-Server Architecture

In this architecture there is an always-on host which has a permanent IP address. This host (server) receives requests from other hosts (clients).

2.1.2 Peer-to-Peer Architecture

In this architecture there is no always-on host. Here the two peers are directly connected to one another. Peers have different IP addresses, and request and provide services to other peers.

2.2 Process Communication

Processes that are executing in different hosts communicate by exchanging messages. These processes are called the **client process** (which initiates the communication) and the **server process** (which awaits to be connected).

2.2.1 Sockets

A socket is the interface between the application and the transport layer within a host (i.e. the API between the application and the network).

2.2.2 Sever Process Identification

In order to identify a server process, two pieces of information are needed:

- **IP Address**

  It is a **32-bit** or **64-bit** (respectively IPv4 and IPv6) quantity that can uniquely identify a host.

- **Port Number**

  Since many different processes could run on a host, the IP is not sufficient to determine a specific process. To identify a process in a host, a port number is needed. Every port number is associated with a specific process.

2.2.3 Services Needed by the Layer

The following services are needed by the application layer:

- **Reliable Data Transfer**

  The transport layer may need to provide a reliable data transfer rate. There exist applications that are **loss-tolerant** and application that require **100% reliable** data transfer rates. In the first case, applications can tolerate some loss of packets.

- **Throughput**

  A transport service may need to provide guaranteed available throughput at some specified rate. There exist applications that are **bandwidth-sensitive**, and other that are **elastic**. In the first case, applications require a minimum amount of throughput to be effective. In the second case, applications make use of whatever throughput they get.

- **Timing**

  A transport service may need to provide timing guarantees available throughput at some specified rate. Low delays are preferred to high delays.

- **Security**

  A transport service may need to provide the application with one or more security services.

2.3 Transport Services

The two protocols offered when creating a network application are **UDP** and **TCP**.

2.3.1 TCP

It includes a connection-oriented service and a reliable data transfer service. After a successful handshake, a connection is said to exist between the sockets of the two processes. Both processes can now send messages at the same time.

The communication process can rely on TCP to deliver all of the data sent, without errors and in the proper order. It also has a congestion-control mechanism.

2.3.2 UDP

It is a connectionless and lightweight protocol. It provides an unreliable data transfer service, and no guarantee that the message will even reach the destination.

2.4 Application-Level Protocols

An application-level protocol defines how an application's processes – running on different end systems – pass messages to each other. In particular they define the following:

- The type of messages exchanged (**request and response messages**)
- The syntax of messages (**fields**)
- The semantics of the fields (**info in the fields**)

- Rules of determining when and how a process sends and receives messages

There are two types of protocols:

- **Open Protocols**

  HTTP, SMTP, FTP, Telnet...

- **Proprietary Protocols**

  Skype, BitTorrent...

2.5 The Web and HTTP

The **HTTP** (HyperText Transfer Protocol) protocol is at the heart of the Web. HTTP is implemented in two separate programs: a client program and a server program. These two programs communicate with each other by means of HTTP messages.

2.5.1 Base URL

The base **URL** (Uniform Resource Locator) of a webpage is composed of the following parts:

$$\overbrace{\text{http}}^{\text{protocol}}\ ://\ \underbrace{\underbrace{\overbrace{\text{www.}}^{\text{subdomain}}\ \overbrace{\text{my-server.}}^{\text{subdomain}}\ \overbrace{\text{com}}^{\text{Top-level domain}}}_{\text{hostname}}\overbrace{:8080}^{\text{port}}}_{\text{authority}}\underbrace{\text{/assets/image.jpg}}_{\text{path name}}$$

2.5.2 HTTP Connection

HTTP uses **TCP** as its underlying protocol. In order to obtain an HTTP connection, the following steps must be followed:

1. The client initiates a TCP connection (by creating a socket) to the server on port 80;
2. The server accept the TCP connection request made by the client;
3. HTTP messages are now exchanged through the TCP link, from the client's to the server's socket;
4. The TCP connection is then dismantled when it is not needed anymore.

2.5.3 Persistent and Non-Persistent HTTP

HTTP is a **stateless protocol**. This means that it does not maintain any information about past client requests. There are two types of stateless HTTP connections:

- **Non-Persistent HTTP**

  At most one object can be sent on this type of TCP connection. Once the response has been sent from the server to the client, the TCP connection is terminated.

  In their default modes, browsers open 5 to 10 parallel TCP connections, and each of these connections handles one request-response transaction.

- **Persistent HTTP**

  Multiple objects can be sent over a single TCP connection between client and server. As little as one RTT is used for all the referenced objects.

  Persistent HTTP connections can also be pipelined, in which case two RTTs are needed to establish the connection, and one RTT is needed for **all** requested objects.

2.6 Round-Trip Time

The **RTT** (Round-Trip Time) is the time taken by a small packet to travel from a client to a server, and back. One RTT is used to initiate the TCP connection, and one RTT is used for the HTTP request and the first few bites of the HTTP response to arrive.

2.7 HTTP Message Format

The following is an example of HTTP request:

```
1  GET /doc/test.html HTTP/1.1
   \_/ \____________/ \______/
    |         |          |
   Method    URL       Version

----------------------------------------------------------------

2  Host: my-server.com
.  ...
5  User-Agent: Mozilla/4.0
   \_________/ \_________/
        |           |
       Name       Value

----------------------------------------------------------------

6
7  userName=Edoardo&userSurname=Riggio
```

Line 1: Request Line;
Lines 2-5: Header Lines;
Line 6: Blank line;
Line 7: Body.

The following is an example of HTTP response:

```
1  HTTP/1.1 200 OK
   \______/ \_/ \/
       |     |   Phrase
    Version Status

----------------------------------------------------------------

2  Date: Sat, 27 March 2021 15:51:20 GMT
.  ...
5  Content-Type: text/html
   \___________/ \_______/
         |           |
        Name       Value

----------------------------------------------------------------

6
7  <h1> Edoardo Riggio </h1>
```

Line 1: Response Line;
Lines 2-5: Header Lines;
Line 6: Blank line;
Line 7: Body.

The HTTP status codes are:

- **1xx** - Informational
- **2xx** - Success
- **3xx** - Redirection
- **4xx** - Client Error
- **5xx** - Server Error

2.8 Cookies

Since HTTP is a stateless protocol, in order to save info about the session we use cookies. In order to send and receive cookies, the following is done by the client and server:

1. The client sends an HTTP request to the server;
2. The server responds with a `Set-Cookie: xxxx` header;

3. The client then does another HTTP request, this time including the `Cookie: xxxx` header;
4. The server now responds with a normal HTTP response.

2.9 Web Caching

A **web cache** is a network entity that satisfies HTTP requests on behalf of an origin web server. The goal of this type of cache is to satisfy requests without involving the origin server, The cache acts as both client – for the main server – and server – for the clients. It is installed by the ISP.

Web caching is used in order to reduce the response time for the client requests and reduce the traffic on an institution's access link.

One problem that caches introduce are possible **stale objects**. HTTP can verify if it's actually stale with a **conditional GET**. This request includes the `If-Modified-Since: xxxx` header, so that the cache can always know if an object if stale or not.

2.10 SMTP

SMTP (Simple Mail Transfer Protocol) is a protocol that uses TCP to reliably transfer mail from client to server. Typically, a SMTP process is located on port 25 of a server.

An email server is directly sent from one SMTP server to another, no in-between server are used. SMTP is mainly a push protocol, and multiple objects in SMTP are sent in multipart messages (MIME – Multipurpose Internet Mail Extensions).

```
        +------------+           +-------------+
        |  Agent #1  | --SMTP--> |  Server #1  | -----+
        +------------+           +-------------+      |
                                                     SMTP
        +------------+    POP3   +-------------+      |
        |  Agent #2  | <--IMAP-- |  Server #2  | <----+
        +------------+    HTTP   +-------------+
```

2.11 Mail Access Protocols

Once the email has been delivered using SMTP, the user needs to access its mailbox in order to read emails. To do so, one of the following protocols is used:

- **POP3**

  The **POP3** (Postal Office Protocol) protocol allows a user to download emails from a server to its mail client. The emails can also be consulted while offline. Emails that are dowloaded are then removed from the server, thus cannot be accessed from multiple devices.

- **IMAP**

  The **IMAP** (Internet Mail Access Protocol) protocol allows a user to consult their emails from a remote server. By doing so, it can be accessed simultaneously by multiple clients.

- **HTTP**

  Mails are passed to the client using the **HTTP** protocol, rather than POP3 or IMAP. In this case emails are sent to the source server also using HTTP (transfer to destination service always happens through SMTP).

2.12 DNS

The **DNS** (Domain Name System) is a distributed database implemented in a hierarchy of DNS servers, and it is an application level protocol that allows hosts to query the distributed database.

The DNS protocol runs over UDP and uses port 53. In addition to translating hostnames into IP addresses, DNS provides a few additional services, such as:

- **Host Aliasing**

  A host with a complicated name could have one or more alias names. In that case, one of them is said to be the **canonical name**.

- **Mail Server Aliasing**

  Give a mail application the canonical address of a certain host.

- **Load Distribution**

  A set of IP addresses could be associated with the canonical hostname (for example when dealing with websites with lots of incoming traffic). The DNS in this case rotates the address every time a request is made.

2.12.1 DNS Servers Hierarchy

DNS uses a large number of servers displayed in a hierarchical fashion. There are three classes of DNS servers:

- **Root DNS Server**

  These servers provide the IP addresses of the TLD servers. There are over 400 root servers around the world.

- **TLD Server**

  A TLD (Top-Level Domain) sever manages a top-level domain (e.g. .it, .com, .net...). TLD servers provide the IP address for authoritative DNS servers.

- **Authoritative DNS Servers**

  Every organisation with publicly accessible hosts on the Internet must provide publicly accessible DNS records. These organisations can either create a personal authoritative DNS server, or use some service provider.

- **Local DNS Server**

  The local DNS server has a local cache of recent name-to-address translation pairs. This is the server where the user query first arrives.

2.12.2 DNS Records

The DNS servers that together implement the DNS distributed database store RRs (Resource Records). An RR is a four-tuple. For example:

```
(www.google.com, 74.125.131.103, A, 600)
 \____________/  \____________/  |  \_/
        |               |        |   |
       Name           Value    Type  TTL
```

There are four types of RRs:

- **Type = A**

  `Name` is the hostname and `Value` is the hostname's IP address. This is a standard hostname-to-IP address mapping.

- **Type = NS**

  `Name` is the domain and `Value` is the domain's hostname or an authoritative DNS server for the domain. This RR is used to route DNS queries further along in the query chain.

- **Type = CNAME**

  `Name` is the alias hostname and `Value` is the hostname's canonical hostname.

- **Type = MX**

  `Name` is the alias hostname and `Value` is the mail server hostname's canonical hostname.

2.12.3 DNS Messages

The DNS query and reply messages are structured as follows:

- **Identification**

  It is a 16-bit number that identifies the query. The same identifier is set to the corresponding answer.

- **Flags**

  There are a series of flags that identify the message. For example we have 1 bit describing if the message is a query or a response, 1 bit to identify if the server is an authoritative server...

- **Questions**

  Contains information about the query that is being made.

- **Answers**

  Resource records for the name that was queried. It can contain multiple RRs.

- **Authority**

  It contains records of other authoritative servers.

- **Additional Information**

  It contains additional information, for example info about the canonical hostname of a server in response to an MX or CNAME query.

2.13 Peer-to-Peer File Distribution

In the P2P paradigm, **peers** are pairs of intermittently connected hosts. These peers communicate directly with each other, and are not owned by any service provider. The following are the two approaches for distributing a file.

2.13.1 Client-Server Approach

- **Sending the File**

$$\frac{F}{u_s} \quad \text{(For 1 copy)}$$

$$N \cdot \frac{F}{u_s} \quad \text{(For N copies)}$$

- **Downloading a file**

$$d_{min} = \min\{d_1, d_2, ..., d_N\} \quad \text{(min download rate)}$$

$$\frac{F}{d_{min}} \quad \text{(min download time)}$$

- **Time to Distribute F to N Peers**

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

2.13.2 Peer-to-Peer Approach

- **Sending the File**

$$\frac{F}{u_s}$$

- **Downloading a file**

$$u_s + \sum_{i=1}^{N} u_i \quad \text{(min download rate)}$$

$$\frac{F}{d_{min}} \quad \text{(min download time)}$$

- **Time to Distribute F to N Peers**

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

2.14 BitTorrent

BitTorrent is a communication protocol for P2P, which enables users to distribute data and electronic files over the internet in a decentralised manner. The BitTorrent network is composed of:

- **Trackers**

  These are servers that track peers participating in a torrent.

- **Torrent**

  This is a group of peers exchanging chunks of a file.

During the request process, the rarest missing chunks are requested first.

2.15 Video Streaming

Videos are pre-recorded and saved onto a server. Users can request these videos on-demand from their end systems. These videos are compressed to a desired bit-rate. The higher the bit-rate, the better the image quality. There are two main methods to encode videos:

- **Spacial Coding**

  This is when instead of sending N values of the same color, only two values are sent: the color value and the times it is repeated.

- **Temporal Coding**

  This is when instead of sending a complete frame $i + 1$ (where i is the current frame), only the differences from frame i are sent.

Here are some of the bitrates of different codings:

- **MPEG 1 -** 1.5 Mbps
- **MPEG 2 -** 3-6 Mbps
- **MPEG 3 -** 64 Kbps-12 Mbps

2.16 DASH

DASH (Dynamic Adaptive Streaming over HTTP) is an HTTP-based streaming protocol. In DASH, the video is encoded into several different formats, everyone with different bit-rates. The servers keeps each version, and offers a manifest file. In this file there are all the bit-rates offered by the server and the location of the corresponding video.

The client can choose the bit-rate which is suitable to its needs, namely the current bandwidth of the client.

2.17 CDN

A **CDN** (Content Distribution Network) manages servers located in multiple geographically distributed locations. It stores a copy of each video in these servers, and attempts to direct each user request to a CDN location that will provide the best user experience.

There are two main server placement strategies for CDNs:

- **Enter Deep**

  Deploy server clusters in access ISPs all over the world. The goal is to get as close as possible to the user. In this approach, maintaining and managing the server clusters is challenging and expensive.

- **Bring Home**

  Build larger server clusters at a smaller number of sites. These CDNs are usually placed in IXPs. This approach has a lower maintenance overhead.

3 Transport Layer

A transport-layer protocol provides for logical communication between the application processes running on different hosts. From an application's perspective, it is as if the two hosts were directly connected.

There are two types of protocol actions in end systems:

- **Sender**

  Breaks the application message into segments, which are then passed to the Network Layer.

- **Receiver**

  Reassembles the segments into messages, which are then passed to the Application Layer.

3.1 Constraints from the Network Layer

The **Network Layer** uses the **IP** (Internet Protocol) protocol. This protocol provides for the logical communication between hosts, where each host has a unique IP address.

The IP protocol is what's known as a best-effort delivery service. This means that there is no guarantees of in-order delivery or even delivery per se.

3.2 Transport Layer Protocol

The transport layer has two protocols:

- **TCP**

  TCP (Transmission Control Protocol) offers a reliable, in-order delivery. It also offers connection setup, flow control and congestion control.

- **UDP**

  UDP (User Datagram Protocol) offers an unreliable and unordered delivery.

Neither TCP nor UDP provide either delay or bandwidth guarantees. Instead, what they both provide is integrity checking and multiplexing-demultiplexing.

3.3 Multiplexing and Demultiplexing

Multiplexing and demultiplexing are used in order to extend the host-to-host delivery service provided by the network layer to a process-to-process delivery service for applications running on the hosts. The following are the ways multiplexing and demultiplexing are implemented:

- **Multiplexing at Sender**

  Data is handled from multiple sockets and adds a transport header. This is later used by the demultiplexer.

- **Demultiplexing at Receiver**

  Use the header info from the multiplexer in order to deliver receive segments to the correct sockets.

3.3.1 Demultiplexing

The host receives the IP datagrams. These datagrams have a source and destination IP address, one transport-layer segment and a source and destination number.

The host uses the IP address and port numbers in order to direct segments to the appropriate socket.

3.3.2 Connectionless Demultiplexing

When creating a segment to send into a UDP socket, the sender must specify the following:

- **Destination IP Address**
- **Destination Port Number**

When the receiving host receives the UDP segment, it does two things:

- Checks the destination port number in the segment
- Directs the UDP segment to the socket with that port number

All UDP datagrams with the same destination port number will be directed to the same socket at the receiving host.

3.3.3 Connection-Oriented Demultiplexing

A TCP connection is defined by a 4-tuple:

- **Source IP Address**
- **Source Port Number**
- **Destination IP Address**
- **Destination Port Number**

In the demultiplexing process, the receiver uses all four values of the tuple to direct the segment to the appropriate socket.

Servers may also support may simultaneous TCP connections. Each connection is identified by its own 4-tuple, and to each of these connections is associated a different connecting client.

3.4 UDP

There are several peculiarities of UDP, such as:

- No connection is established (a connection would add RTT delay)
- There is no connection state neither at the sender nor at the receiver
- It has a very small header (8 bytes)
- No congestion control

The following are the headers of a UDP segment:

- **Source Port**
- **Destination Port**
- **Length**

  This field specifies the total number of bytes in the UDP header (header + data)
- **Checksum**

  It is used to verify the integrity of the UDP segment by revealing bit-flips.

3.4.1 UDP Checksum

The checksum is treated in different ways by both server and receiver.

- **Server**

  It treats the UDP segment fields contents as sequences of 16-bit integers. These integers are all added up, and the final checksum is calculated as the one's complement of the actual sum. Finally the result is stored inside of the checksum field of the UDP segment.

- **Receiver**

  All of the segment's fields are summed up and it is added to the sender's checksum. If any bit with value of 0 is detected in the final sum, then there is an error in the packet. The packet is thus discarded by the receiver.

```
- Sender

  Source Port:       0110011001100000 +
  Destination Port:  0101010101010101 +
  Length:            1000111100001100 =
                     ------------------
                     0100011011000001
                             |
                             | 1’s complement
                             V
                     1011100100111110

- Receiver

  Source Port:       0110011001100000 +
  Destination Port:  0101010101010101 +
  Length:            1000111100001100 =
                     ------------------
                     0100011011000001 +
  Sender Checksum:   1011100100111110 =
                     ------------------
                     1111111111111111   <- The checksum is valid
```

3.5 Reliable Data Transfer

Reliable data transfer is when no transferred data bits are corrupted or lost, and all transferred data bits are delivered in the order they where sent in.

TCP offers reliable data transfer.

3.5.1 Rdt 2.0

The underlying channel in charge of sending packets may flip bits while transmitting. In order to recover from such bit-flips, we use:

- **ACKs (Acknowledgments)**

  The receiver specifically tells the sender that the packet arrived correctly.

- **NAKs (Negative AcKnowledgments)**

  The receiver specifically tells the sender that the packet arrived with errors. The sender needs to retransmit the package.

This version of the protocol uses a **Stop & Wait** technique, meaning that the sender sends the packet and waits for a response to arrive.

3.5.2 Rdt 2.1

In this version of the protocol a sequence number is added to the packet. By doing so, the infrastructure has to remember whether the expected packet should have a sequence number of 1 or 0. Furthermore, it needs to check is the packet that was sent is a duplicate or not.

This protocol still uses ACKs and NAKs in order to check if the packets are corrupted.

3.5.3 Rdt 2.2

This protocol has the same features that version 2.1 has. In this case, though, the protocol only uses ACKs. Thus the receiver sends back another ACK in the case that everything is OK, and the packet arrived correctly.

3.5.4 Rdt 3.0

By using this protocol we assume that the channel can also lode packets. In such cases retransmission is not enough. A **timeout** is introduced.

The sender waits a predetermined amount of time (the timeout) in order for it to receive the ACK. If after this amount of time no ACK has been received, then the packet is retransmitted.

The performance of this version is poor. This is because it spends more time waiting than working.

3.5.5 Rdt 3.0 with Pipelining

In pipelining operations, the senders allows for multiple yet-to-be-acknowledged packets to be sent.

This operation can increase by a significant amount the usage and performance of the infrastructure. This is where the **Go-Back-N** protocols come to play.

3.5.6 Go-Back-N

The **Go-Back-N** protocol allows the sender to transmit multiple packets without waiting for an acknowledgment.

The sender can have at most N unacknowledged packets in the pipeline. The sender holds a "window" of up to N consecutive transmitted but unACKed packets.

The following function is to indicate that *n* packets (*n* included) have to be all ACKed.

```
ACK(n)
```

The following function indicates that all packets with a higher sequence number than *n* in the window have to be retransmitted.

```
Timeout(n)
```

If a packet has been received out-of-order, it can either discard it or buffer it. The out-of-order packet is then reACKed based on which packet has the highest sequence number.

The main problem with GBN is that a packet error would cause the retransmission of many other packets. This is why **selective repeat** is used instead. In selective repeat the receiver has to individually acknowledge specific packets as they are received. After a time-out these unACKed packets are retransmitted.

3.6 TCP

TCP (Transmission Control Protocol) implements a reliable data transfer service. This includes services such as:

- **Error Detection**
- **Retransmission**
- **Cumulative Acknowledgments**
- **Timeouts**
- **Header Fields for Sequence Numbers**
- **Header Fields for Acknowledgment Numbers**

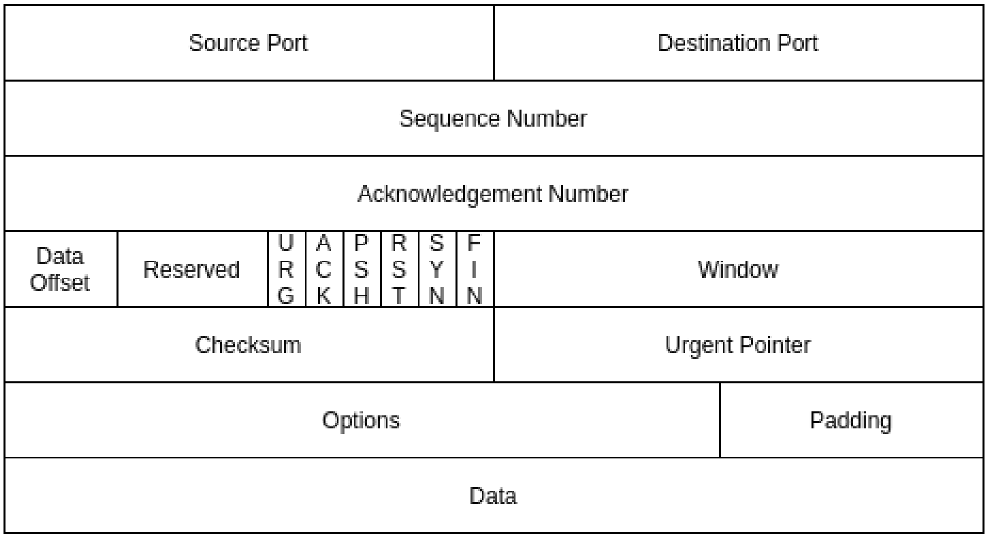
TCP is a **connection-oriented** protocol. This connection is full-duplex and point-to-point.

3.6.1 MSS and MTU

The **MSS** (Maximum Segment Size) is the maximum amount of application-layer data in the segment. The **MTU** is the Maximum Transmission Unit.

3.6.2 TCP Segment Structure

The following is the structure of a TCP segment:



<table>
<tr><td colspan="8">Source Port</td><td colspan="2">Destination Port</td></tr>
<tr><td colspan="10">Sequence Number</td></tr>
<tr><td colspan="10">Acknowledgement Number</td></tr>
<tr><td>Data
Offset</td><td>Reserved</td><td>U
R
G</td><td>A
C
K</td><td>P
S
H</td><td>R
S
T</td><td>S
Y
N</td><td>F
I
N</td><td colspan="2">Window</td></tr>
<tr><td colspan="8">Checksum</td><td colspan="2">Urgent Pointer</td></tr>
<tr><td colspan="9">Options</td><td>Padding</td></tr>
<tr><td colspan="10">Data</td></tr>
</table>

Some of the main sections are:

- **Sequence Number**

  It is the byte-stream number of the first byte in the segment.

- **Acknowledgment Number**

  It is put by the host and is the sequence number of the next byte the host is expecting from the other host in the connection.

TCP only acknowledges bytes up to the first missing byte in the stream. For this reason, TCP is said to provide **cumulative acknowledgments**.

TCP also uses timeouts in order to determine if a packet has to be retransmitted or not. The timeout value can be difficult to set. If it is too short, there is the risk of premature timeouts and unnecessary retransmissions. On the other hand, if the timeout is too long, there is a slow reaction to segment loss.

3.6.3 Estimating the RTT

In order to estimate the round trip time and find a suitable timeout value, we use the following formula:

$$RTT_{Est.} = (1 - \alpha) \cdot RTT_{Est.} + \alpha \cdot RTT_{Sample}$$

This is also known as the **EWMA** (Exponential Weighted Moving Average). Typically we have $\alpha = 0.125$.

3.6.4 Variability of RTT

In order to set the proper timeout value, we have to need a "safety margin". Mathematically, this is defined as:

$$Ti = RTT_{Est.} + 4 \cdot RTT_{Dev}$$

Where:

$$RTT_{Dev} = (1 - \beta) \cdot RTT_{Dev} + \beta \cdot |RTT_{Sample} - RTT_{Est.}|$$

Here typically we have $\beta = 0.25$.

3.7 TCP Flow Control

The **receive window** field inside of the TCP segment controls the sender. This means that the sender will never overflow the receiver's buffer by transmitting too much, too quickly.

3.8 TCP Connection Management

Before sending and receiving data, the sender and receiver handshake. This is done in order to agree to establish connection, and on the connection parameters.

Two-way handshakes do not always work in networks. There are several reasons why. For example there can be variable delays, some messages could be re-transmitted or there could be message reordering. In such cases an half-opened connection could be established. This is why a three-way handshake is instead needed.

3.9 Congestion Control

A congestion happens when too many senders are sending too much data at once. This could cause long delays and packet loss.

These are some congestion insights:

- Throughput can never exceed capacity
- Delay increases as the maximum capacity is approached
- Packet loss and re-transmission decreases the effective throughout
- Unneeded duplicates further decreases the effective throughput
- Upstream transmission capacity and buffering are wasted for packets lost downstream

3.9.1 End-to-End Congestion Delay

In order to mitigate congestion, TCP provides an **end-to-end congestion control**. In this case there is no explicit feedback from the network, and the congestion is inferred from the amount of packets that are lost and from the delay.

3.9.2 Network-Assisted Congestion Control

In the case of **network-assisted congestion control**, the routers provide direct feedback to sending and receiving hosts with flows passing through the congested router. This could indicate the congestion level or explicitly set the sending rate.

3.10 TCP Congestion Control

TCP implement what is known as **AIMD** (Additive Increase/Multiplicative Decrease). In this approach senders can increase the sending rate until packet loss occurs. When this happens, the sender will decrease the sending rate on loss events.

Additive increase will increase the sending rate by 1 maximum segment size every RTT until loss is detected. When loss is detected, we have a **multiplicative decrease**, where the sending rate is cut in half.

Loss is detected by the arrival of a triple ACK, and the rate is cut in half. When loss detection by timeout is detected, instead, will make sure that the rate is cut to 1 MSS (Maximum Segment Size).

A TCP connection is initially slow, with an initial cwnd of 1 MSS. For every

RTT where there is no loss, the cwnd is doubled. This is done by incrementing the cwnd by one for every ACK the sender receives.

4 Network Layer

The network layer's goal is to transport the segment from the sending to the receiving host. The network layer is present in every Internet device.

A **router**'s job is to examine the header files in all IP datagrams passing through it. After being examined, the datagrams are moved from input ports to output ports, in order to transfer them along their end-to-end path.

There are two key functions of the network layer, these are:

- **Forwarding**

  To move packets from a router input link to the appropriate output link.

- **Routing**

  To determine the route taken by packets from source to destination.

There are also two main elements in the network layer:

- **Data Plane**

  It is a local, per-router function. It determines how datagrams arriving in the router are forwarded to router output ports.

- **Control Plane**

  It is a network-wide logic. This determines how datagrams are routed among end-to-end paths from source to destination.

  There are two control plane approaches. The first involves **traditional routing algorithms**, and are directly implemented in routers. Another approach involves **SDN** (Software-Defined Networking), which is implemented in remote servers. Thus there are two kinds of control planes:

  – **Per-Router Control Plane**

    The individual routing algorithm components in each router interact inside of the control plane.

  – **SDN Control Plane**

    The remote servers compute and install the forwarding tables directly on the routers.

4.1 Network Service Model

There are no guarantees on the following:

- Successful datagram delivery to destination
- Timing or order of delivery
- Bandwidth available to end-to-end flow

It simply offers a best-effort service. The mechanism is fairly simple, sufficient provisioning of bandwidth allows performance of real-time applications to be ”good enough”. The replication of application-layer distributed services allow these services to be provided from multiple locations.

4.2 Architecture of a Router

The four core components of a router are:

- **Input Ports**
- **Output Ports**
- **Switching Fabric**
- **Routing Processor**

While the first three components are part of the **Forwarding Data Plane**, the last component is part of the **Control Plane**.

4.2.1 Input Ports

Input ports are composed of three layers:

- **Line Termination**

  It is the physical layer of the router.
- **Link Layer Protocol**
- **Lookup, Forwarding and Queuing**

  If the datagrams arrive faster than anticipated, then they will enter a queue. The header of the packet is used in order to lookup the output port – which is the second function of this layer.

  When looking in the forwarding table for a given destination address, the router uses the **longest prefix** that matches the destination address.

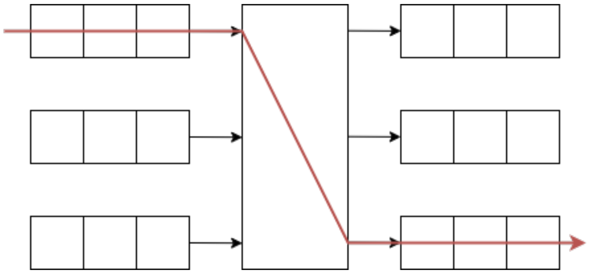
4.2.2 Switching Fabrics

The goal of the switching fabric is to transfer the packet from the input port to the appropriate output port.

The **switching rate** is the rate at which packets can be transferred from input to output ports.

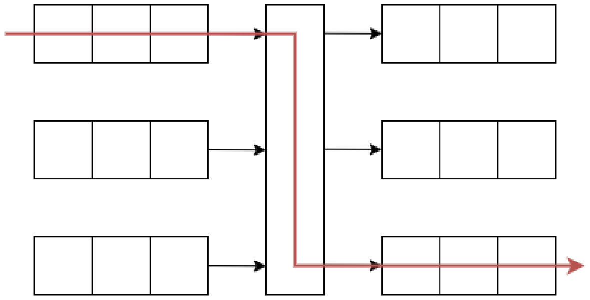
There are three main types of switching fabrics:

- **Memory**

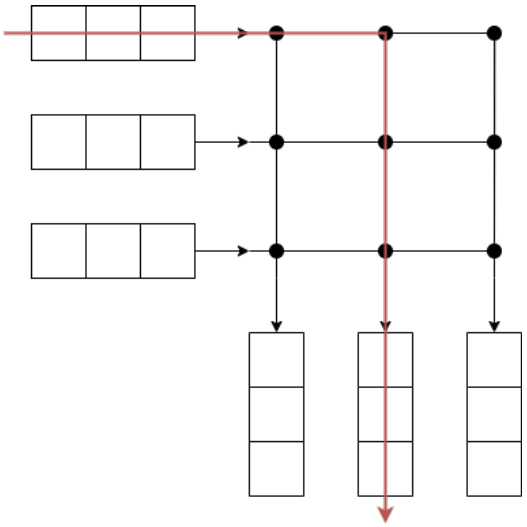
  The packet is copied to the system's memory, and then transferred to the correct output port. The speed is limited by the memory bandwidth and by its physical space.

- **Bus**

  The datagram is passed from the input port to the output port via a shared bus. The switching speed is limited by the bus' bandwidth.

- **Interconnection Network**



It is a $n \cdot n$ multiple-stage switch made up from smaller switches. In this case parallelism can be exploited for better performances.

4.2.3 Input Port Queuing

If a switch fabric is slower than the input and output port rate combined, queuing may occur at the input ports.

We have **HOL** (Head-Of-the-Line) blocking when queued datagrams at the front of the queue prevent others in the queue from moving forward. There are two steps in which the queues are handled:

- **Output Port Contention**

  Only the first in the queue is transferred, while the others are blocked.

- **One Packet Time Later**

  The next packet in the queue will suffer from HOL blocking.

4.2.4 Output Port Queuing

Buffering is required when datagrams arrive from the fabric faster than the link transmission rate. This could cause datagrams to be lost – congestion or lack of buffers could be the causes.
The scheduling discipline chooses among queued datagrams those to be transmitted – this is called **priority scheduling**. In order to calculate the buffering,

we can use the following formula:

$$Buf = RTT \cdot \frac{C}{N}$$

The router buffers drop packets whenever they are full. There are two ways packets can be dropped:

- **Tail Drop**

  Drop the incoming packet.

- **Priority Drop**

  Drop or remove a packet on a priority basis.

4.3 Packet Scheduling

The scheduling decides which packet to send next on the link. There are several scheduling algorithms, such as:

- **FCFS/FIFO**

  FCFS (FIst Come, First Served) is when packets are transmitted in order of arrival, to the output port.

- **Priority**

  The arriving packets are classified by any header field of the packet. The packets with the highest priority are sent first.

- **Round Robin**

  The arriving packets are classified by any header field of the packet. The server now cyclically scans the queues, sending on complete package from each input in turn.

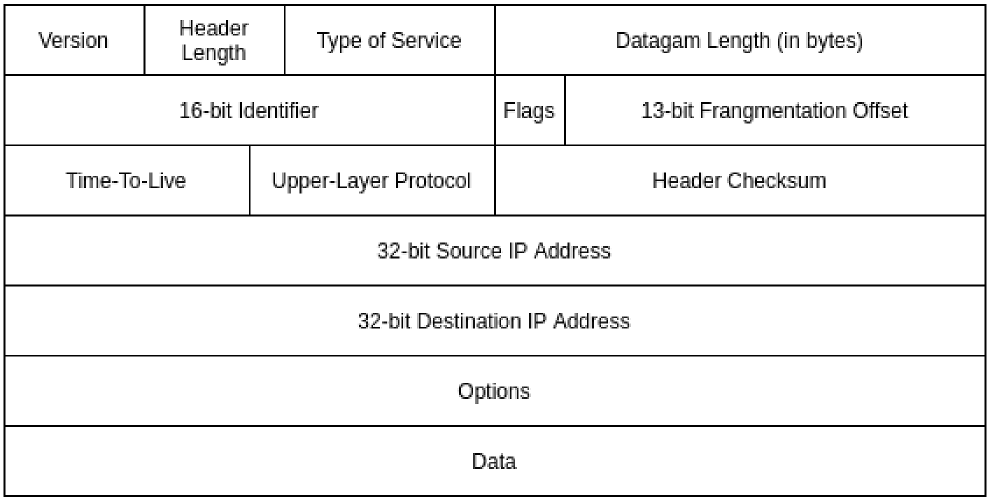
- **WFQ**

  WFQ (Weighted Fair Queuing) is a generalized Round Robin algorithm. Each input has a weight. This weight is calculated using the following formula:

  $$W_a = \frac{W_i}{\sum_i i}$$

4.4 IPv4 Datagram Format

The IPv4 datagram is composed by several field. The following is its visual representation:



<table>
<tr><td>Version</td><td colspan="2">Header Length</td><td>Type of Service</td><td colspan="2">Datagam Length (in bytes)</td></tr>
<tr><td colspan="4">16-bit Identifier</td><td>Flags</td><td>13-bit Frangmentation Offset</td></tr>
<tr><td colspan="2">Time-To-Live</td><td colspan="2">Upper-Layer Protocol</td><td colspan="2">Header Checksum</td></tr>
<tr><td colspan="6">32-bit Source IP Address</td></tr>
<tr><td colspan="6">32-bit Destination IP Address</td></tr>
<tr><td colspan="6">Options</td></tr>
<tr><td colspan="6">Data</td></tr>
</table>

Some of the fields are:

- **Version**

  Indicates the version of the IP protocol.

- **Type of Service**

  It indicates the Internet service quality selection (real-time or non-real-time)

- **Identifier**

  Uniquely identifies the fragments of a particular datagram.

- **Flags**

  It is composed of 3 bits. **Bit 1** must be 0, **Bit 2** indicates if the datagram may fragment (0 - Yes, 1 - No), and **Bit 3** indicates if this is the last fragment (0 - Yes, 1 - No).

- **Time-To-Live**

  It is decremented by one for every time that the datagram is processed by a router. If the TTL == 0, the datagram is dropped.

- **Upper-Layer Protocol**

  Indicates the protocol to which to pass the datagram at the final destination (e.g. 6 = TCP, 17 = UDP...).

- **Header Checksum**

  The Internet checksum of the datagram is computed by treating each 2 bytes in the header as a number. This must be recomputed and stored again at each router.

- **Destination IP Address**

  It is inserted by the sender of the datagram, and determined by the DNS.

4.5 IPv4 Addresses

There are 2^{32} possible addresses in this format. To create such addresses, we use the dotted-decimal notation. The following is an example:

```
11000001 | 00100000 | 11011000 | 00001001
\______/   \______/   \______/   \______/
   |          |          |          |
  193    .    32    .   216    .    9
```

4.6 Network Interface

The boundary between the host and the physical link is called an **interface**. An IP address is technically associated with an interface, rather than the host containing that interface.

Each interface on every router and host must have an IP address that is globally unique.

4.7 Subnet Mask

In order to determine the subnets, detach each interface from its host/router, creating islands of isolated networks. Each of these isolated networks is called a **subnet**.

4.8 IP Assignment

IP can be assigned using classful addressing – which is obsolete, or by **CIDR** (Classless InterDomain Routing).

When a host sends a datagram with destination address 255.255.255.255, the message is delivered to all hosts on the same subnet. This address is also commonly referred as the **broadcast address**.

4.8.1 Internet Number Registry System

The **ICANN** (Internet Corporation for Assigned Names and Numbers) distributes IP addresses. It also manage DNS root servers, assigns domains...

While router addresses are manually configured by network administrators, host addresses are configured using **DHCP** (Dynamic Host Configuration Protocol). The goal of DHCP is to dynamically assign an IP address to the host, the moment it joins the network. The following are the steps of the DHCP:

1. The host broadcasts a **DHCP discover message**.
2. The DHCP server responds with a **DHCP offer message**.
3. The host requests the IP address by means of a **DHCP request message**.
4. The DHCP server sends back the address by means of a **DHCP ACK message**.

A DHCP server can return much more than just the allocated IP address on the subnet. DHCP can also return:

- The address of the first-hop router for the client.
- The name and IP address of the DNS.
- The network mask.

4.8.2 Network Address Translation

Thanks to the NAT, as far as the rest of the Internet is concerned, on one IPv4 address is used by a local network.

All of the devices in the local network have a 32-bit address in a private IP space. Some of the possible masks are:

- 10/8
- 172.16/12
- 192.168/16

The NAT has several jobs, such as:

- **Replace Outgoing Datagrams**

  The source IP address of every outgoing datagram is repplaced by the NAT IP address. Same thing goes for port numbers.

- **Remember**

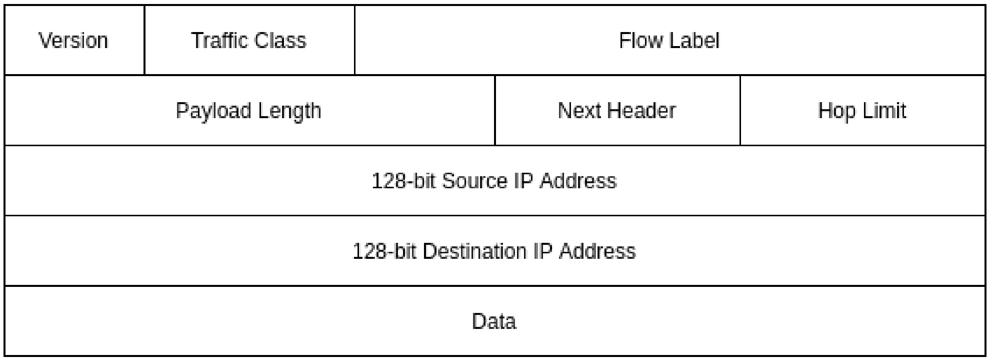
  It has to internally store a table containing every translation pair.

- **Replace Incoming Datagrams**

  The NAT IP address of every incoming datagram is replaced by the source IP address stored in the NAT. Same thing goes for port numbers.

4.9 Internet Protocol Version 6

IPv6 was created because of the exhaustion of 32-bit IPv4 addresses. The following is a visual representation of an IPv6 datagram:



<table>
<tr><td>Version</td><td>Traffic Class</td><td colspan="3">Flow Label</td></tr>
<tr><td colspan="3">Payload Length</td><td>Next Header</td><td>Hop Limit</td></tr>
<tr><td colspan="5">128-bit Source IP Address</td></tr>
<tr><td colspan="5">128-bit Destination IP Address</td></tr>
<tr><td colspan="5">Data</td></tr>
</table>

Some of the fields are:

- **Version**

  Version of the IP protocol.

- **Traffic Class**

  It is used to give priority to certain datagrams.

- **Flow Label**

  It identifies a flow of datagrams.

- **Next Header**

  Indicates the protocol to which to pass the datagram at the final destination.

- **Hop Limit**

  Same as TTL in IPv4 datagrams.

4.10 Transition from IPv4 to IPv6

The Internet can operate with mixed IPv4 and IPv6 router thanks to **tunneling**. Tunneling enables for IPv6 datagramsto be transmitted as the payload of IPv4 datagrams among IPv4 routers.

4.11 Generalized Forwarding

In order to forward packets, each router has a **forwarding table**. Routers also follow a **match+action** abstraction, meaning that they match the bits in the arriving packet, and then take action.

There exist two main types of forwarding:

- **Destination-Based Forwarding**

  The match and action is solely based on the destination IP address.

- **Generalized Forwarding**

  Many header fields can determine the action to be taken by the router. Many actions are possible, such as: drop, copy, modify, log the packet.

4.12 OpenFlow

OpenFlow is a communications protocol that gives access to the forwarding plane of a network switch or router over the network.

4.12.1 Flow Tables

The flow of the datagram is defined by the header field value of the datagram.

The actions of flow tables determine the processing that needs to be applied to a packet that matches a flow table entry. Some of the most important actions are: forwarding, dropping and modify fields.

4.13 Middleboxes

A middlebox is any intermediary box performing functions apart from normal, standard functions of an IP router on the data path between a source host and a destination host.

Examples of middleboxes are:

- NAT
- Firewall and IDS (Intrusion Detection System)
- Load balancers
- Caches

The services provided by the middleboxes can also be divided into three main categories:

- **NAT Translation**
- **Security Services**
- **Performance Enhancers**

Although middleboxes were initially proprietary, they have now moved towards **”whitebox” hardware** implementing an open API.

NFVs (Network Functions Virtualization) are programmable services that operate over whitebox networking.

4.14 Architectural Principles of the Internet

The goal of the Internet is connectivity, the tool is the Internet Protocol, and the intelligence is the end-to-end rather than hidden in the network.

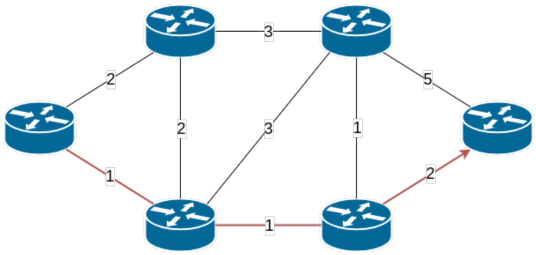
5 Control Plane

The main function of the control plane is **routing**. This means that it needs to determine the route that packets need to take to arrive to their destination.

5.1 Routing Protocols

Routing protocols are used to determine the best path, from sending to receiving host, through the network of routers.

A **path** is a sequence of routers, packets need to traverse from a given initial source host to a final destination host.



There are four types of routing algorithms:

- **Centralized**

  All routers have a complete topology and link cost information. These are the **link state** algorithms.

- **Decentralized**

  Iterative process of computation, where there is an exchanging of info with the neighbors. The routers initially only know the link cost to the attached neighbors. These are the **distance vector** algorithms.

- **Dynamic**

  Routes change more quickly. There are periodic updates in response to link cost changes.

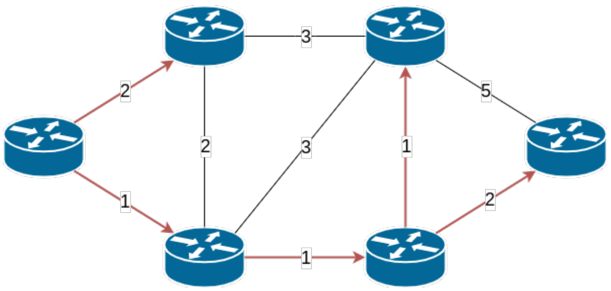
- **Static**

  The routes change slowly over time.

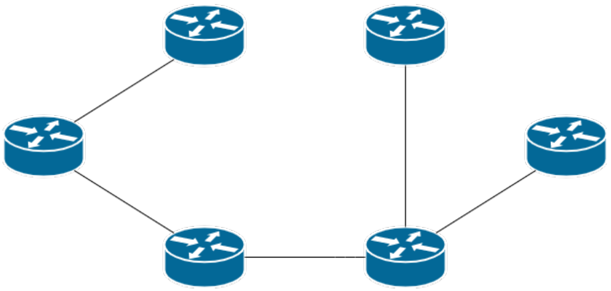
5.1.1 Dijkstra's Link-State Routing Algorithm

This is a **centralized** algorithm, which computes the least cost paths from one node to all other nodes.

For example, the following graph:



Would generate the following least-cost-path tree – from the left-most router:



5.1.2 Distance-Vector Algorithm

This algorithm is based on the Bellman-Ford equation. The equation is the following:

$$D_x(y) = \min_v \{c_{x,v} + D_v(y)\}$$

The cost of the least-cost path from x to y is calculated as the minimum – taken over all neighbors v of x – of the sum the direct cost of the link from x to v, plus the least-cost path from v to y.

This algorithm is **distributed**, as well as self-stopping, iterative and asynchronous.

Whenever a link cost changes, the node detects the change and recalculates the local DV. In the case that the DV changes, then the neighbors are notified.

5.2 Scalable Routing

Both LS and DV assume all routers are identical and that the network is flat. This is unrealistic in the real world, thus the solution is to group routers into **Autonomous Systems**.

An Autonomous System is a group of routers under the same administrative control. An algorithm running inside of an AS is called an **intra-autonomous system routing protocol**.

There are two types of routing – regarding scalable routing:

- **Intra-AS**

  Routing within the same AS.

- **Inter-AS**

  Routing between ASs.

5.2.1 Interconnected ASs

Being ASs interconnected, the forwarding table of each router is configured accordingly. **Intra-AS routing** determines the entries for destinations within the AS, while **Intra- and Inter-AS** both determine the entries for external destinations.

5.2.2 Routing Protocols

The most common Intra-AS routing protocols are the following:

- **RIP (Routing Information Protocol)**

  Here the DVs are exchanged every 30 seconds. It is no longer widely used.

- **EIGRP (Enhanced Interior Gateway Routing Protocol)**

  It is DV based, and formerly a Cisco-proprietary software for decades.

- **OSPF (Open Shortest Path First)**

  It is a link-state routing protocol. OSPF uses Dijkstra's algorithm in order to compute the forwarding table. All OSPF messages are authenticated.

  OSPF has two hierarchies: **local area** and **backbone**. Each node has a detailed area topology, and only knows the directions to reach other destinations.

- **BGP (Border Gateway Protocol)**

  This is a decentralized and asynchronous protocol, and is based on DV. BGP provides each AS means to:

  - Obtain subnet reachability information from neighboring ASs (**eBGP**).
  - Propagate reachability information to all AS-internal routers (**iBGP**).
  - Determine good routes to other networks based on the reachability information and policy.

  The BGP advertised route is composed of a **Prefix + Attributes**. The prefix is the destination being advertised. While the attributes are: **AS-PATH**, which is a list of ASs through which prefix advertisement has passed; **NEXT-HOP**, indicates a specific internal-AS router to the next-hop AS.

  Gateways receiving route advertisements use **import policies** to accept or decline paths.

  The BGP messages are exchanged between peers over a TCP connection. These messages are:

  - **OPEN**

    Opens TCP connection to remote BGP peer and authenticates sending BGP peer.
  - **UPDATE**

    Advertises a new path, or withdraws an old one.
  - **KEEPALIVE**

    Keeps the connection alive in absence of UPDATEs, it also ACKs open requests.
  - **NOTIFICATION**

    Reports errors in previous messages, it is also used to close opened connections.

The route selected is based on:

- Local preference value attribute (**policy decision**).
- Shortest AS-PATH.
- Closest NEXT-HOP router (hot potato routing)
- Additional criteria

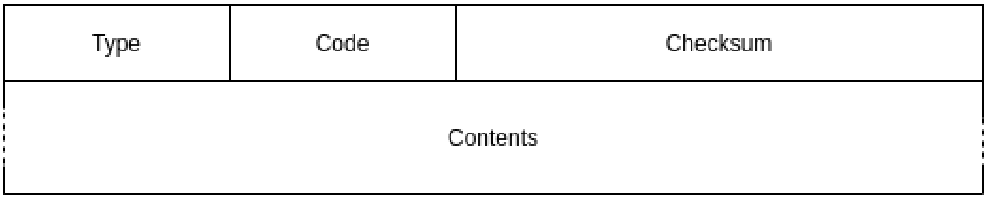
- **Hot Potato Routing**

  The gateway that has the least intra-domain cost will be chosen for routing. In this case there is no need to worry about the inter-domain cost.

5.3 ICMP

ICMP (Internet Control Message Protocol) is a protocol used by hosts and routers to communicate network-level information .

A visual representation of an ICMP message is the following:



Some fields are:

- **Code**

  The ICMP subtype

- **Contents**

  Copy of the IP header and at least 8 bytes of IP data.

6 The Link Layer and LAN

The basic service of the link layer is to move a datagram from one node to an adjacent node over a single communication link.

6.1 Definitions

A **communication network** is a system that allows two or more endpoints to be connected and to exchange data.

An **endpoint** is any kind of equipment that is able to connect to the network.

A **node** is any device that runs a link-layer protocol.

A **link** is a communication channel that connects adjacent nodes along the communication path.

6.2 MAC and LLC

MAC (Medium Access Control) and **LLC** (Logical Link Control), are two sublayers of the Data Link layer.

Most of the Link layer is implemented in hardware, while high-level functionalities may be implemented in software that runs on the host's CPU.

7 APPENDIX A - Formulae Glossary

- **a**

  Average rate of packets per second.

- **C**

  Link capacity.

- $\mathbf{D}_x\mathbf{(y)}$

  The cost of the least-cost path from x to y

- **d**

  Distance between the nodes. **Unit:** meters

- **F**

  File size in bits. **Unit:** bits

- **L**

  Length of a packet in bits. **Unit:** bits

- **N**

  Number of hops/peers/ports.

- **R**

  Transmission rate. **Unit:** bits/second

- **s**

  Propagation speed of the link. **Unit:** meters/second

- **T**

  Transfer rate. **Unit:** bits/second

- **u**

  Peer's upload rate. **Unit:** bits/second

- $\mathbf{u}_s$

  Server's upload rate. **Unit:** bits/second

8 APPENDIX B - Standard Ports

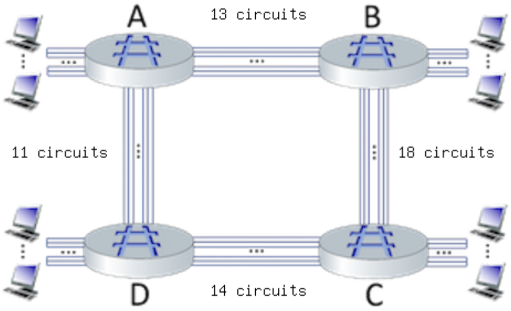
- **22** - SSH
- **23** - Telnet
- **25** - SMTP
- **53** - DNS
- **80, 8080, 8008** - HTTP
- **110** - POP3
- **143** - IMAP
- **465** - SMTP (secure)
- **853** - DNS (secure)
- **992** - Telnet (secure)
- **993** - IMAP (secure)
- **995** - POP3 (secure)

9 APPENDIX C - Exercises

9.1 Chapter 1

9.1.1 Circuit Switching

Consider the circuit-switched network shown in the figure below, with circuit switches A, B, C, and D. Suppose there are 16 circuits between A and B, 11 circuits between B and C, 14 circuits between C and D, and 13 circuits between D and A.



Questions and answers:

- What is the maximum number of connections that can be ongoing in the network at any one time? - **54 (it is the sum of all the circuits)**

- Suppose that these maximum number of connections are all ongoing. What happens when another call connection request arrives to the network, will it be accepted? - **No (There are no free circuits available)**

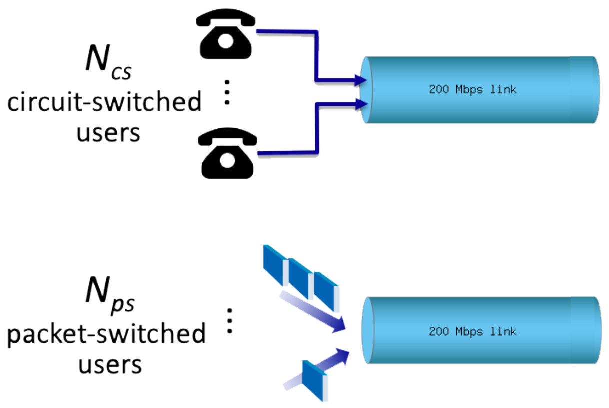
- Suppose that every connection requires 2 consecutive hops, and calls are connected clockwise. For example, a connection can go from A to C, from B to D, from C to A, and from D to B. With these constraints, what is the is the maximum number of connections that can be ongoing in the network at any one time? - **11 + 13 = 24 (which are the sums of the minimum between ADC and ABC)**

- Suppose that 12 connections are needed from A to C, and 12 connections are needed from B to D. Can we route these calls through the four links to accommodate all 24 connections? - **Yes (ADC = 12 connections occupied and 1 connection AB free; BCD = 11 connections occupied, BAD = 1 connection occupied)**

9.1.2 Packet Switching and Circuit Switching

A circuit-switching scenario in which N$_{cs}$ users, each requiring a bandwidth of 25 Mbps, must share a link of capacity 200 Mbps.

A packet-switching scenario with N_{ps} users sharing a 200 Mbps link, where each user again requires 25 Mbps when transmitting, but only needs to transmit 30 percent of the time.



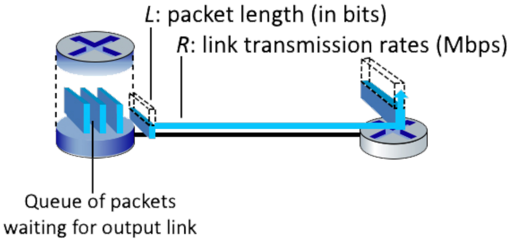
Questions and answers:

- When circuit switching is used, what is the maximum number of users that can be supported? - **200 / 25 = 8 (connection speed divided by the bandwidth for each user)**
- Suppose packet switching is used. If there are 15 packet-switching users, can this many users be supported under circuit-switching? - **No (15 > 8)**
- When one user is transmitting, what fraction of the link capacity will be used by this user? Write your answer as a decimal. - **25 / 200 = 1 / 8 = 0.125 (bandwidth divided by the connection speed)**

9.1.3 One-Hop Transmission Delay

Consider the figure below, in which a single router is transmitting packets, each of length L bits, over a single link with transmission rate R Mbps to another router at the other end of the link.

Suppose that the packet length is L = 4000 bits, and that the link transmission rate along the link to router on the right is R = 1000 Mbps.


L: packet length (in bits)
R: link transmission rates (Mbps)
Queue of packets waiting for output link

Questions and answers:

- What is the transmission delay? - **L / R = 0.000004 ms**
- What is the maximum number of packets per second that can be transmitted by this link? - **R / L = 250000 packets**

9.1.4 Queuing Delay

Consider the queuing delay in a router buffer, where the packet experiences a delay as it waits to be transmitted onto the link. The length of the queuing delay of a specific packet will depend on the number of earlier-arriving packets that are queued and waiting for transmission onto the link. If the queue is empty and no other packet is currently being transmitted, then our packet's queuing delay will be zero. On the other hand, if the traffic is heavy and many other packets are also waiting to be transmitted, the queuing delay will be long.

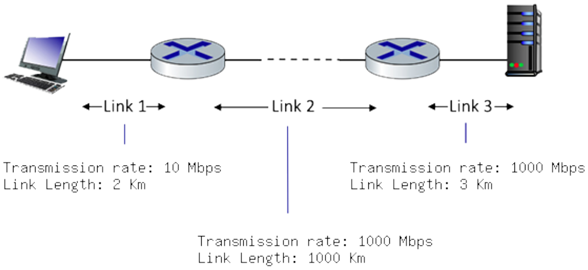
Assume a constant transmission rate of R = 1800000 bps, a constant packet-length L = 2400 bits, and a is the average rate of packets/second. Traffic intensity I = La / R, and the queuing delay is calculated as I(L / R)(1 - I).

Questions and answers:

- In practice, does the queuing delay tend to vary a lot? - **Yes**
- Assuming that a = 24000, what is the queuing delay? - **(La / R) * (L / R) * (1 - (La / R)) * 1000 = 0.032 * (2400 / 1800000) * (1 - 0.032) * 1000 = 0.0413 ms**
- Assuming that a = 84, what is the queuing delay? - **(La / R) * (L / R) * (1 - (La / R)) * 1000 = 0.112 * (2400 / 1800000) * (1 - 0.112) * 1000 = 0.133 ms**
- Assuming the router's buffer is infinite, the queuing delay is 0.1326 ms, and 1443 packets arrive. How many packets will be in the buffer 1 second later? - **a - floor(1000/delay) = 1443 - floor(1000/0.1326) = 0 packets**
- If the buffer has a maximum size of 706 packets, how many of the 1443 packets would be dropped upon arrival from the previous question? - **1443 - 706 = 737**

9.1.5 End-to-End Delay

Consider the figure below, with three links, each with the specified transmission rate and link length.


Link 1
Link 2
Link 3
Transmission rate: 10 Mbps
Link Length: 2 Km
Transmission rate: 1000 Mbps
Link Length: 3 Km
Transmission rate: 1000 Mbps
Link Length: 1000 Km

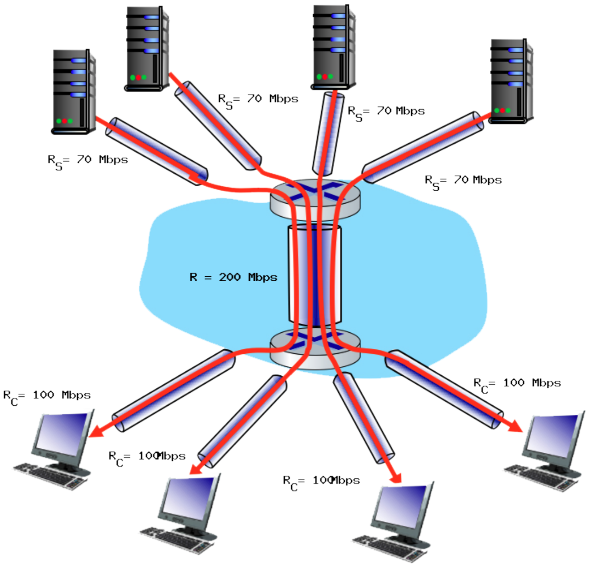
Consider the figure below, with three links, each with the specified transmission rate and link length. Assume the length of a packet is 4000 bits. The speed of light propagation delay on each link is 3x10^8 m/sec.

Questions and answers:

- What is the transmission delay of link 1? - **L / R = 0.004 / 1000 = 0.000004 ms**
- What is the propagation delay of link 1? - **d / s = 2000 / 300000000 = 0.00000667 ms**
- What is the total delay of link 1? - **0.000004 + 0.00000667 = 0.00001067 ms**
- What is the transmission delay of link 2? - **L / R = 0.004 / 1 = 0.004 ms**
- What is the propagation delay of link 2? - **d / s = 5000000 / 300000000 = 0.01666667 ms**
- What is the total delay of link 2? - **0.004 + 0.01666667 = 0.02066667 ms**
- What is the transmission delay of link 3? - **L / R = 0.004 / 1000 = 0.000004 ms**
- What is the propagation delay of link 3? - **d / s = 1000 / 300000000 = 0.00000333 ms**
- What is the total delay of link 3? - **0.000004 + 0.00000333 = 0.00000733 ms**
- What is the total delay? - **0.00001067 + 0.02066667 + 0.00000733 = 0.02068467 ms**

9.1.6 End-to-End Throughput

Consider the scenario shown below, with four different servers connected to four different clients over four three-hop paths. The four pairs share a common middle hop with a transmission capacity of R = 400 Mbps. The four links from the servers to the shared link have a transmission capacity of R$_S$ = 40 Mbps. Each of the four links from the shared middle link to a client has a transmission capacity of R$_C$ = 70 Mbps.



Questions and answers:

- What is the maximum achievable end-end throughput (in Mbps) for each of four client-to-server pairs, assuming that the middle link is fairly shared (divides its transmission rate equally)? - **50 Mbps**
- Which link is the bottleneck link? - **R$_S$**
- Assuming that the servers are sending at the maximum rate possible, what are the link utilisation for the server links (R$_S$)? - **R bottleneck / R$_S$ = 1**
- Assuming that the servers are sending at the maximum rate possible, what are the link utilisation for the client links (R$_C$)? - **R bottleneck / R$_C$ = 0.5714**
- Assuming that the servers are sending at the maximum rate possible, what is the link utilisation for the shared link (R)? - **R bottleneck / (R / 4) = 0.4**

9.2 Chapter 2

9.2.1 DNS – Basics

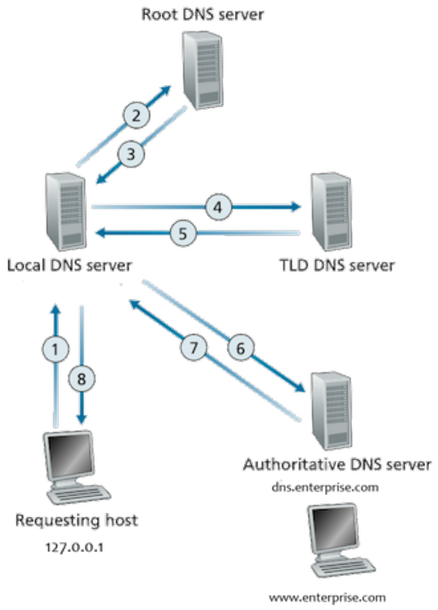
Imagine that you are trying to visit www.enterprise.com, but you don't remember the IP address the web-server is running on. Assume the following records are on the TLD DNS server:

- (www.enterprise.com, dns.enterprise.com, NS)
- (dns.enterprise.com, 146.54.102.239, A)

Assume the following records are on the enterprise.com DNS server:

- (www.enterprise.com, west3.enterprise.com, CNAME)
- (west3.enterprise.com, 142.81.17.206, A)
- (www.enterprise.com, mail.enterprise.com, MX)
- (mail.enterprise.com, 247.29.55.224, A)

Assume your local DNS server only has the TLD DNS server cached.



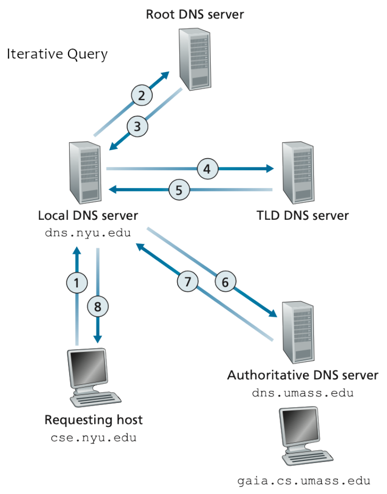
Questions and answers:

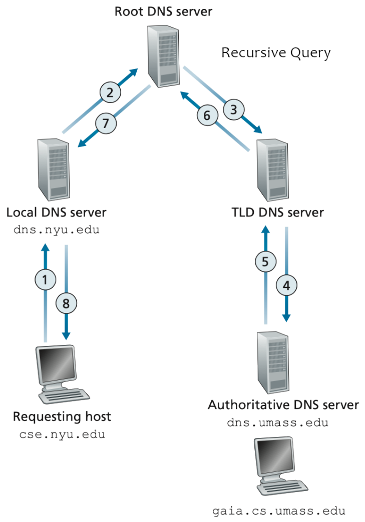
- What transport protocol(s) does DNS use: TCP, UDP, or Both? - **Both**
- What well-known port does DNS use? - **53**

- How many types of Resource Records (RR) are there? **4**
- Can you send multiple DNS questions and get multiple RR answers in one message? **- Yes**
- To which DNS server does a host send their requests to? **- Local DNS server**
- Which type of DNS server holds a company's DNS records? **- Authoritative DNS server**
- In the example given in the problem, what is the address of the DNS server for enterprise.com? **- dns.enterprise.com**
- When you make the request for www.enterprise.com, your local DNS requests the IP on your behalf. When it contacts the TLD server, how many answers (RR) are returned? **- 2 (a NS record, and an A record)**
- In the previous question, there were two responses, one was a NS record and the other an A record. What was the content of the A record? **- dns.enterprise.com, 146.54.102.239**
- Assume that the enterprise.com website is actually hosted on west3.enterprise.com, what type of record is needed for this? **- CNAME record**
- Now imagine we are trying to send an email to admin@enterprise.com, and their mail server has the address mail.enterprise.com. What type of record will we receive? **- MX record**
- In that MX record, what are the contents? **- www.enterprise.com, mail.enterprise.com**
- Does your local DNS server take advantage of caching similar to web requests? **- Yes (the retrieval of content is faster)**

9.2.2 DNS – Iterative vs Recursive

Assume that a user is trying to visit gaia.cs.umass.edu, but his browser doesn't know the IP address of the website. In this example, examine the difference between an iterative and recursive DNS query.





Questions and answers:

Iterative

- Between steps 1 and 2, where does the Local DNS server check first? - **DNS Root server**
- Between steps 2 and 3, assuming the root DNS server doesn't have the IP we want, where does the response link? - **DNS TLD server**
- Between steps 4 and 5, assuming the TLD DNS server doesn't have the IP we want, where does the response link? - **DNS Authoritative server**
- Between steps 6 and 7, the authoritative DNS server responds with the IP we want. What type of DNS record is returned? - **A**
- Which type of query is considered best practice: iterative or recursive? - **Iterative**

Recursive

- Between steps 1 and 2, where does the Local DNS server check first? - **DNS Root server**
- Between steps 2 and 3, where does the root DNS forward the request to? - **DNS TLD server**
- Between steps 4 and 5, where does the authoritative DNS forward the response to? - **DNS TLD server**
- In steps 6-8, the response is sent back in the reverse direction until it reaches the user. What type of DNS record is returned? - **A**

9.2.3 DNS and HTTP Delays

Suppose within your Web browser you click on a link to obtain a Web page. The IP address for the associated URL is not cached in your local host, so a DNS lookup is necessary to obtain the IP address. Suppose that four DNS servers are visited before your host receives the IP address from DNS. The first DNS server visited is the local DNS cache, with an RTT delay of RTT0 = 1 ms. The second, third and fourth DNS servers contacted have RTTs of 21, 1, and 11 ms, respectively. Initially, let's suppose that the Web page associated with the link contains exactly one object, consisting of a small amount of HTML text. Suppose the RTT between the local host and the Web server containing the object is $\text{RTT}_{HTTP} = 3$ ms.

Questions and answers:

- Assuming zero transmission time for the HTML object, how much time (in ms) elapses from when the client clicks on the link until the client receives the object? - **1 + 21 + 1 + 11 + 6 = 40ms**
- Now suppose the HTML object references 9 very small objects on the same server. Neglecting transmission times, how much time (in ms) elapses from when the client clicks on the link until the base object and all 9 additional objects are received from web server at the client, assuming non-persistent HTTP and no parallel TCP connections? - **1 + 21 + 1 + 11 + (10 * 6) = 94 ms**
- Suppose the HTML object references 9 very small objects on the same server, but assume that the client is configured to support a maximum of 5 parallel TCP connections, with non-persistent HTTP. - **1 + 21 + 1 + 11 + (3 * 6) = 52 ms**
- Suppose the HTML object references 9 very small objects on the same server, but assume that the client is configured to support a maximum of 5 parallel TCP connections, with persistent HTTP. **1 + 21 + 1 + 11 + 6 + (3 * 6) = 46 ms**
- What's the fastest method we've explored: Non-persistent-serial, Non-persistent-parallel, or Persistent-parallel? - **Persistent parallel**

9.2.4 Browser Caching

Consider an HTTP server and client as shown in the figure below. Suppose that the RTT delay between the client and server is 30 ms; the time a server needs to transmit an object into its outgoing link is 0.75 ms; and any other HTTP message not containing an object has a negligible (zero) transmission time. Suppose the client again makes 80 requests, one after the other, waiting for a reply to a request before sending the next request.
Questions and answers:

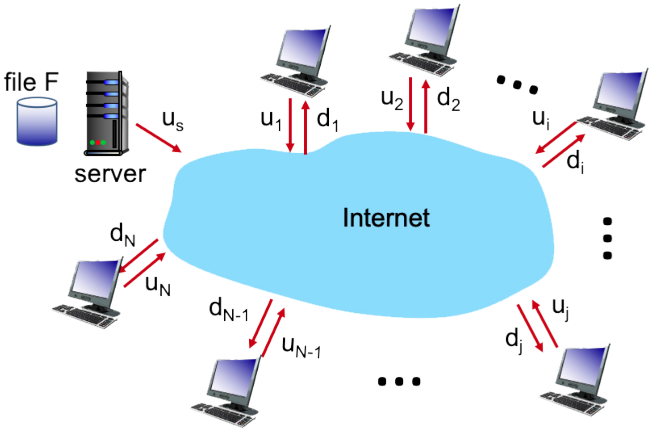
- How much time elapses (in milliseconds) between the client transmitting the first request, and the completion of the last request? - **(30 * 80) + (80 * ((100-40) / 100) * 0.75) = 2436 ms ((RTT * NUM-PACKETS) + (NUM-PACKETS * (PERCENT-NOT-CACHED / 100) * TRANS-DELAY)**

9.2.5 Client-Server and P2P File Distribution Delays

The problem is to distribute a file of size F = 6 Gbits to each of these 8 peers. Suppose the server has an upload rate of u = 72 Mbps.

The 8 peers have upload rates of: u1 = 16 Mbps, u2 = 20 Mbps, u3 = 24 Mbps, u4 = 19 Mbps, u5 = 21 Mbps, u6 = 25 Mbps, u7 = 15 Mbps, and u8 = 22 Mbps

The 8 peers have download rates of: d1 = 27 Mbps, d2 = 25 Mbps, d3 = 11 Mbps, d4 = 10 Mbps, d5 = 10 Mbps, d6 = 15 Mbps, d7 = 32 Mbps, and d8 = 33 Mbps



Questions and answers:

- What is the minimum time needed to distribute this file from the central server to the 8 peers using the client-server model? - **D**$_{CS}$ **= max{NF / us , F / dmin} = max{666.67 , 600} = 666.67 s**
- For the previous question, what is the root cause of this specific minimum time? Answer as 's' or 'ci' where 'i' is the client's number - **s**
- What is the minimum time needed to distribute this file using peer-to-peer download? - **D2P = max{F / us , F / dmin , NF / us + Σ(i=1, N) ui} = max{83.3, 600, 296, 296} = 600 s**

- For question 3, what is the root case of this specific minimum time: the server (s), client (c), or the combined upload of the clients and the server (cu)? - **c**

10 APPENDIX D - Useful Links

- **Interactive Exercises**: *interactive end-of-chapter exercises*
- **Unit converter**: *google unit converter*