Computer Networks

Diego Vicente Martín

November 11, 2017

Contents

1 Introduction **2**
- 1.1 What is the Internet? . . . 2
- 1.2 Network Core . . . 2
- 1.3 Delay, Loss & Throughput in Packet-Switched Networks . . . 3
- 1.4 Protocol Layers & Service Models . . . 4

2 Application Layer **5**
- 2.1 Principles of Network Applications . . . 5
- 2.2 DNS . . . 5

3 Transport Layer **6**
- 3.1 Transport Layer Services . . . 6
- 3.2 Multiplexing & Demultiplexing . . . 6
- 3.3 Connectionless Transport: UDP . . . 7
- 3.4 Principles of Reliable Data Transfer . . . 8
- 3.5 Connection-Oriented Transport: TCP . . . 9
- 3.6 Principles of Congestion Control . . . 10
- 3.7 TCP Congestion Control . . . 11

4 Network Layer **12**
- 4.1 Introduction . . . 12
- 4.2 Virtual Circuit & Datagram Networks . . . 12
- 4.3 What's Inside a Router . . . 13
- 4.4 IP: Internet Protocol . . . 14
- 4.5 Routing Algorithms . . . 16
- 4.6 Routing in the Internet . . . 18
- 4.7 Broadcast & Multicast Routing . . . 18

5 Link Layer & LANs **19**
5.1 Introduction 19
5.2 Multiple Access Protocols 20
5.3 Link-Layer Addressing 22
5.4 Ethernet 22
5.5 Link Layer Switches 23
5.6 Link Virtualization: ATM and MPLS 25

1 Introduction

1.1 What is the Internet?

- It is formed by millions of **computing devices** (hosts/end systems) running network apps, **communication links** (with a certain bandwidth) and **routers** that forward packets.
- We can also use the **nuts and bolts** view, where protocols are used for sending and receiving messages. Internet is a **network of networks**, loosely hierarchical. Some standards for the public Internet are defined in RFC (Request For Comments) and IETF (Internet Engineering Task Force).
- Similar to a human protocol, a machine **protocol** defines a format and order of messages sent among the network entities, along with actions taken on messages transmission an receipt.

1.2 Network Core

- Is a mesh of interconnected routers, where data is transferred by **circuit switching** or **packet switching**.
- **Circuit Switching**: Where some end-to-end resources are reserved (dedicated) for the connection.
  - This results in a complete use of the link bandwidth and switch capacity, plus a circuit-like (guaranteed) performance.
  - However, it requires a setup.
  - To obtain it, the **network resources are divided**. That way, we can allocate a call to each of the **pieces** (using frequency division or time division).

- **Packet Switching**: Where each end-to-end data stream is divided into **packets**.
  - User A and B share network resources but each packet is able to use the full link bandwidth.
  - That way the resources are used as needed.
  - Resource contention: The aggregate resource demand can exceed amount available. That causes congestion, where packets queue waiting for link use.
  - **Store and forward**: Packets move one hop at a time; the packet is completely received by the node before being forwarded. If we have a packet of length L and a link of rate R, the transmission delay is $d_{trans} = L/R$.
  - **Statistical multiplexing**: The bandwidth is not shared in a fixed pattern, instead, it is shared on demand.
- Packet switching allows more users to use the network. Furthermore, it is more suitable for bursty data since there is no call setup needed. The main drawback is the packet delay and loss: **we need protocols to ensure reliable data transfer and congestion control**.
- The internet is roughly hierarchical:
  - **Tier-1 ISPs**, that provide national and international coverage.
  - **Tier-2 ISPs**, that connects to one or more tier-1, possibly other tier-2. They pay to tier-1 ISPs to be connected to the rest of the Internet.
  - **Tier-3 ISPs**, which are the closest to end systems. They are customer so higher tier ISPs to obtain connectivity

1.3 Delay, Loss & Throughput in Packet-Switched Networks

- There are 4 types of packet delays:
  - **Processing**: checking bit errors, determining the output link...
  - **Queueing**: waiting at output link for transmission, depending on congestion.
  - **Transmission**: Time to send bits into the link (L/R where L is the length of the message and R the rate.)

  - **Propagation**: Time to let the bits traverse the link (d/s where d is the length of the link and s the propagation speed).

- Thus, the nodal delay is $d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$
- Talking about queueing delay we have the traffic intensity, where a is the average packet arrival rate. If $La/R \sim 0$ the average queueing delay is small. When $La/R \to 1$ the delay becomes large, and for $La/R1$ it becomes infinite.
- The delays can be measured using the `traceroute` command.
- Another important measurement was the **throughput** (bits/time) at which bits are transferred between sender/receiver.
- Having 2 links, if the throughput of the first one is larger than the second, we suffer a **bottleneck**.
- In the internet, the per connection end-to-end throughput is $\min\{R_c, R_s, \frac{R_b}{n}\}$ in a connection with R_c to end-systems, R_s to server and R_b in a backbone with n connections.

1.4 Protocol Layers & Service Models

- In order to deal with complex systems, we explicit a structure that allows identification and relation of the system's pieces: a **layered reference model**. Modularisation makes it easy to maintain and update the system.
- We have then the **Internet protocol stack**:
  1. **Application**: network applications (FTP, SMTP, HTTP)
  2. **Transport**: process-to-process data transfer (TCP, UDP)
  3. **Network**: route datagrams for source to destination (IP).
  4. **Link**: data transfer between neighboring network elements (PPP, Ethernet).
  5. **Physical**: bits "on the wire".
- We can also take into account the **ISO/OSI reference model**, that adds presentation and session layer, that can be inferred from the Application layer from the stack.
- To go through the network layers require encapsulation.

2 Application Layer

2.1 Principles of Network Applications

- A network application can have different architectures

  1. **Client-Server Architecture**: Formed by a always-on, permanent IP address host (or server farms for scaling) and clients that communicate with the server.
  2. **Pure P2P Architecture**: There is not an always-on server, instead, the end-systems directly communicate. The peers are intermittently connected and can change IP addresses. It is highly scalable, but difficult to manage.
  3. **Hybrid Client-Server & P2P**: Used by some applications like Skype o instant messaging, where the communication between users is P2P but they use a centralized user for authentication or finding addresses.

- In a local application, processes communicate using inter-process communication in the OS. When the processes are in different hosts, they exchanges **messages**.

- The process can be a **client process** (if initiates the communication) or a **server process** (if it waits to be contacted). A P2P application have both.

- Each of the processes has an assignated **socket** that it uses to receive and send messages.

- However, to successfully use them, we need protocols that ensure certain aspects depending on our application, (i.e data loss, throughput, timing, security...)

- There are 2 services that we are going to study: **TCP** (reliable, in-order delivery) and **UDP** (unreliable, unordered delivery).

2.2 DNS

- The **Domain Name System** is a distributed database implemented in hierarchy of many name servers. It is an application layer protocol used to resolve names in communications (IP address & name translation).

- The hierarchy flows in query the root server, then the .com DNS server, then the amazon.com DNS server and then obtain the address for amazon.com.
- The DNS servers can be **Top-Level Domain** (in charge of com, org, net), **Authoritative DNS Servers** (web or mail for a organization), or a **Local Name Server** (doesn't belong to the hierarchy, insted its used as a proxy).
- The resolutions works as recursively when querying. TLD servers are cached in local name servers, so they are usually not accessed.
- DNS store in their database **Resource Records (RR)**, which have a format of name, value, type, ttl. There can be:
  - Type A when value is an IP address.
  - Type NS when value is the hostname of the authoritative server for the domain.
  - Type CNAME when name is the alias for a canonical name.
  - Type MX when the values is the mailserver associated to name.
- The **DNS Protocol** is used to query and reply messages with the same format (using a 16 bit header).

3 Transport Layer

3.1 Transport Layer Services

- The transport layer provides **logical communication** between app processes running in different hosts.
- These protocols run in end systems, where they break the message in segments (in the sender side) and reassembles the segments into messages that are passed to the app layer (in the receiver side).

3.2 Multiplexing & Demultiplexing

- In the send host, **multiplexing** is to gather data from multiple sockets and envelope it with a header that will be used to demultiplexing.

- In the receiver side, **demultiplexing** is to deliver the received segments to the correct sockets. To do this, the host uses the IP address and the port number stored in the datagram to direct the segment to the appropriate socket.
  - **Connectionless demultiplexing** creates sockets with port numbers. An UDP socket is identified by a destination IP and a destination port. When a host receives an UDP segment, checks the destination port and directs it to the destination socket, despite of the source address or port.
  - In **connection-oriented demultiplexing**, a TCP socket is identified by the desintation IP and port and the source IP and port. That way, the reciever directs to a certain socket taking into account the source, which lets the server host support many simultaneous TCP sockets.

3.3 Connectionless Transport: UDP

- "No frills", "bare bones" Internet transport. Segments may be **lost** or delivered **out of order**.
- However, it is **connectionless** (no handshaking to add delay), it is **simple**, has a **small segment header** (8 bytes) and there is **no congestion control needed**.
- It is often used for multimedia streaming apps, that are loss tolerant and rate sensitive. Other uses are DNS or SNMP.
- Reliability must be added at application layer: UDP requires an **application-specific error recovery**.
- UDP uses a **checksum** to detect errors like flipped bits in transmitted segments:
  - The sender computes the 1's complement sum of the segment contents (treated as 16 bit integers) and puts it in the checksum field.
  - The reciever computes it again and checks if the value is correct or not.

3.4 Principles of Reliable Data Transfer

- Important in application, transport and link layers, the characteristics of the unreliable channel will determine the complexity of the RDT protocol.
- We can say that the can be separated (in a simplification) in 4 steps:
  1. The **reliable send** (that wraps the segment in a RDT protocol).
  2. The **unreliable send** (that sends the aforementioned segment through the unreliable channel).
  3. The **reliable receive** (that checks the RDT protocol).
  4. The **delivery of data** to the upper layer.
- There are different RDT versions:
  - **RDT 1.0**: the underlying channel is reliable, thus the sender sends the data to the receiver.
  - **RDT 2.0**: the underlying channel may flip some bits in the packet, so we use a **checksum** to detect errors and ACKs or NAKs (acknowledgements or negative acknowledgements) as **receiver feedback**.
  - RDT 2.0 has flaws: what if ACKs/NAKs get corrupted? How can we handled duplicate packages resulting from errors? A solution is to stop and wait, or to add a **sequence number** to each packet (and ACK/NAK). This is known as **RDT 2.1**, but we have twice as many states and its more complex. Furthermore, the receiver cannot know if its last ACK/NAK was or not corrupted.
  - In **RTD 2.2**, we use a **NAK-free solution**: the receiver sends ACKs with the sequence number of the last packet correctly received. If the sender receives a duplicate ACK, it retransmits the packet.
  - In **RDT 3.0**, we also assume that the underlying channel can lose packets. The approach we follow is to implement a **timer** that states the time the sender must wait to receive the ACK. If no ACK is received, the sender retransmits. The receiver can receive duplicate packages, but sequence numbers already handle this.
  - Since the performance of RDT 3.0 is not acceptable, we can improve it using **pipelining** (multiple "in flight", yet to be acknowledged, packets), which will increase the utilization.

  - There are 2 pipelining protocols:
    * **Go-Back-N**: Sender can have up to N unacked packets in pipeline, and the receiver only sends cumulative ACKs (no ACK if there is a gap). The sender sets a timer for the oldest unacked packet and resends all the unacked packets if it times out. If it receives an unexpected package (out of order) it discards it (no receiver buffer).
    * **Selective Repeat**: Sender can have up to N unacked packets in pipeline, and the receiver sends ACKs for individual packets. The sender has a timer for each unacked packet and resends only that packet if necessary.

3.5 Connection-Oriented Transport: TCP

- TCP has the next characteristics:
  - **Point-to-point**: One sender, one receiver.
  - **Reliable, in-order byte steam**: No message boundaries.
  - **Pipelined**: RCP congestion and flow control set window size.
  - **Send & receive buffers**.
  - **Full duplex data**: bi-directional data flow in same connection (MSS: maximum segment size).
  - **Connection-oriented**: Handshaking before data exchange.
  - **Flow controlled**: Sender will not overwhelm receiver.
- **TCP header** contains optional fields, and its size may vary from 20-60 bytes.
- In TCP we use sequence numbers (first byte in segment) and ACKs numbers of the next byte expected from other side. ACKs are cumulative in TCP.
- The handling of out-of-order segments is not defined in TCP, is up to the implementor.
- Since it uses a timer, we need to define a **TCP timeout value**, longer than round trip time. If we set it too short we will suffer unnecessary retransmission and if we set it too long we will have slow reaction to segment loss. To estimate it, we will try to use a smooth approximation to the `SampleRTT` and a safety margin of deviation.

- To ensure a **RDT**, TCP uses a single retransmission timer in the pipeline. A retransmission can be triggered by a **timeout event** or a **duplicate ACK**.
- The sender detects lost segments via duplicate ACKs, and if a sender receives 3 ACKs for the same data, it supposes that segment after ACK is lost, so it triggers a **fast retransmit** (resend segment before timer expires).
- **More about this on 3.7**.
- As a **flow control**, TCP uses a **speed-matching service**: the send rate matches the receiver's drain rate. The spare room in the receiver's buffer is advertised in the field `RcvWindow` in the segments.
- As **connection management**, TCP uses a **three way handshake**.
  1. Client host sends `SYN` to server to specify the initial sequence number.
  2. Server host receives it, and replies with a `SYNACK` segment to allocate server buffers and specifies the initial sequence number for the server.
  3. Client receives it, and replies with an ACK segment that may contain data.
- To close a connection, client sends a `FIN` control segment to the server, that replies with an ACK, then closes connection and sends a `FIN`. Once it is acknowledged the connection is closed.

3.6 Principles of Congestion Control

- Congestion is when too many sources are sending to much data too fast for the **network** to handle. Since the network is the one involved, this is different from flow control.
- We can infer congestion if there is **packet loss** (due to buffer overflows) or **long delays** (due to queueing in the buffers).
- The costs of congestion are more work and multiple copies of a single packet, both due to unnecesary retransmissions. It also lead to wasted transmissions if the packets are dropped.
- There are 2 broad approaches towards congestion control:

  - **End-to-end**: no explicit feedback from the network, congestion is inferred from loss and delay in end-systems. This is the approach taken by TCP.
  - **Network-assisted**: routers provide feedback by a congestion bit and a explicit rate sender should send at.

3.7 TCP Congestion Control

- The approach taken by PCP is to increase the transmission rate (`CongWin = LastByteSent - LastByteAcked`), probing for usable bandwidth, until loss occurs. This is done by **additive increase** (increase the congestion window by 1 MSS per RTT until loss) and **multiplicative decrease** (cut the window in half when loss detected).

- We can roughly define the rate as the coefficient between `CongWin/RTT` Bytes/sec.

- Apart from AIMD, we also use **slow start**: when a connection begins, `CongWin = 1 MMS`. Then, since the bandwidth may be much larger than it, start to **increase the rate exponentially fast** until the first loss event.

- The sender perceives congestion when a loss event happen, that may be:

  - After **3 duplicate ACKs**, using **fast recovery** will set `CongWin' = 1/2 CongWin`, and also `Threshold = 1/2 CongWin`. After this, `CongWin` will grow linearly.
    - In **TCP Reno** we also must take into account the possibility of having **fast retransmit / fast recovery**, which would made `CongWin' = 1/2 CongWin + 3`, and increase it in 1 each duplicate ACK more received. Once a correct ACK is received, we exit fast recovery setting `CongWin' = Threshold` and growing linearly.
  - After a **timeout**, `CongWin` is set to **1MMS**, and then grows **exponentially (slow start) until** `Theshold`, then linearly. That is, also, `Threshold = 1/2 CongWin`.

- This distinction is done because a timeout indicates a more alarming congestion scenario than 3 duplicate ACKs.

- We also must take into account in cognestion the **TCP fairness**: if k TCP sessions share the same bottleneck link of bandwidth R, each session should have an average rate of R/k.
- We say that TCP is fair because when there are two competing sessions, additive increase gives slope of 1, as throughput increases; and multiplicative decrease decreases throughput proportionally.
- In this explanation we are assuming TCP Reno. We can also be using other forms of TCP like **TCP Tahoe**, which differs from Reno in its congestion avoidance: doesn't matter if the packet loss is a timeout or a triple ACK, it will set `CongWin = 1MMS` and `Threshold = 1/2 CongWin` in both cases. TCP Tahoe also lacks fast retransmit.

4 Network Layer

4.1 Introduction

- Network layer encapsulates segments into **datagrams** on the sending side, and makes sure that the segments are passed to the transport layer in the receiving side.
- Network layer protocols **are present in every host and router**. A router checks the header fields in all IP datagrams passing through it.
- There are 2 key functions in the network layer:
  - **Forwarding**: move a packet from a certain router input to the appropriate router output.
  - **Routing**: determine the route taken by the packets from source to destination.
- Before the datagrams flow, two hosts (and intervening routers) must **establish a connection**.
- The difference between network and transport layer is that the **network layer connects hosts** (and routers involved) and **transport layer connects processes**.

4.2 Virtual Circuit & Datagram Networks

- A Virtual Circuit network provides a connection service, while a Datagram provides a connectionless service.

- In **VC networks**, source-to-destination paths behave much like telephones circuits:
  - A **call setup** is required before data flow.
  - Each packet carries a **VC identifier** (rather than destination address).
  - Every router on the path **maintains state** for each connection.
  - Link and router resources **may be allocated to VC** (to make a predictable service).
- A VC consists of a path from source to destination, VC numbers and entries in forwarding tables along the path.
- The VC numbers can be changed on each link if there are new VC numbers in forwarding tables.
- In the other hand, **Datagram networks** doesn't need a call setup, the routers don't have end-to-end connections knowledge and packets are forwarded using the **destination host address**.
- In forwarding tables we usually use the **longest prefix match**.
- **Internet is a datagram network**, since it lets computers exchage data as an "elastic" service using smart end systems (computers) using many link types.
- **ATM is a VC network**, since it evolved from telephone and requires strict timing and reliability, using "dumb" end systems where the complexity relies inside the network.

4.3 What's Inside a Router

- Every router has 2 key functions: **routing** (RIP, OSPF, BGP) and **forwarding**.
- Routers use a **decentralized switching** to forward datagrams to the appropriate output port. The goal is to do it at line speed, but if datagrams arrive faster than the forwarding rate into the switching fabric we need to use queueing.
- There are 3 types of switching fabrics:

  - **Memory**: Traditional computers with switching under CPU. The packet is copied to the system's memory and its speed is limited by memory bandwidth.
  - **Bus**: Datagram goes from input to output via a shared bus. The switching speed is limited by the bus bandwidth (**bus contention**).
  - **Interconnection network**: Uses Banyan networks and other interconnections with advanced design that fragment the datagram into fixed length cells and switches them through the fabric.

- In the output ports we can find two more mechanisms:
  - **Buffering** mechanisms when the packets arrive from the fabric faster than they can be transmitted. If the buffer overflows we can lose packets. To choose the buffering size, we can use
    * The **rule of thumb** (RFC 3439) typical RTT times link capacity, $RTT \cdot C$ (i.e C=10Gps, RTT=250ms, buffer=2.5Gbit)
    * A more recent recommendation: for N flows, $\frac{RTT \cdot C}{\sqrt{N}}$.
  - **Scheduling disciplines** that can choose among queued datagrams for transmission.

- If the fabric is slower than input ports combined, queueing may occur at input ports. If a queued datagram at front of the queue precents other from moving forward, we have a **Head-of-Line (HOL) blocking**, that may lead to delay and packet loss due to input buffer overflows.

4.4 IP: Internet Protocol

- An **IP datagram** includes a **20 bytes header**, which using TCP leads to a 40 bytes + app layer overhead per datagram.

- Network links have **MTU** (maximum transfer units). Different links have different MTUs.

- If an IP datagram is larger thatn the MTU it is divided ("fragmented") within net, and one datagram becomes several datagrams that are reassembled at final destination.

- To fragment and reassemble, the IP header has a `fragflag=1` if the datagram has been fragmented and an `offset=185` if the first bit in the packet is the 1480 byte of the first segment (185=1480/8), for instance.

- IP uses **IP addresses** (IPv4), which are 32-bits identifier for hosts or routers interfaces (being an interfaces a connection with a physical link). In an IP address we can see a subnet part (high order bits) and a host part (lower order bits).
- A **subnet** is a group of interfaces that can reach each other without using a router. We can isolate networks "detaching" routers, and each isolated network is a subnet. Each subnet has a **subnet mask** in format /X, where X points how many bits out of a 32 bit address are dedicated to the subnet part.
- The **Routing Information Base (RIB)** is where all the IP routing information is stored. It is not used to forward datagrams, instead it is used to infer de **Forward Information Base (FIB)** which contains all the information necessary to forward IP datagrams.
- FIB contains the network prefix, the router port identifier (or local interface IP) and the next hop identifier. The RIB also contains the next hop information (one or many) and metrics (one per hop).
- A **forwarding table** maps each IP prefix to the next-hop link(s). It is a destination-based forwarding that indetifies the longest-matching prefix, then the next IP hop and then the outgoing link.
  - Note: in class exercises, the format of a forwarding table is Network Prefix (a.b.c.d/x) – Next Hop (a.b.c.d)
- A host gets an IP address using **Dynamic Host Configuration Protocol (DHCP)**, that dynamically gets the address from a server:
  - Host broadcasts `DHCP discover` message.
  - DHCP server responds with `DHCP offer`.
  - Host requests IP address with `DHCP request`.
  - DHCP server sends address with a `DHCP ack` message.
- The **Internet Control Message Protocol (ICMP)** is used by hosts and routers to communicate network level information (like error reporting or echo request/reply). It is used in commands like `ping` and `traceroute`.
- **Network Address Translation (NAT)** allows to translate the IP addresses from a subnet to a single subnet IP address that is used by the rest of the internet.

- This data is stored in a **NAT translation table** and stores (source IP, port) to (NAT IP, port) pair. The NAT router changes incoming and outgoing datagrams using this table.
- ISPs gets a block of addresses from the **Internet Corporation for Assigned Names and Numbers (ICANN)**, who allocates addresses, manages DNS and assigns domain names.
- Since IPv4 addresses are soon going to be completely allocated, we are forced to develop **IPv6**:
  - Uses 128 bits addresses and a 40 byte header.
  - Doesn't allow fragmentation.
  - Some header fields are removed (like `checksum`) and other ones added (`pri`, `flow label`). Options are allowed using the `next hdr` field.
  - It also includes **ICMPv6** with multicast group management funtions and additional message types (i.e. "Packet Too Big").
- Not all routers can be updraded simultaneously, so we u **tunneling**: IPv6 carried as payload in IPv4 datagram among IPv4 routers. (IPv6 inside IPv4).

4.5 Routing Algorithms

- There are 2 main classifications:
  - **Global information**, where all routers have complete topology and link cost info (link state algorithms); or **decentralized information**, where routers know physically-connected neighbours and uses and iterative process of computation to exchange info with them (distance vector algorithms)
  - **Static**, where routers change slowly over time; or **dynamic**, where routers change more quickly in periodic updates or in response to changes.
- A **link-state routing algorithm** is **Dijkstra's algorithm**, that uses an iterative approach based in:

$$D(v) = \min\{D(v), D(w) + c(w, v)\}$$

Being $D(v)$ the distance from source to v adjacent to w and not already in the set N', until all nodes are stored in set N'.

- The **distance-vector algorithm** is based in the Bellman-Ford equation:

$$d_x(y) = \min_v\{c(x,v) + d_v(y)\}$$

Being $d_x(y)$ the cost of least cost path from x to y, and being v all the neighbours of x. Each node x knows the cost to each neighbour $c(x,v)$ and the node stores its distance vectors $[D_x(y), \forall y \in N]$ and its neighbours' distance vectors $D_v = [D_v(y), \forall y \in N]$

- Distance-vector is **asyncronous**, when a node x receives a new distance vector from a neighbour, it updates its own using the Bellman-Ford equation. In the same way, a node notifies its neighbours of a change in its distance vector.
- Link cost changes can produce a **"count to infinity" problem**. One way of solving it is using **poisoned reverse**: if Z routes through Y to get to X, Z will tell to Y that its distance to X is infinity, to prevent Y routes to Z via X.
- Comparing both algorithms in message complexity and speed of convergence is tricky since DV varies both depending of the network. However, we can deduce that **LS is more robust than DV**, since the errors can propagate through the network in the latter.
- Both algorithms are assuming a flat network, which is not true in real life, where we use **hierarchical routing**:
  - We aggregate routers in **autonomous systems (AS)**, that will run the same intra-AS routing protocol and will have a **gateway router** that will be the link to other AS.
  - The gateway router must learn which destinations are reachable through which AS and propagate this reachability info to all routers in its AS.
  - If a subnet is reachable from various AS, the gateway router will send it to the closest AS available: this is **hot potato routing**.

4.6 Routing in the Internet

- There are different intra-AS routing protocols, also known as **Interior Gateway Protocols (IGP)**.
- **Routing Information Protocol (RIP)** is a distance vector algorithm:
  - Exchange DVs among neighbours every 30 seconds in **advertisements** (up to 25 destination subents per advertisement).
  - If no advertisement is heard after 180 seconds, the neighbour/link is declared dead: the routes via neighbour are invalidated, new advertisements are sent and the news are quickly propagated to the entire net (poison reverse is used to prevent ping-pong loops).
  - **RIP routing tables are managed by an application level daemon**. Advertisements are sent in UDP packets.
- **Open Shortest Path First (OSPF)**: is an open link-state algorithm:
  - OSPF advertisements carry one entry per neighbour router and are disseminated to the entire AS using **flooding**. They are carried directly over IP (rather than UPD/TCP).
  - Another features in OSPF are **security**, **multipaths**, **integrated uni- and multicast (MOSPF)** and **hierarchical OSPF** in large domains.
  - **Hierarchical OSPF** consists of a two-level hierarchy (local area and backbone that connects different AS)
- The point of having different intra-AS and inter-AS routings is due to policy, scale and performance reasons.

4.7 Broadcast & Multicast Routing

- **Broadcast Routing** delivers packets from source to all other nodes.
- Since source duplication is inneficient, we use **in-network duplication**:
  - **Flooding**: when a node recieves a broadcast packet, sends a copy to all neighbours. This may cause cycles and storms.

  - **Controlled flooding**: node only broadcasts a packet if it hasn't broadcasted it before. To do this he has to keep track of the broadcasted packets or use **revere path forwarding (RPF)**: only forward a packet if it arrived on shortest path between node and source.
  - **Spanning tree**: No redundant packets received by any node since the nodes are forwarded through a previously constructed spanning tree. This tree is constructed by making all nodes forward an unicast message to a **center node** until it arrives to a node already in the tree.

- In **multicast routing** the goal is to find a tree (or trees) connecting routers having local multicast group members. It can be **source-based** (different from each sender to receivers) or **shared** (used by all group members).

5 Link Layer & LANs

5.1 Introduction

- We have to take into account the basic terms: All hosts and routers are **nodes**, the channels between adjacent nodes are **links** and a layer-2 packet is a **frame**.

- The responsibility of the data-link layer is to transfer the datagram from one node to the adjacent one over a link.

- Some of the link layer services are:

  - **Framing**: encapsulate a datagram into a frame, adding a header and trailer. The **MAC addresses** are used in frame headers to identify source and destination.
  - **Reliable delivery between adjacent nodes**: as stated in chapter 3. Seldom used on low bit-error link, but wireless links have high error rates.
  - **Flow control**: pacing between adjacent sending and receiving nodes.
  - **Error detection**: errors are caused by signal attenuation, noise. If errors are detected, the sender is signaled for retransmission or the frame is dropped.

  - **Error correction**: Receiver identifies and corrects bit errors without resorting to retransmission.
  - **Half-duplex and full-duplex**: with half-duplex, nodes at both ends of link can transmit, but not at the same time.

- The link layer is implemented in each and every host's **adaptor** or **Network Interface Card (NIC)**, that attaches into host's system buses.

5.2 Multiple Access Protocols

- There are two types of links:
  - **Point-to-point**: for dial-up access, or link between Ethernet switch and host.
  - **Broadcast**: shared wire or medium.
- Two or more simultaneous transmission by nodes cause an **interference**, which can be a **collision** if a node receives two or more signals at the same time.
- A **multiple access protocol** is a distributed algorithm that determines how nodes share a channel (for instance, when a node can transmit).
- One of the main problems is that communication about channel sharing must use the channel itself.
- The **ideal mutliple access protocol** would be that M nodes can transmit at R/M bps through a channel of rate R bps.
- There are 3 classes of MAC protocols:
  - **Channel Partitioning**: divide the channel in smaller pieces (time slots, frequency, code...) and allocate a piece to node for exclusive use.
  - **Random Access**: channel is not divided, and allows collisions by providing a way to detect and recover from collisions. Each node can transmit at full channel data rate.
  - **"Taking turns"**: nodes take turns proportionally to what they are sending.

- **Time Division Multiple Access (TDMA)** is a channel partitioning protocol that lets the nodes acces the channel in "rounds", where each station gets a fixed length slot in each round. Unused slots go idle.
- **Frequency Division Multiple Access (FDMA)**: is a channel partitioning protocol that divides the channel spectrum into frequency bands, where each station gets a fixed frequency band. Unused bands go idle.
- **Slotted ALOHA** is a random access protocol which assumes all frames are the same size, and divides time into equal size slots (time to transmit a frame). Nodes start transmitting only on slot beginning, so it is **synchronous**. If 2 or more nodes transmit at the same time, all detect a collision: then, node retransmits frame in each subsequent slot with probability p until success.
  - A single active node can continuously transmit at full rate of channel, and it is highly decentralized and simple.
  - However, there are collisions wasting slots, apart from idle slots. Nodes may be able to detect collision in less time to transmit packet, and requires clock synchronization.
  - Furthermore, the efficiency in the best case scenario (p maximizes) is $\lim_{N\to+\infty} Np(1-p)^{(1-p)} = \frac{1}{e}$, approximately **37%**.
- **Pure (unslotted) ALOHA** is simpler, since it is **asynchronous**. When frame first arrives, it transmits immediately. However, collision probability increases, since frame sent at t_0 collides with other frames sent at $[t_0 - 1, t_0 + 1]$.
  - Its efficiency is $\frac{1}{2e}$, approximately 18%, even **worse that slotted ALOHA**.
- **Carrier Sense Multiple Access (CSMA)** is based in "listen before transmit" principle: if the channel is sensed idle, transmit entire frame. If the channel is sensed busy, defer transmission. Collisions may still occur since propagation delay may prevent two nodes from hearing each other.
- **CSMA/CD (Collision Detection)** makes collisions detected within short time, letting the nodes abort the transmissions and reducing channel wastage.

  - In wired LANs we can easily measure signal strengths, and compare transmitted and received signals.
  - In wireless LANs the received signal strength may be overwhelmed by local transmissions' strength.

- **Polling** is a "taking turns" protocol where a master node invites slave nodes to transmit in turns. It is used in "dumb" slave devices and concerns polling overhead, latency and a single point of failure (master).
- **Token passing** is a "taking turns" protocol where a control token is passed from one node to next sequentially. Concerns token overhead, latency and single point of failure (token).

5.3 Link-Layer Addressing

- To get a frame form one interface to another physically connected interface we usually need a **MAC address**, 48 bit long and burned in NIC ROM (sometimes software settable).
- MAC addresses allocation is administred by IEEE, and manufacturers buy portions of MAC address space to "assure" uniqueness.
- Each IP node has a **Address Resolution Protocol (ARP) table**, with IP/MAC address mappings for some LAN nodes with a certain Time-To-Live (TTL), after which the record will be forgotten (typically 20min).
- If A wants to send a datagram to B without his MAC address, A broadcasts (FF-FF-FF-FF-FF-FF) an **ARP query** packet containing B's IP address. Once B receives the ARP packet, replies (unicast) to A with its (B's) MAC address. Then, A saves IP/MAC pair in its ARP table until it times out.
- **ARP is plug-and-play**: nodes create their ARP tables without and administrator.

5.4 Ethernet

- Ethernet is the dominant wired LAN technology: cheap, first widely used and able to kept up with speed race (10 Mbps - 10Gbps).

- **Bus topology** was the most popular topology through 90s, whera ll nodes are in the same collision domain. Today **star topology** prevails, with an active **switch** in the center that "speaks" separately with each node (no collisions with other nodes).
- An **Ethernet frame** encapsulates an IP datagram with a 14 bytes header plus a preamble of 8 bytes used to synchronise receiver and sender clock rates. It also has a 4 bytes CRC to detect errors.
- Ethernet is **connectionless** (no handshaking needed) and **unreliable** (reliability is assumed to be provided by TCP).
- Ethernet's MAC protocol is **unslotted CSMA/CD**:
  - NIC only transmits if senses the channel idle.
  - If it detects another transmission while transmitting, aborts and sends a jam signal.
  - Then, enter **exponential backoff**: after the nth collision, chooses a random $K \in \{0, 1, 2 \ldots, 2^n - 1\}$.
  - Then waits $K \cdot 512$ bit times (1 msec fro 10 Mbps, i.e. for K=1023 is about 50 msec) and tries again.
- Efficiency of CSMA/CD goes to one as the max propagation delay between 2 nodes goes to 0 and the time to transmit a max-size frame goes to infinity. It has a much better performance that ALOHA, it is simple, cheap and decentralized.
- There are different Ethernet standards, with the same MAC protocol and frame format but different speeds and physical layer media.
- Ethernet 10BaseT also features **Manchester encoding**, where each bit has a transition and allow clocks in sending and receiving nodes to **synchronize** each other without a centralized global clock among nodes.

5.5 Link Layer Switches

- **Hubs** are physical-layer dumb repeaters where bits coming in one link go out all other links at the same rate. All nodes can collide to one another and lacks frame buffering and CSMA/CD (hosts NICs must detect collisions).

- **Ethernet switches** are link-layer devices, smarter than hubs, that are:
  - **Active**: they store and forward Ethernet frames, examining frames' MAC addresses, and selectively forwarding them to one or more outgoing links, using CSMA/CD.
  - **Transparent**: Hosts are unaware of switches.
  - **Plug-and-play, self learning**: switches don't need configuration.
- Switches allow **multiple simultaneous transmissions**: hosts hace dedicated connections to them, and switches buffer packets. Ethernet protocol is used in each incoming link, but there are no collisions since it is **full-duplex** (each link is its own collision domain). They are able to switch two different transmissions at the same time without collisions (impossible with a hub).
- Switches have a **switch table**, where each record has a MAC address of host, interface to reach it and a time stamp. To fill this table, the switch **learns** which has can be reached through which interface:
  - When a frame is received the switch learns the location of the sender.
  - If the frame destination is unknown, the switch **floods** (switches the frame to all links except the incoming one) and waits for an answer from the destination to learn its location.
  - If the frame destination is known, it performs a **selective send** through the adequate link.
- Switches can be connected together and they will work in the exact same way.
- Both **switches and routers are store-and-forward** (routers are network-layer and switches link-layer) and **both have forwarding tables** (routers fill tables with IP addresses using routing algorithms and switches learn MAC address forwarding using flooding).
- Switches supporting **Virtual Local Area Network (VLAN)** can create multiple VLANs over a single physical LAN.
- In a **Port-based VLAN**, switch ports are grouped so that a single physical switch operates as multiple virtual switches.

  - **Traffic isolation**: frames from a group of ports can only reach ports from that group (can also be defined based on MAC addresses).
  - **Dynamic membership**: ports can be dynamically assigned among VLANs.
  - **Forwarding between VLANS** is done via routing (just as with separate switches).
  - **Trunk port** is used to carry frames between VLANS defined over multiple physical switches. These frames must be 802.1Q VLAN frames.

- **802.1Q VLAN frames** add some VLAN ID and control information fields.

5.6 Link Virtualization: ATM and MPLS

- In **Cerf & Kahn's Internetwork Architecture**, there are:
  - Two layers of addressing: internetwork and local network.
  - The IP layer makes everything homogeneous at internetwork layer.
  - The underlying local network technology in invisible at internetwork layers: Looks like a link layer-technology to IP.

- The **Multiprotocol Label Switching (MPLS)** was intended to speed up IP forwarding by using a fixed length label (instead of IP address) to do forwarding.
  - Using a fixed length identifier is faster than a shortest prefix matching.
  - It is a VC-like approach.
  - However, IP datagram still keeps the IP address.

- A **MPLS capable router or label-switched router** forwards packets to outgoing interface based only label value (ignoring IP addresses with a separate MPLS forwarding table). The signaling protocol is needed to set up forwarding and allows things like **source-specific routing** (not possible with IP alone).

- MPLS is used for traffic engineering but it also must co-exist with IP-only routers.