Chapter 1

Definition 1.0.1: Key points

1. Networks allow computers to communicate information. Communication requires a shared medium, a common language, and a protocol.
2. Protocols are used at all network layers to define the structure, content, timing, and actions involved in communicating between systems.
3. Understand the difference between packet switching and circuit switching and the fundamental Internet components.
4. The Internet supports two transmission protocols: TCP (connection-oriented reliable service) and UDP (connectionless unreliable service).

Definition 1.0.2: Communication

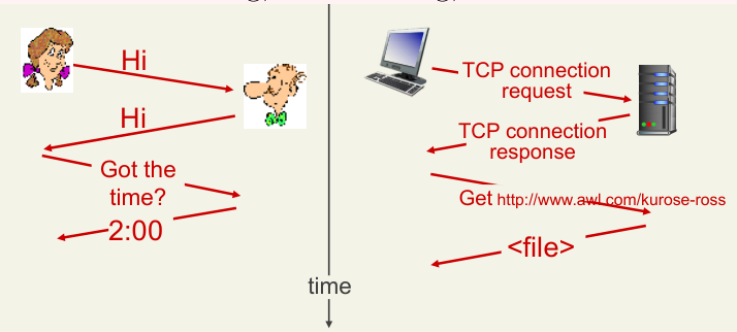
- The act of sending information from one party to another
- Sender transmits the info to one or more receivers
- For effective communication we need
  - Shared medium accessible to both sender and receiver
  - Language or encoding representing the information sent
  - Protocol or set of rules explaining how the medium is used by both the sender and receiver

Types of communication

- Broadcast communication: there is a single sender and multiple receivers
- Point-to-point communication: there is a single sender and single receiver

Definition 1.0.3: Protocol

Defines the format and order of messages exchanged between communicating parties and the actions associated with receiving, transmitting, and other events



Basically a ruleset, there are network protocols that function at various layers such as

- Physical layer protocols - define how bits are represented, encoded/decoded, and transmitted on the medium.
- Link layer protocols - define how bits are segmented and grouped into frames or packets.
- Protocols in upper layers define various services such as connections, error-handling, addressing, and routing.

Definition 1.0.4: TCP/IP (Transmission Control Protocol/Internet Protocol)

The structure/language and protocol used for communicating between computers on the internet
How it works

- Information is broken into sequential fixed-size units called **packets**
- Each packet has space for the unit of data, and also the destination IP address, and it's number in the sequence for reconstruction
- Packets are sent over the internet one at a time with whatever route is available, **this is called packet switching**
- Due to the fact that packets can take multiple routes, congestion and service interruptions do not delay transmissions

Network Services

- Client and server exchange transport-layer control messages before application data is transmitted (**handshaking**)
- Handshaking creates a TCP connetion between the sockets that is full duplex (both sending and receiving for both)
- Data transmission is reliable and in proper order
- Offers flow control so sender won't overwhelm receiver
- Congestion control: throttles sender when network is overloaded

Note:-

Standards for internet protocols are created by the **Internet Engineering Task Force (IETF)**. The documents are called **Requests for comments (RFCs)**

Corollary 1.0.1 User datagram protocol (UDP)

Provides connectionless service that performs no handshaking thus is faster than TCP, but also does not provide reliability, flow control or congestion control

Network Services

- No handshaking, unreliable transfer
- Packets may arrive out of order
- No congestion control, guarantee of transmission rate or delay, no flow control, no security, no connection setup

Theorem 1.0.1 Sending packets of data: host perspective

- Gets application message
- Breaks into packets of length L bits
- Transmits packets into access network at *transmission rate/link capacity,link bandwidth* R

$$\text{Packet transmission delay} = \text{Time needed to transmit L-bit to link} = \frac{L}{R\text{bits/sec}}$$

Corollary 1.0.2 Packet Switching: Store-and-Forward

- Transmission delay: takes L/R seconds to transmit L-bit packet into link at R bps
- Store and forwared: The entire packet must arrive at router before it can be transmitted on next link
- End-to-end Delay: $2\frac{L}{R}$(assuming zero propogation delay)

If the arrival rate is higher than the transmission rate of link then

- Packets will be queued and wait to be transmitted on link
- Packets can be dropped if memory buffer fills up

Packet Vs Circuit Switching

- Circuit switching: Networks reserve the resources needed for the duration of a communication before sending the data. These resources include buffers at switches and bandwidth on the links. Each communication link is divided among the number of circuits and the bandwidth is $\frac{1}{n}$ of the total bandwidth supported on the link. This is devided up either by a fixed value (FDM), or each circuit gets the full bandwidth for a set block of time and rotates between them (TDM)
- Packet switching: Networks don't reserve resources and the communicated data use the necessary resources on demand. Due to the store-forward method, there is a delay at *every link* the packet traverses; therefore the delay is $\frac{L}{R}$ bps. Also incurs **queue delays** when waiting for the buffer to free up, depends on network traffic; if very busy could result in packet loss

Note:-

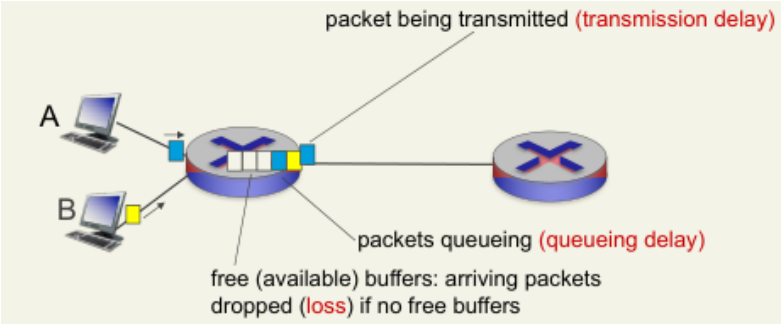
There are two main issues with circuit switching

1. They waste bandwidth and resources during idle times as each circuit is garanteed a dedicated bandwidth at all times whether it uses it or not

2. There is a significant overhead and complexity to setup circuits and the associated resources. This setup time introduces an initial delay that may be significant for short communications

1.1 Packet Delay and Loss

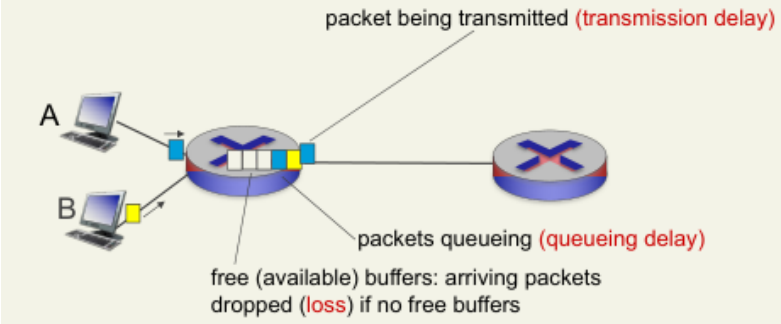
Figure 1.1: How packet loss and delay occurs



Sources of Packet delay

1. Processing delay: Time to read header and direct packet
2. Queuing delay: Time waiting in queue to be transmitted (depends on congestion/load)
   - It's traffic delay can be characterized by $\frac{La}{R}$ where a is the **average packet arrival time**
   - $\frac{La}{R}0$: Average queueing delay small
   - $\frac{La}{R} \longrightarrow 1$: Average queueing delay large
   - $\frac{La}{R} > 1$: More work than can be serviced at once
3. Transmission delay: Time to transmit packet on link
4. Propogation delay: Time to propogate from one end of link to the other (near speed of light)

Figure 1.2: Sources of packet delay



These delays add up to a total delay called $d_{\textbf{nodal}}$

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

- d_{proc}:
  - Checks for bit errors
  - Determine the output link

  - Typically under a milisec
- d_{queue}:
  - Time waiting for output link for transmission
  - Depends on congestion level of router
- d_{trans}:
  - L: Packet length (bits)
  - R: Link **bandwidth** (bps)
  - $d_{\textbf{trans}}=\frac{L}{R}$
- d_{prop}
  - d: Length of physical link
  - s: Propogation speed ($2*10^8 \frac{m}{\text{sec}}$)
  - $d_{\textbf{prop}}=\frac{d}{s}$

Theorem 1.1.1 Throughput

The rate $\frac{\text{bits}}{\text{time}}$unit at which bits transferred between sender and receiver

- Instantaneous: Rate at any given time
- Average: Rate over a longer period of time

Note:-

Bottlenecking: Our throughput is bottlenecked to what the **lowest throughput** of any link in the chain is

Definition 1.1.1: Layering

The modularization of features in complex systems by adding features through each "layer" which relies on the services provided by lower layers.
In reference to networking this is most commonly applied through the **Internet protocol Stack**

Corollary 1.1.1 Internet Protocol Stack

- Application: supporting network applications (transmitted data called messages)
  - FTP, SMTP, HTTP
- Transport: process-process data transfer (transmitted in segments)
  - TCP, UDP
- Network: routing of datagrams from source to destination (datagrams)
  - IP, routing protocols
- Link: data transfer between neighboring network elements (transmitted element is frame)
  - Ethernet, 802.111 (WiFi), PPP

- Physical: bits “on the wire”

Note:-

Protocol Data Unit: A PDU is a specific block of information transferred over the network that describes the different types of data being transferred

Applicantions use **Processes** to communicate with another machine, communicating to the process running on a different machine.

Definition 1.1.2: Sockets

Processes send and recieve information for applications through their **Sockets**.
Think of it like the two usb holes for a cable to connect the two and communicate data, the socket is where info is sent and received

Every message must have an *Identifier* of some kind, which both includes the IP address aswell as port numbers

Definition 1.1.3: Application Layer Protocol

How processes in a distributed application communicate by exchanging messages.

- Types of messages exchanged
- Syntax of each message type
- Semantics of each message and message fields
- Actions to perform when sending and receiving certain messages and the rules of communication

1.2 HTTP

Definition 1.2.1: HTTP Properties

Http uses TCP transport

- Creates socket on port 80
- Server accepts TCP connection from client
- Messages are exchanged between browser and server
- TCP connection closed

Note:-

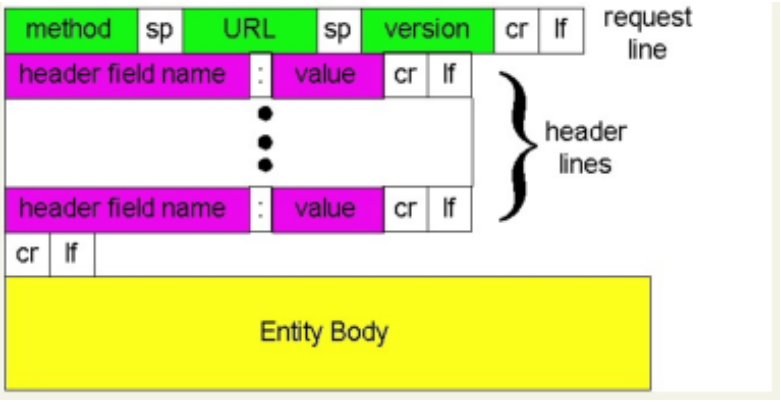
HTTP is stateless therefore the server maintains no info about past client requests

HTTP can be either persistent or not, difference being that non-persistence only sends *one* object over the TCP connection then closes it
As each new item requires a new connection, since we close the TCP connection after each object, we calculate time in terms of **round-trip time (RTT)**

Note:-

For non-persistent HTTP, response time is: $2RTT$+file transmission time

HTTP Request Message Format



Corollary 1.2.1 Method types

A method type is a particular type of interaction with the server that is specified in the method field

- Method types in HTTP/1.0: GET, POST, HEAD
- Method types in HTTP/1.1: GET, POST, HEAD, PUT, DELETE

Types

- GET: request file from server (no entity body)
- POST: request file from server (entity body has data)
- HEAD: asks server to leave requested object out of response
- PUT: uploads file in entity body to path specified in URL field
- DELETE: deletes file specified in the URL field

Corollary 1.2.2 Status codes

- 200 OK
- 301 Moved Permanently
- Bad Request: request message not understood by server
- 404 Not found
- 505 HTTP Version not supported

In earlier version of HTTP, due to the nature of the first-come-first-served resonses; small objects could get trapped behind bigger ones even if they're more essential. (Head-of-line blocking)
With the introduction of HTTP/2 transmission order was changed to be based on client-specified object priority

Definition 1.2.2: Simple Mail Transfer Protocol (SMTP)

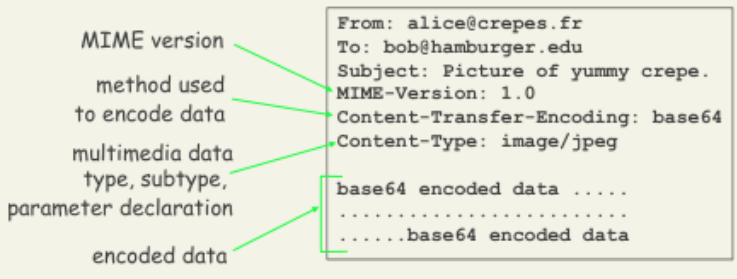
Uses TCP to transfer emails over port 25 and is a direct transfer from sender to receiver
Phases

1. Application-layer handshaking
2. Transfer of messages
3. Closure

Note:-

All commands and responses are in ASCII, messages are in 7-bit ASCII

Uses persistent connections. Uses CRLF to determine the end of message



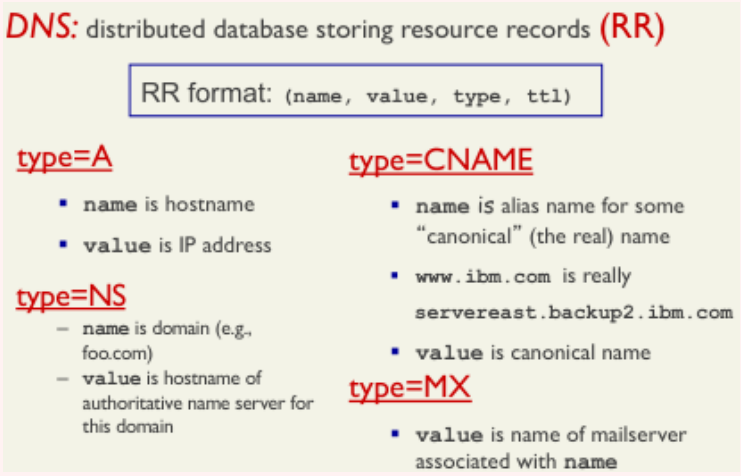
Definition 1.2.3: DNS

Associates unique host names to IP addresses for hosts. IPv4 Addresses are hierarchical and consist of 4 bytes. This is distributed across many name servers across the world.
An application-layer protocol that allows hots, routers, and name servers to communicate to resolve names.
Uses UDP port 53
As DNS is hierarchical, therefore higher-level domains (com,org,uk,ca) are responsible for everything else within their lower-level domains.
Sometimes within an organization they use their own DNS servers to provide authoratative hostnames for IP mappings

Note:-

Local domain name server: Default name server that ISP assigns

Once any name server learns a mapping it caches it as it's likely to be requested again sometime soon, expires after a timeout



What do you do when you need to stream videos that can't be embedded in the page?

Definition 1.2.4: Dynamic,Adaptive Streaming over HTTP (DASH)

Server:

- Divides video file into multiple chunks
- Each chunk stored, encoded at different rates
- *manifest file*: provides URLs for different chunks

Client

- Periodically measures server-to-client bandwidth
- Consulting manifest, requests one chunk at a time
  - Chooses maximum coding rate sustainable given current bandwidth
  - Can choose different coding rates at different points in time (depending on available bandwidth at the time)
- **When** to request chunk (such that buffer starvation, or overflow does not occur)
- **What encoding rate** to request
- **Where** to request chunk (can request from URL server that is "close" to client or has high available bandwidth

Streaming video = encoding + DASH + playout buffering

Definition 1.2.5: CDNs

Content Distribution Networks are a distributed set of servers in which are used as a choice for the closest, and therefore best, server to deliver the video. May change its path throughout the video.