Networking

Salvatore Andaloro

1 Introduction

1.1 Structure of the internet

Internet is made of hosts (devices that connect to the network) - network edge and ISPs - network core, which connect to each other and provide connectivity. ISPs are structured in a hierarchy:

1. Level 1 ISPs: only a handful, have global coverage, are tightly interconnected
2. Level 2 ISPs: have national coverage and connect to level 1 ISPs to provide global coverage
3. Level 3 ISPs: networks to which clients connect to (Access ISPs)

1.2 Packet transmission

1.2.1 Packet delay

There are 4 main reasons for packet delay:

1. Delay in processing at node: node requires time to verify the correctness of data and to determine the output link
2. Queuing delay: outgoing packets are put in a queue and have to wait their turn to be sent
3. Transmission delay: depends on size of transmission in bytes (L) and transmission speed in bit/s (R) => $t = \frac{L}{R}$
4. Propagation delay: depends on length of medium (d) and propagation time of medium (s) => $t = \frac{d}{s}$

Total delay:

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

1.2.2 Packet loss

Packets before being sent are put in a queue (buffer), which has a finite capacity. If the queue is full the packets get dropped. They can be then retransmitted by the sending node.

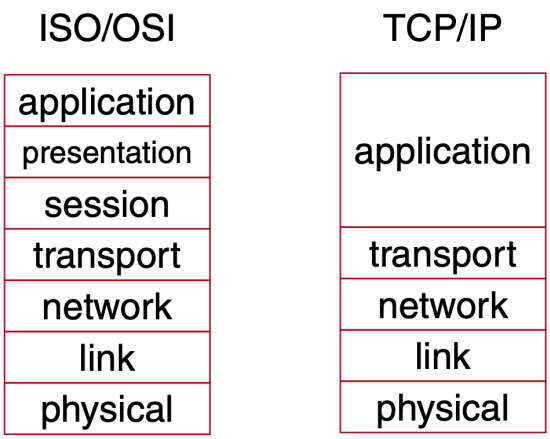
1.2.3 Throughput

Speed at which data is transmitted between sender and receiver (bits/time unit). Can be expressed as instantaneous (rate at a given point in time) or average over a period of time.

1.3 Protocols

Protocols define format and order of messages exchanged by the entities. More specifically they define the syntax of the messages (ex. their fields, order of bits), semantics (meaning of the messages), behaviour (ex. what to do if I lose a packet), temporisation (timing of acknowledgment).

Multiple protocols are grouped in so called **stacks**, defining a physical and logical communication architecture. There exist two main protocol stacks: the ISO/OSI stack and the TCP/IP stack. The ISO/OSI stack is just a reference model defining some fundamentals principles out of which real world models can be crafted. The TCP/IP stack is used in real world communication networks.



1.3.1 ISO/OSI structure

1. Physical: takes care of transmitting raw bits on the physical medium (electrical, electro-magnetic, light, sound)
2. Link: handles single hop communication and deals with error discovery and correction
3. Network: delivers packets from one host to another (deals with routing and congestion control)
4. Transport: performs application multiplexing/demultiplexing, segmentation/reassembling of data and deals with quality of service problems
5. Session: setups and maintains a communication between entities, therefore masks network interruptions from the transport layer
6. Presentation: data representation and encoding, encryption
7. Application: provides to applications the means to exchange data

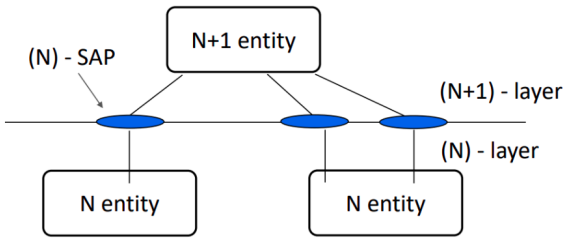
1.3.2 Layering

Layering protocols this way brings many advantages, namely:

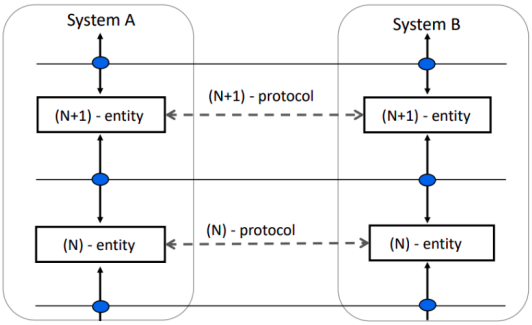
- Change of one layer independent from other layers
- Clearly defines what each component of the system is responsible for, therefore simplifying the overall analysis of the system, maintenance and updating

Each layer provides services to the layer above and sees the lower layer as a black box out of which it gets services from.

Communication between layers is provided through a programming interface called SAP (Service access point). On the other hand exchange of information between same layers is defined by protocols.



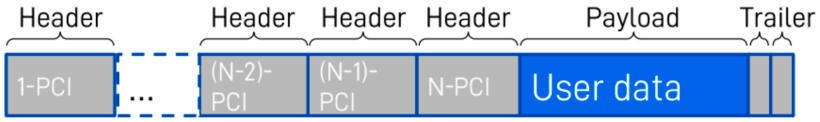
(a) *Comunicazione tra servizi di diverso livello*



(b) *Comunicazione tra servizi dello stesso livello*

1.3.2.1 Data units

Because of layering, we have to define how data travels through these layers. In a system with M layers, user data is packed in a M-SDU (Service data unit). Then layer M adds a M-CPI (Protocol control information). The final result is a M-PDU (Protocol data unit). The lower layer takes the M-PDU as a closed envelope and uses it as the (M-1)-SDU. On the receiving side the process is inverted, until we get the original data.



Data units can be

- segmented if data cannot be fit in a single SDU
- assembled to avoid inefficiencies, so multiple SDUs are grouped into one
- reassembled (opposite of segmentation)

Name of PDUs of the various layers:

- Application: messages
- Transport: segments
- Network: packets or datagrams
- Data link: frame

2 Application layer

2.1 Application architectures

2.1.1 Client server architecture

In the client-server architecture there exists an always on host, which usually has a fixed IP address to which all clients connect to obtain resources or to use a service. Clients talk to the server only when they need to, may have dynamically changing IP addresses and are not connected directly to each other.

Pros: easy to implement, clients don't need to be powerful

Cons: single point of failure -> if server goes down, everything goes down, complicated to scale

Figure 4: Client-server architecture

2.1.2 Peer to peer architecture

In the peer to peer architecture there is no central node, but all clients act as both servers and clients. In this architecture every connected node is called peer. Peers are intermittently connected and change IP addresses. Some P2P networks have some central nodes that act as directory nodes, which hold information about all peers connected to the network.

A P2P connection is implemented on a host by creating two processes: a client process which communicates with another peer and a server process which waits to be contacted.

Pros: self-scalability (if clients joins the network in brings with it more capacity)

Cons: difficult to implement

2.2 Sockets

A socket is an interface created by the application and managed by the operating system, through which the application can exchange messages with other processes. To communicate with other processes, it is necessary to know the IP address (network layer) and the port number (transport layer) of the other end. This couple is referred to as socket.

2.3 Application level protocols

The application layer protocols define types of messages exchanged, message syntax, message semantics (meaning of message), rules on when and how processes send and respond to messages. These specifications for open protocols can be found in RFCs to allow for interoperability between different applications.

For an application level protocol to work correctly the underlying layers may need to provide the following functionalities:

- Data integrity: some apps require 100% reliability (ex. file transfer), others not (ex. videoconferencing)
- Timing: some apps require low latency (ex. gaming)
- Throughput: some apps require a minimum amount of throughput (ex. streaming), others are flexible (ex. file transfer)
- Security: encryption, data integrity

2.3.1 HTTP

A web page consists of objects (HTML files, images, audio…). The HTML contains references to all other objects by means of their URL.

HTTP (Hypertext transfer protocol) is a protocol based on the client server model: the client uses the browser to request and receive objects from the server. The communication is done by using TCP. The client initiates a TCP connection on port 80, which the server then accepts. At the end of the communication then the connection gets closed. HTTP connections can be:
- Non persistent: this means that the connection if closed just after sending one object
- Persistent: multiple objects can be sent on a single TCP connection

RTT (round trip time): time for a small packet to travel from client to server and back

Non persistent HTTP requires 2 RTT per object, because each time a new connection has to be opened. This induces OS overhead, which has to constantly open new connections. Browsers may also decide to open multiple parallel TCP connections. On the other hand with persistent TCP all these problems vanish and we can get as little as 1 RTT per object. HTTP/1.0 used non persistent connections, while HTTP/1.1 uses persistent connections.

There are two types of HTTP messages: request and response.
- Request: contains the request method (GET, PUT, POST…) with the requested page, a header with the host and a body, lines are terminated with \r\n
- Response: contains status line (2xx - ok, 3xx - redirect, 4xx - client error, 5xx - server error)

HTTP is stateless, that means that the server maintains no information about past client requests. To simulate state we use cookies, which are small files generated by the server and stored by the client. The client then sends these cookie along with every request to the server.

2.3.1.1 Proxy server

A proxy server is a server between the client and the actual server. When the client asks for a resource the proxy server first checks if it can provide the resource itself, otherwise asks the server for the resource, stores it locally and returns it to the client. Usually proxy servers are installed by ISPs or by companies, universities.

To optimise link utilisation, client can use a conditional GET: specify **If-modified-since** field in header, if file modified after that date send new file, otherwise return just 304 Not Modified

Pros of caching: reduces response time for client request, reduce outgoing traffic

2.3.2 Electronic mail

In mail exchange there are 3 mail components:
- User agent: writes and reads emails
- Mail server: contains messages to be delivered to the user and queue of messages to send
- Transfer protocol: regulates the transfer of mail between client and server and between servers

2.3.2.1 SMTP protocol

The SMTP protocol is used to reliably transfer email messages by connecting directly the sending server and receiving server. The transfer is made of three parts: handshaking, transfer of messages and closure. Messages are encoded in 7 bit ASCII and there is a command/response interaction (like HTTP). Because multiple commands are sent in the same connection, the connection is persistent.

STMP interaction:
1. Alice sends email using STMP from her user agent to her mail server
2. Alice's mail server goes through the queue of email and using STMP connects to Bobs mail server and sends the email
3. Bob's mail server places the mail in Bob's mailbox, ready to be read

Commands: HELO, MAIL FROM, RCPT TO, DATA. Message is terminated with a dot and a new line.

2.3.2.1.1 Message format

In the Data field the sender can specify and additional header with the following fields: To, From, Subject. Note that the To field here doesn't actually send the message to those recipients, in is specified purely for the receiver's information.

There have been developed extensions to the standard which allow multimedia messages using MIME encoding. This encoding encodes data in base64.

2.3.2.2 Mail access protocols

While SMTP just deals with exchanging mail between server, there is also the need for a protocol to retrieve the received mail from the server. There have been developed a few of them:
- POP3 (Post Office Protocol): client authenticates itself and asks for mail stored on the server (and possibly deletes them in the process). Protocol is stateless
- IMAP (Internet Mail Access Protocol): keeps the emails on the server and allows the client to organise mail in folders, therefore keeps state between sessions
- HTTP: used by email web clients like GMail, Yahoo…

2.3.3 Domain Name System (DNS)

The internet gives access to any kind of resource, but it's hard to address these resources. A directory service enables to find resources based on a key-value system (like a phonebook). One implementation of a Distributed Directory Service (DDS) is the Domain Name System (DNS).

DNS allows to retrieve the IP address associated to a domain name. Its structure is hierarchical. DNS offers the following additional services:
- Host aliasing: it is possible to associate some aliases to a domain name (works also for mail), that is, multiple domain names point to the same IP
- Load distribution: DNS can return multiple IP addresses, swapping the order if one gets congested

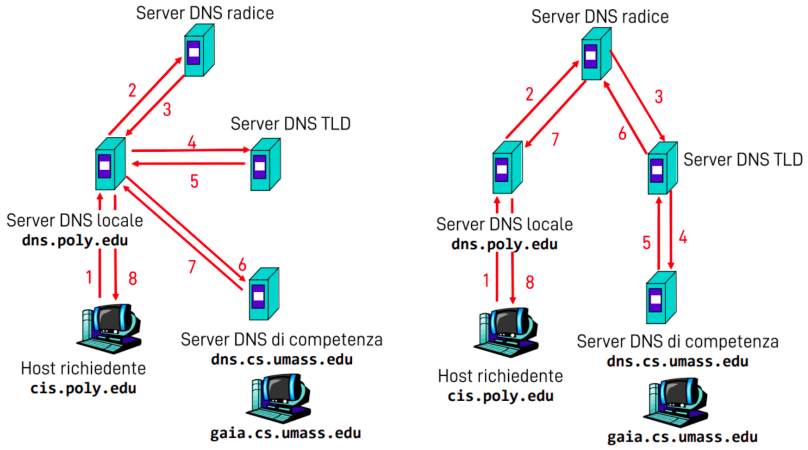
Pros of a distributed DNS: no single point of failure, traffic gets split among multiple servers, databases can be put closer to user, can be scaled.

Structure of DNS:
1. Root nameservers: 13 logical root nameservers in the world (although many physical copies of them), returns IP address of authoritative nameserver for the requested top-level domain (TLD) server (ex. .com, .edu, .it…)
2. TLD servers: have the addresses of all authoritative name servers of domain names with that TLD
3. Authoritative name servers: name server of the organization, provide DNS resolution for their domains

Each ISP has also its own "local" DNS name server, which provides DNS resolution for their customers. This local DNS server acts like a proxy which caches records and forwards queries into hierarchy if it does not have them cached.

2.3.3.1 DNS querying

When a host wants to resolve a domain name, it asks its local DNS server. If the local DNS server has the response in cache, it passes it directly to the host, otherwise it interrogates the DNS hierarchy. There are two ways of asking: iteratively and recursively. In the iterative approach the DNS servers reply with the address of the next server in the hierarchy, while in the recursive approach the server actually asks the next server for the answer and returns the final answer to the client. Since this method would put a very heavy load on the root nameserver, almost no server supports this method anymore.



(a) Query *iterativa* (b) Query *ricorsiva*

2.3.3.2 DNS caching

When a nameserver learns about the mapping of an IP address, it stores it in a cache. These entries get deleted after some time (TTL), because they might be outdated.

2.3.3.3 DNS resource records (RR)

DNS servers store resource records (RR) in this format:

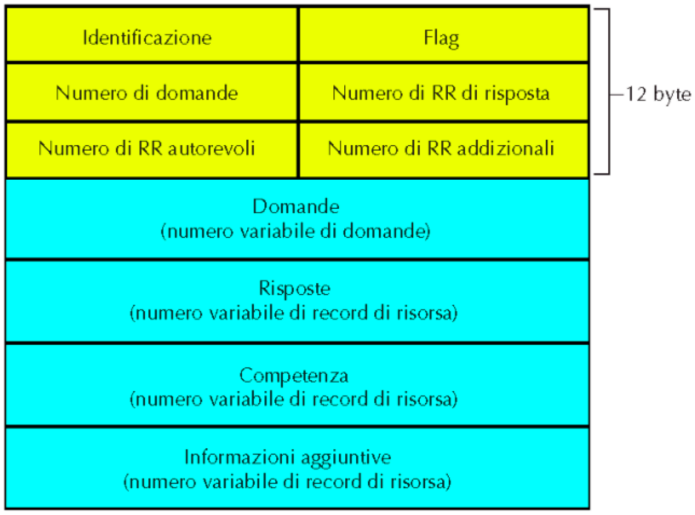
```
(name, value, type, ttl)
```

Common values for the type field are:

- A: associates an IPv4 address to a domain name (name: hostname, value: IPv4 address)
- NS: associates an authoritative nameserver to the whole domain (name: domain, value: name of the nameserver)
- CNAME: associates an alias to the canonical name (name: alias hostname, value: real hostname)
- MX: associates the mail server to the domain name (name: domain name, value: domain of the mail server)

2.3.3.4 DNS protocol messages

The DNS protocol specifies two types of messages: request and response. Both have the same format.



The fields are:
- Identification: 16 bit field which uniquely identifies the answer and the response
- Flag: query/reply, recursion desired, recursion available, reply is authoritative
- Questions: name and type fields for a query
- Answers: requested RRs
- Authority: records for authoritative nameservers
- Additional information which can be useful

2.3.3.5 Inserting new records into DNS

When we register a new domain name at a registrar, the regitrar inserts two RRs into the TLD nameserver: the domain name of the authoritative nameserver and the IP address of the authoritative name server.

Example:
- (networkutopia.com, dns1.networkutopia.com, NS)
- (dns1.networkutopia.com, 212.212.212.1, A)

2.4 P2P architectures

In a pure P2P architecture there is no central server, but the clients directly communicate with each other intermittently.

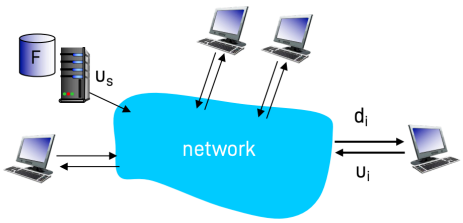
2.4.1 File distribution time

To compute the time required to share a file from a server to all peers in the network we must consider the following factors:
- upload capacity u_N and download capacity d_N of peer N
- upload capacity of server u_S

In the case of a tradition client-server configuration the time would be:

```
F = file size
Time to send one copy: F/u_S
Time to send N copies: NF/u_S
Minimum guaranteed client download rate: d_min
Minimum guaranteed client download time: F/d_min
Time to distribute F to N clients: >= max{NF/u_S, NF/d_min}
```

The time increases linearly in N!



If we use a distributed approach we get:

- $\frac{F}{u_S}$ - Time for the server to distribute one copy of the file among the peers
- $\frac{F}{d_{\min}}$ - Minimum time for one peer to download the whole file (possibly from multiple sources)
- Clients in total have to download NF bits -> download is limited by the maximum upload rate $u_S + \sum u_i$
- $\frac{\mathrm{NF}}{u_S + \sum u_i}$ - Total upload capability of the network

Finally the time to distribute F to N clients hat to be $\geq \max\left(\frac{F}{u_S}, \frac{F}{d_{\min}}\right), \frac{\mathrm{NF}}{u_S + \sum u_i}$. This is still dependent on N, but when we add new peers we also increase the total bandwidth of the network!

2.4.2 BitTorrent implementation

When a peer joins the BitTorrent network, it will announce its presence to the tracker server and start collecting chunks from other peers and uploading chucks it already has. The rarest chucks are requested first. The peer will upload chucks only for the 4 peers which are sending to it chunks at the highest rate. Every 30 seconds there is a reevaluation and the peer will optimistically change a peer with a different one.

2.5 Multimedia distribution

The major consumer on internet bandwidth is multimedia. Due to the high volume of traffic and the heterogeneity of the consuming devices, special solutions have to be implemented so that multimedia distribution works at scale.

2.5.1 DASH protocol

DASH (Dynamic Adaptive Streaming over HTTP) is a protocol which deals with optimising streaming of multimedia over the internet. The server divides the video file into multiple chunks and each chunk is stored in different encodings. Then it produces a manifest file which contains the URLs for the different chunks. The client periodically measures server-to-client bandwidth and, by using the manifest, requests chunks by choosing the most optimal encoding that the network can handle at that point.

2.5.2 Content Delivery Networks (CDN)

Due to the high volume of traffic that multimedia generates, it is advisable to distribute the same resources across multiple servers. With this approach we are avoiding having a single point of failure, a single point of network congestion and a possibly long distance between users and server. These servers are called CDNs and are strategically placed close to the end users to reduce latency.

3 Transport layer

The transport layer provides tools to enable logical communication between processes on different hosts. The transport protocols are run exclusively on the transmitting and receiving side.

3.1 Services offered by the transport layer

There are two main transport protocols: TCP and UDP. TCP provides the ordered delivery of segments, deals with congestion control and connection setup. UDP, on the other hand, just transmits segments with the lowest possible overhead, but without guarantees of delivery or ordering.

3.1.1 Multiplexing and demultiplexing

Multiplexing and demultiplexing is the operation that allows using multiple sockets on the same host. To achieve this, the sender adds information about the source port and the destination port in the header of the segment. The receiver reads this information and directs the datagram to the right socket.

The port is a 16-bit integer, which uniquely identifies a process on the machine.

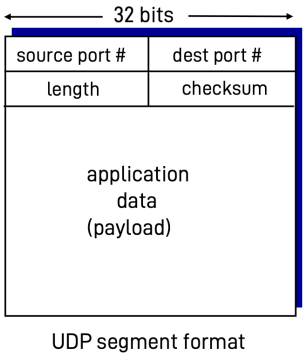
Servers usually offer services on well-known ports (static), which are often defined in standards and don't change. Conversely, clients often open connections on random ports (dynamic) assigned by the operating system. Usually server and client port numbers are never the same, otherwise if a client opens multiple processes to the same server port collision would happen (ex. client open multiple browser tabs).

A flow is a group of data belonging to the same logic communication. An application can open multiple connections and therefore manage multiple flows.

3.2 UDP protocol

The UDP (User Datagram Protocol) is a protocol offering a "best effort service", therefore it does not guarantee that the USP segments will reach the destination or that they will arrive in order. UDP is connectionless, that means that there is no handshaking between sender and receiver, so each segment is independent from the others. If we want to, we can add reliability to the protocol at the application layer.

Advantages of UDP: less delay because there is no connection establishment, no need to maintain state of communication, smaller header size, no limit on the amount of data sent because there is no congestion control.



UDP segment format

3.2.1 Error control

In the header of the UDP segment there is a checksum field. The sender splits the data (including header) in 16 bit integers, sums them and flips all the bits (ones' complement). The receiver just sums the data and adds that sum to the checksum. If the checksum is correct, the final result should be all

ones, otherwise there has been an error in transmission. Checksum field is optional in IPv4, required in IPv6.

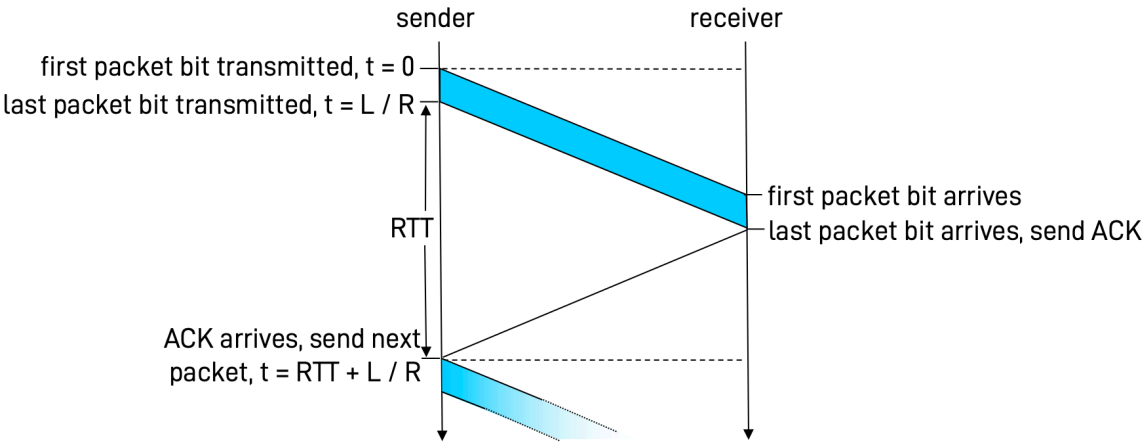
3.3 Reliable data transfer

There exists a class of protocols which deal with packet losses called ARQ (Automatic Repeat reQuest). These protocols use a special packet called ACK (Acknowledgement) to notify the transmitter of the reception.

3.3.1 Stop-and-wait protocol

For this protocol the transmitter sends a PDU and keeps a copy and waits to receive the ACK. If it doesn't receive the ACK before the timer runs out it resends the PDU. When an ACK is received it checks the sequence number and the checksum of the ACK and if correct sends the next PDU.

On the other side when the receiver gets a PDU it checks the sequence number and the checksum. If they are correct it sends back an ACK and forwards the SDU to the higher level, otherwise it drops the PDU.



3.3.1.1 Efficiency of the protocol

Assuming R = 1Gbps link, 15ms propagation delay (RTT 30ms), L = 8000 bit packet, then the transmission delay (time to send 8000 bits) is:

$$d_{\text{trans}} = \frac{L}{R} = \frac{8000\text{bit}}{10^9\ \text{bps}} = 8\mu s$$

This is the speed at which the router can push data onto the wire, but it's not the only factor involved in computing the delay. We must also consider the propagation delay (time the data needs to travel through the wire) and the processing delay at the other end to create the ACK packet. Therefore the final perceived throughput is:

$$\frac{L}{d_{\text{trans}} + \text{RTT}} = \frac{8000\text{bit}}{8\mu s + 30\text{ms}} = 33\text{kB/s}$$

(we need to add to the link speed the propagation delay to get the ACK back) The efficiency of the link is:

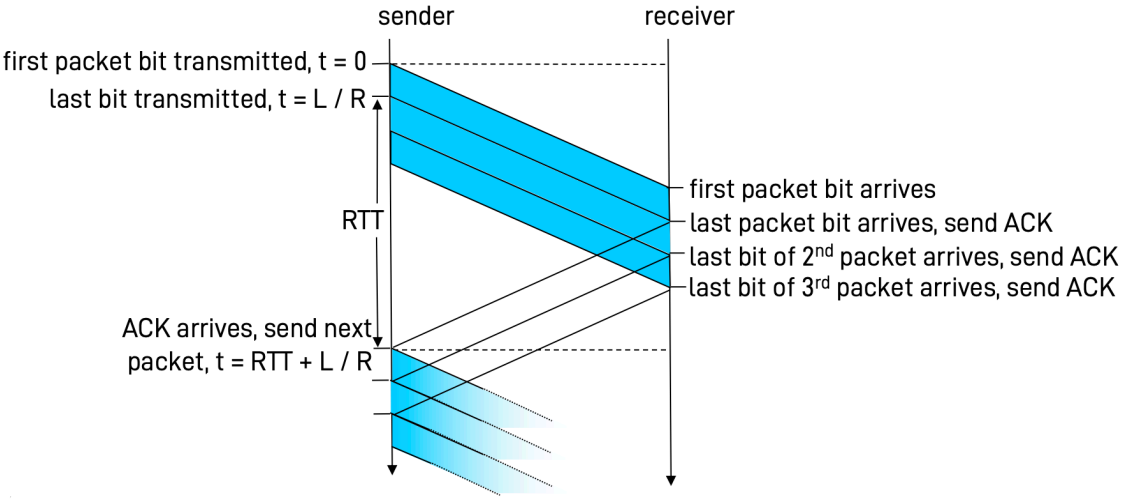
$$\frac{d_{\text{trans}}}{d_{\text{trans}} + \text{RTT}} = \frac{8\mu s}{30.008\text{ms}} = 0.00027 = 0.027\%$$

We can see from the computations that this method is extremely inefficient!

3.3.2 Pipelined protocols

In pipelined protocols the sender sends multiple packets at once and keeps track of their sequence number. Using pipelining we can increase the throughput of the Stop-And-Wait protocol. For example, if we send three packets at once, we triple the throughput.

$$\frac{3L}{d_{\text{trans}} + \text{RTT}} = \frac{24000\text{bit}}{8\mu s + 30\text{ms}} = 100\text{kB/s}$$



Note that the time start when the first packet is transmitted and ends when the first acknowledgment arrives.

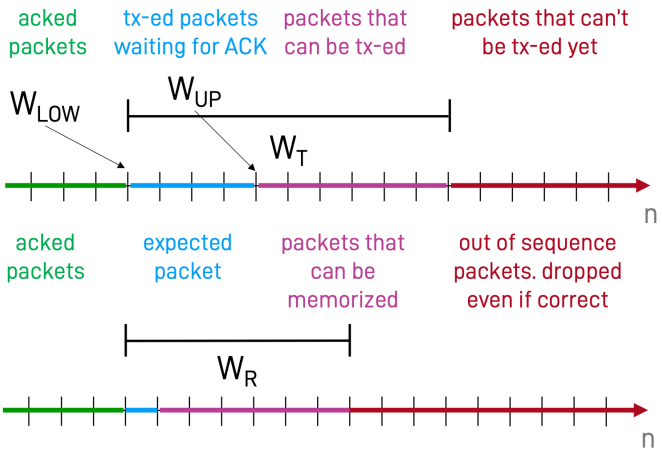
3.3.2.1 Terminology of pipelined protocols

Transmission window W_T: set of PDUs that the transmitter is allowed to transmit without receiving an acknowledgement. The maximum dimension of the window is limited by the allocated memory at the transmitter side and is denoted by $|W_T|$.

Receive window W_R: set of PDUs that the receiver is capable to accept and memorise. The maximum dimension of the window is limited by the allocated memory at the receiver side.

Low pointer W_{LOW}: pointer to the first packet in the transmission window W_T

Up pointer W_{UP}: pointer to the last packet that has already been transmitted



3.3.2.2 Acknowledgement packets

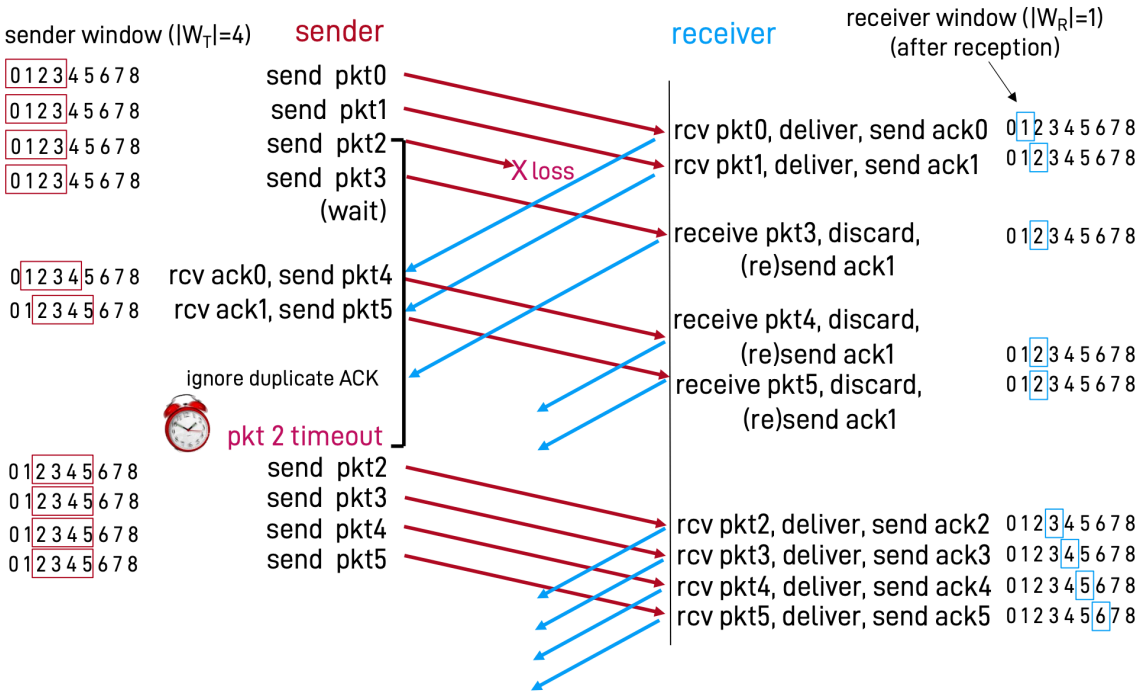
There actually exist multiple types of acknowledgment packets:

- Individual ACK: indicate the reception of a specific packet; $\text{ACK}(n)$ means that packet n has been received
- Cumulative ACK: indicates the correct reception up to a certain packet; $\text{ACK}(n)$ means that all packets up to n (non included) have been received
- Negative ACK: requests the transmission of a single packet; $\text{NACK}(n)$ means send packet n again

With the **piggybacking** technique we can send an acknowledgment inside a normal packet.

3.3.2.3 Go-back-N protocol

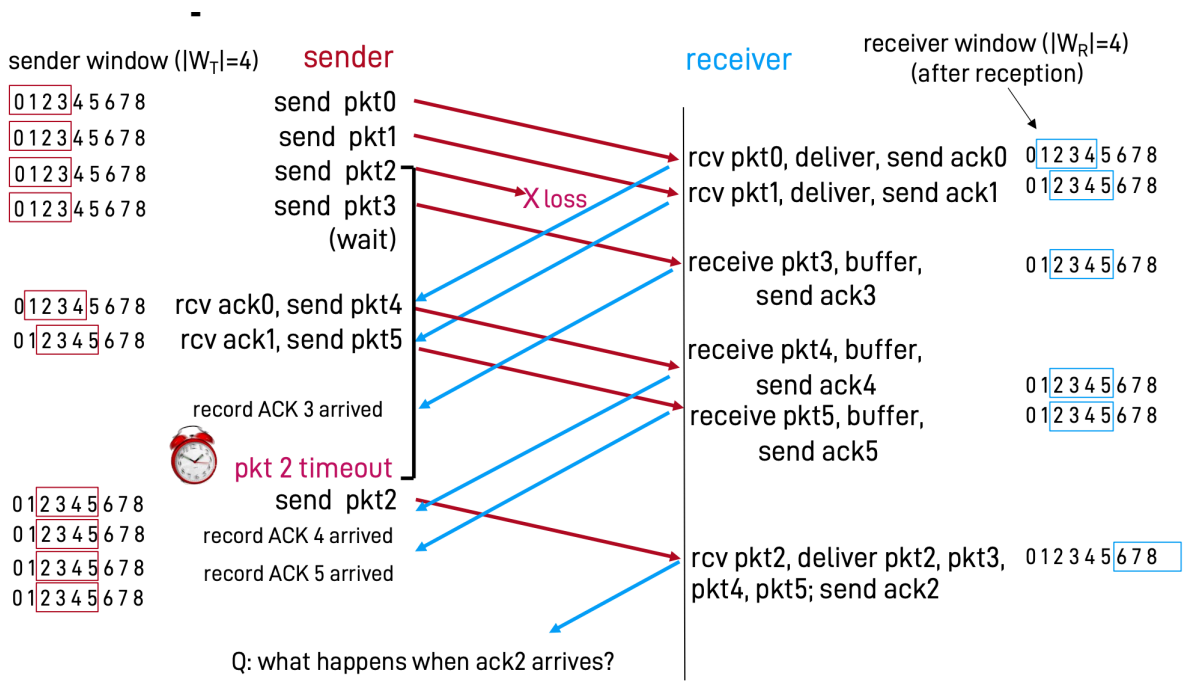
In this protocol the sender can have up to N unacked packets in the pipeline. The receiver uses cumulative ACK to signal that it has received the packets. If it detects a gap in the packets for every received packet it will retransmit the last ACK sent. That means that it doesn't accept any more packets until the missing packet is received. The sender keeps a timer starting from the oldest unacked packet. When this timer expires, it retransmits all packets from the expired one. This timer is reset every time an ACK is received. Because all packets get retransmitted, even those that may have been actually received after the missing one, there is an intrinsic inefficiency with this approach.



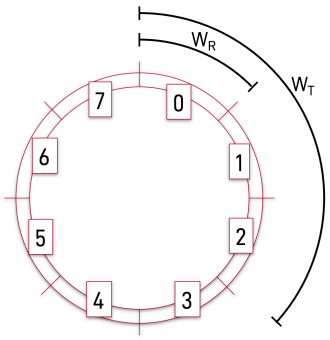
Note: if the sequence numbers are encoded using k bits, the maximum sender window size has to be $|W_T| \leq 2^k - 1$, because otherwise there could be multiple packets with the same sequence number in the same window.

3.3.2.4 Selective repeat

As in Go-Back-N, the sender can have up to N unacked packets in the pipeline, but this time the receiver will send individual ACKs. The sender keeps a timer for every unacked packet and will retransmit that packet when the timer expires. The receiver will keep the packet in a buffer until all packets before that one have been received.



In **selective repeat** the maximum window sizes has to be $|W_T| + |W_R| <= 2^k$, so that the receiving window size doesn't wrap around past the sending window size and risk having two packets with the same number in the same window.

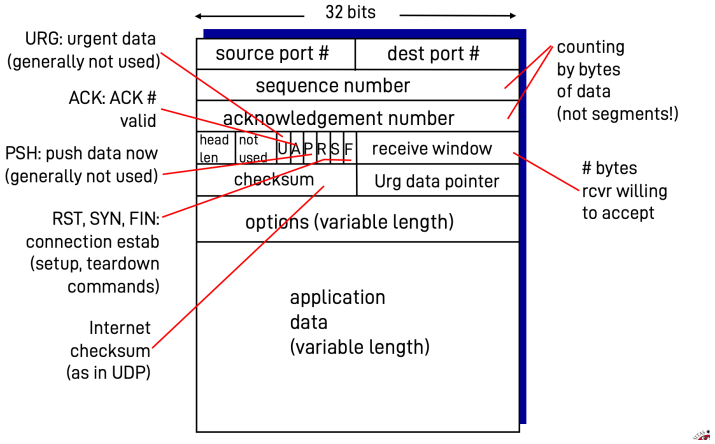


3.4 TCP protocol

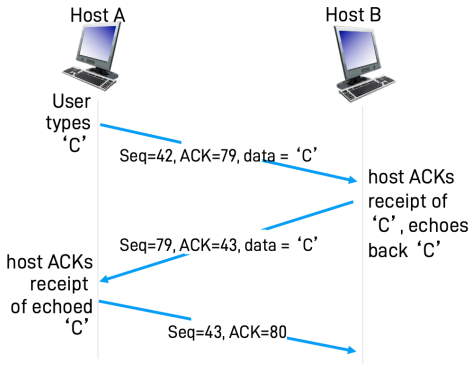
TCP is a connection-oriented (has handshaking), point-to-point protocol that allows reliable, in-order segment transmission. TCP uses pipelining, where the size of the sender and receiver windows are set by flow control (sender should not overwhelm receiver) and congestion control (network should not get clogged) algorithms. Data flow is bidirectional (full-duplex) and the protocol is connection-oriented, because there is handshaking.

3.4.1 TCP segment structure

The number in bytes that the receiver is capable of buffering (receive window) is specified in the RWND (receiver window) field. Assuming that the sender can transmit very quickly, it will have to stop when it sends as many bytes as specified by RWND. Since it is a 16-bit field the maximum data is $2^{16} = 64$kB, but we can specify a scaling factor so we can have larger windows.



The field **Sequence number** indicates the number of the first byte of that segment in that flow of data, while **Acknowledge number** indicates the sequence number of the next byte that we are expecting to receive from the other end, noting that TCP uses cumulative acknowledgments.

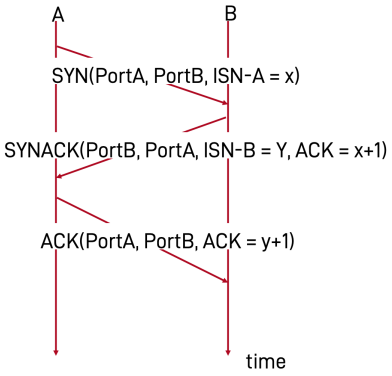


simple telnet scenario

3.4.2 TCP connection setup

TCP connection is setup using a mechanism called 3-way handshake:

1. Host A initialises the connection sending a segment with the SYN flag set, source port set to A, destination port set to B and initial sequential number x
2. Host B replies with a segment with SYN and ACK flags set, source port B and destination port B, sequential number y and ACK x+1
3. Host A sends an ACK segment with source port A, destination port B and ACK y+1



3.4.3 TCP connection teardown

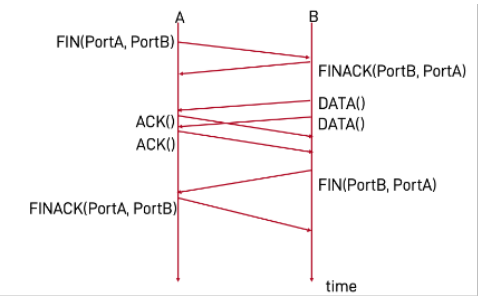
Since TCP communication is bidirectional, teardown of the connection is required in both directions.

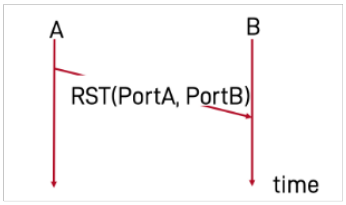
3.4.3.1 Gentle teardown

The host that wants to terminate the connection sends a segment with the FIN flag set and the receiver replies with ACK. Now the connection is half closed, because the first host cannot send anything anymore, but communication in the other direction is still allowed. To terminate the connection in the other direction also the other host must send a segment with the FIN flag set.

3.4.3.2 Rude teardown

This method is used for connections that are in an error state (ex. an ACK gets received on a non-existing connection). To terminate the connection the host sends a segment with the RST flag set. Then both hosts can free the resources needed by the connection.





3.4.4 Maximum segment size (MSS)

Although TCP works with bytes, it doesn't send single bytes, but rather tries to send segments as big as MSS. MSS depends a parameter from the lower level called MTU (Maximum Transfer Unit), which in turn relies on the MTU of the Data link layer. To determine the size TCP proceeds by trial and error, until a segment gets lost. By default the MSS is set to 1460 bytes (ethernet has a 1500 bytes MTU - 40 bytes for the headers).

3.4.5 Estimating the round trip time (RTT) and choosing timeouts

Choosing the retransmission timeout (RTO) is complicated because if it is too short, then unnecessary retransmissions might happen, while if it is too long TCP will have a slow reaction time to losses. For sure the timeout time before a packet has to be retransmitted again must be grater that the round trip time. Also RTT estimation must be done carefully. In principle RTT is the time from when the packet is sent to when the ACK is received (ignoring retransmissions). Since this value is changing constantly, an average or among several transmissions might give us a better estimate. To compute it we use this formula:

$$\mathrm{SRTT} = (1 - \alpha) \cdot \mathrm{SRTT} + \alpha \cdot \mathrm{RTT}$$

Where SRTT is the "smoothed" round trip time. This is an exponential weighted moving average, where influence of past samples decreases exponentially fast. A value of $\alpha = \frac{1}{8}$ is typically used. The retransmission timeout is then given by SRTT plus a safety margin. To decide the safety margin we can compute the deviation of the actual RTT from the SRTT in the following way:

$$\mathrm{RTTVAR} = (1 - \beta) \cdot \mathrm{RTTVAR} + \beta \cdot |\mathrm{SRTT} - \mathrm{RTT}|$$

Where RTTVAR is an exponentially weighted moving average computed over the difference between SRTT and RTT. Usually $\beta = \frac{1}{4}$. Then: $\mathrm{RTO} = \mathrm{SRTT} + 4 \cdot \mathrm{RTTVAR}$ The standard defines that these values are initialised to: SRTT=RTT of first packet, RTTVAR = RTT/2, RTO = 1s.

3.4.6 TCP flow control

Flow control is a functionality of the TCP protocol that allows the receiver to control the seed of transmission of the sender. To do this, the receiver indicates in the RWND field of every segment the size

of its receive window. Then, the sender sets its sender window to RWND. This guarantees that the receiver will not overflow its buffer.

3.4.7 TCP congestion control

Congestion happens when too many receiver are sending too much data and the network is not able to handle that traffic. In a congested network packets get dropped because the buffers in routers overflow and there are high delays because there are long queues in the router buffers.

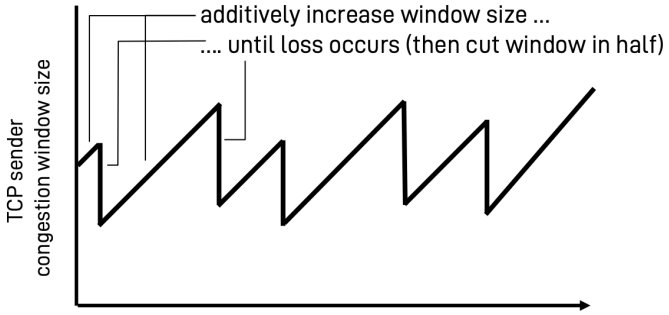
Congestion control is a very complicated problem and many algorithms have been proposed, such as TCP Tahoe, Reno, New Reno, Vegas... Because of this, there does not exist a single TCP implementation, but many of them.

There are two possible approaches to this problem:
- End-to-end: congestion is estimated by looking at packet losses and delays
- Network-assisted: routers provide feedback on congestion of the network

3.4.7.1 AIMD (Additive Increase Multiplicative Decrease)

In this algorithm the sender progressively increases the transmission rate by increasing the window size until a loss occurs. When a loss occurs the window gets shrunk.

The protocol has two phases:
- Additive increase: the window size gets increased by 1 MSS every RTT until a loss is detected (NOTE: it increases the window size every RTT, not when an ACK is received!)
- Multiplicative decrease: after a loss, the window size gets halved



The AIMD protocol ensures fairness in the network. That means that if K connections share the same link with bottleneck R, each connection has a bandwidth of R/K.

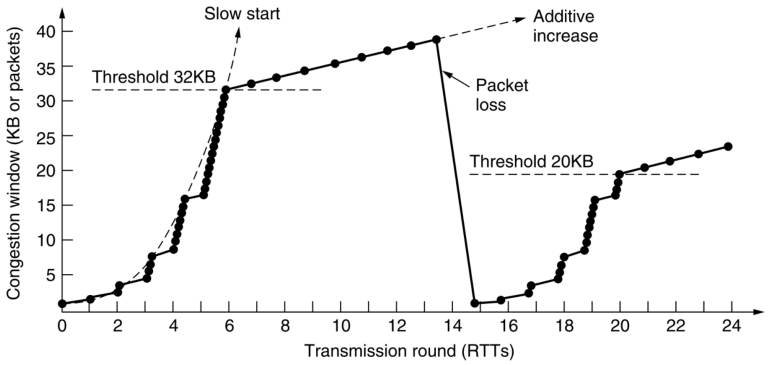
3.4.7.1.1 Implementation in TCP

Congestion window (CWND): number of bytes that the transmitter is allowed to inject into the network. The sender window is therefore given by:

$$|W_T| = \min(\text{CWND}, \text{RWND}) = \min(\text{CWND}, |W_R|)$$

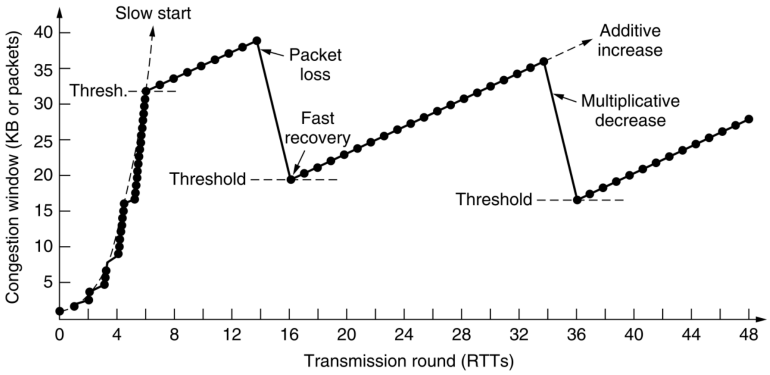
TCP adapt the size of CWND by implementing the AIMD algorithm using two phases to reach quickly the optimal point:
- Slow start: for every ACK received by the transmitter, the window size gets increased by 1. This induces an exponential growth (fucking compound interest goes brrrr). When CWND reaches SSTHRESH (slow start threshold), TCP switches to congestion avoidance
- Congestion avoidance: for every ACK received by transmitter, CWND increased by $\text{MSS} \cdot \frac{\text{MSS}}{\text{CWND}}$. That means that if all packets in a window get ACKed, then the window size increases by one MSS. This induces a linear increase in time.

At the beginning of the transmission CWND gets initialised to 1 and SSTHRESH to RWND. When a timeout happens the algorithm always switches to slow start, retransmits the unACKed packet, SSTHRESH gets set to $\max\left(\frac{\text{CWND}}{2}, 2\right)$, RTO gets doubled and CWND gets set to 1.



This method is very inefficient and does not consider the number of duplicated acknowledgements we get, which provide information about the working of the network (remember that TCP uses cumulative acknowledgements).

In the fast retransmit and fast recovery algorithm after three duplicated acknowledgements are received the sender retransmits the segment indicated by the duplicated acknowledgements (fast retransmit) and enters fast recovery mode. SSTHRESH gets set to $\frac{\text{CWND}}{2}$, W_{LOW} still points to the first unACKed segment and CWND gets set to SSTHRESH + 3 MSS. After the sender receives another duplicated acknowledgement CWND gets increased by 1 MSS. After a valid ACK has been received (this means that the lost segment has been successfully received) CWND gets set to SSTHRESH and the sender exits fast recovery and enters congestion avoidance.



3.4.8 TCP fairness

The TCP protocol does not solve all fairness problems:

- TCP connections may limit their bandwidth if other UDP connections, which do not have congestion control, are using the link at the same time
- Applications can open multiple TCP connections in parallel

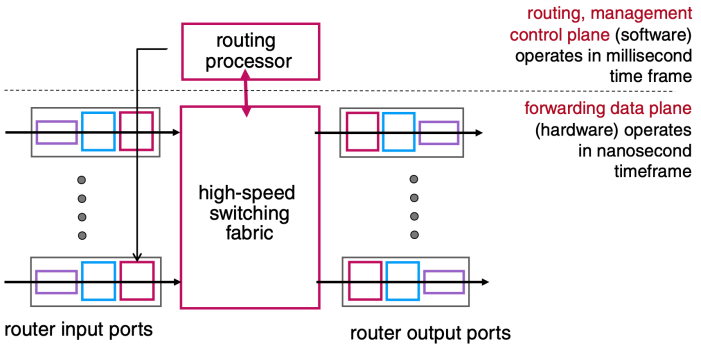
4 Network layer

The main network layers functions are:

- Forwarding the packet from the router input to the router output (data plane)
- Routing packets from source to destination using routing algorithms (control plane)

4.1 Router architecture

Routers implement the network stack up to the network layer. They are logically divided in the control plane and the data plane. The control plane deals with deciding where a packet should go and is carried out by a routing processor. The control plane deals with the physically forward packets from the input port to the output port.



4.1.1 Switching fabric

The switching fabric deals with transferring the packet from the input to the output buffer. The performance is measured as multiples of input/output line rate. There exist three types of switching fabrics:

- Memory based: works as in normal computers, packets get saved to memory. The speed is limited by the memory bandwidth and memory has to be accesses twice to transfer one packet
- Bus based: all input and output ports share the same bus, speed limited by bus bandwidth
- Interconnection network: uses a matrix of connections to enable extremely high speeds

4.1.2 Input port functions

The input port of a router is composed of:

- Line termination: receives the bits from the wire
- Link layer: protocol from the link layer to interpret the received data from medium (ex. Ethernet, fiber)
- Queue

4.1.3 Consequences of switching delays

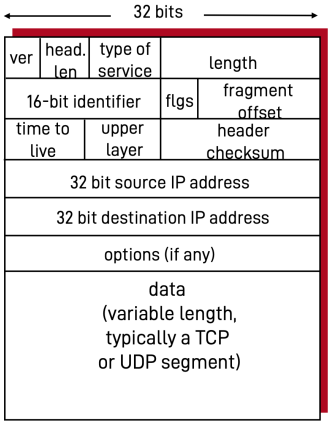
If the switching fabric is slower than the input ports combined then we will have queueing. Head-of-line queueing happens when the first datagram in the queue blocks all other datagrams behind because his output port is already being used. We can have queueing also at the output ports. To deal with this we can implement some scheduling algorithms that prioritise some packets rather than others (problem for net neutrality!).

4.2 IP protocol

The IP packet has a header organised in 32 bit blocks. Its most important fields are:

- Version: IPv4 or IPv6
- Header length: number of 32-bit blocks
- Total length: number of bytes including header and data
- Identification+fragment flag+fragment offset: used to reconstruct whole packet if it was split

- Time To Live (TTL): 8-bit field initialised by the sender which will get decremented by 1 every time it goes through a router. When it gets to 0 the packet gets dropped
- Protocol: transport protocol used in the data
- Checksum: 16-bit one's complement of the header
- Source and destination ip address: 32 bit addresses
- Additional options + padding



4.2.1 IP fragmentation

Every hardware technology specifies a maximum dimension of the frame (layer 2 PDU) that is transiting on it known as Maximum Transmission Unit (MTU). Since different networks can have different MTUs, it may happen that a datagram has to be split. In this case some header fields are modified:

- the fragment offset field tells at which position in the original datagram the fragment belongs
- The Flags field [0, D, M] contains the following information:
  - D - The packet must mot be fragmented
  - M - This packet is a fragment and more fragments are expected
- the identifier field uniquely identifies fragments of the same datagram

Fragments get reassembled only at the destination, so the routers don't have to deal with waiting for all fragments to arrive and fragments don't need to take the same route. The receiver waits for all fragments to arrive by buffering them. If the maximum waiting time is exceeded, fragments that have arrived are dropped and the sender has to send them again.

Fragments is nowadays disabled on the public internet, therefore datagrams that are too large get automatically discarded by routes. This is due to security risks:

- firewalls may be deceived by checking a fragment, allowing it through and then not checking new fragments that are sent if the initial ones get lost
- Memory exhaustion attacks by intentionally omitting fragments
- Buffer overflows due to a bad implementation of the reassembling procedure

4.2.2 IPv4 addressing

Interface: connection between host/router and physical link

IP addresses are 32 bit numbers assigned to host and router interfaces. IP addresses are represented in dotted decimal notation, where each 8-bit segment is represented in decimal value, with periods as separators. IP addresses are divided in two parts:

- Network ID, which identifies the physical network to which the host is attached
- Host ID, which identifies the network interface

For the network to work there must be the guarantee that each host has a unique IP address and that network ID are coordinated globally (host IDs are coordinated locally instead).

4.2.2.1 Addressing classes

Initially to distinguish between the network part and the host part of an IP address static classes of IP addresses were defined. Since everyone wanted the bigger classes, they run out of them quickly, so a new method of assigning Ip addresses had to be found.

4.2.2.2 Classless addressing

A central authority called ICANN (Internet Corporation for Assigned Names and Numbers) to handle address assignment has been established. The authority gives to registrars the authority to assign IP blocks to ISPs, which in turn sell them to organisations and customers.

In classless addressing the number of bits that denote the network is customisable. This is done by specifying the network mask. The network mask is a 32-bit number, where the bits are set to 1 if there are part of the network ID. Therefore, to identify the Network ID of an IP address it is sufficient to do a bitwise AND between the IP address and the network mask. The network mask can be also represented in CIDR (Classless Inter-Domain Routing) notation, which is a decimal number denoting the number of bits denoting the Network ID.

4.2.2.2.1 Forwarding table

Routers internally use a forwarding table, which indicates on which interface the incoming packet should be routed. To choose the right interface, they perform a bitwise and between the destination IP address and the network mask of every entry in the table. If the result matches an entry in the table, the packet is forwarded on that interface. If multiple entries match, the entry with the longest Network ID is chosen. The last entry is the table is usually 0.0.0.0/0, which matches all IP addresses, and packets get forwarded to this interface if no packet matches.

If there are multiple routers in the same network, the router must save in the forwarding table also the IP address of the router which should handle the packet we are forwarding. This field is called **next hop**.

4.2.3 Private IP addresses and NAT

Not all IP addresses are public: some are reserved for private networks. These IP addresses can be used only in local networks and are not reachable by devices outside that specific network. There are three blocks of IP addresses assigned to private networks:

- 10.0.0.0/8: from 10.0.0.0 to 10.255.255.255
- 172.16.0.0/12: from 172.16.0.0 to 172.31.255.255
- 192.168.0.0/16: from 192.1168.0.0 to 192.168.255.255

Since they are not accessible from external networks, all local networks can reuse the same IP address blocks.

Since it is not possible to communicate from the network to devices with private IP addresses, two solutions have been developed to address this problem:

- NAT (network address translator): device which swaps the source IP address and port of the packets it receives with the public IP address and a random port
- Proxy: a computer with a public and a private IP address which receives the requests from private hosts and executes them on their behalf on the public network. This is done at the application layer

The NAT has an internal translation table, which keeps track of the internal private IP address and port which has sent the packet and the corresponding public destination IP address and port, so when the NAT receives a response back from the public side, knows to which host to forward the packet to. The advantages of using NAT are that we can have $2^{\{16\}} \approx 60000$ connections with a single IP address and that local hosts are unaccessible from the external network, except when they are the ones establishing

the connection. Although is has the benefit of addressing the IP shortage, the implementation of NAT is controversial because routers modify packets at level 3 and 4, therefore applications at higher levels have to manage also this cases (ex. P2P applications). Moreover, special techniques have to be adopted to make a local client accessible from the outside world (ex. port forwarding or hole punching).

4.2.4 Special IP addresses

4.2.4.1 Network addresses

They are addresses that refer to the prefix of the network and have all host bits set to 0.

$$\underbrace{10000000.11010011.00000000.0001}_{\text{NetID}}\underbrace{0000}_{\text{HostID}}$$

4.2.4.2 Direct broadcast addresses

When a packet reaches the network specified in the NetID, the packet gets sent to all the hosts in the network. This address has all host bits set to 1.

$$\underbrace{10000000.11010011.00000000.0001}_{\text{NetID}}\underbrace{1111}_{\text{HostID}}$$

4.2.4.3 Limited broadcast addresses

Packets with this broadcast address never get forwarded by routers, so they are directed only to hosts in the same network.

$$255.255.255.255 \Rightarrow 11111111.11111111.11111111.11111111$$

4.2.4.4 This computer address

When a host does not yet know its IP address, it can use this special IP address to identify itself.

$$0.0.0.0 \Rightarrow 00000000.00000000.00000000.00000000$$

4.2.4.5 Loopback address

Loopback addresses are all addresses in the 127.0.0.0/8 subnet and are used to test applications running in the same host. When an application sends a packet with this address, the protocol stack travels down to the network layer and then directly travels up to forward the packet to the right application without ever leaving the host.

4.2.4.6 Multicast

There are various ways of achieving a multicast communication (sending the same packet to a subset of host).

4.2.4.6.1 Multicast addresses

Multicast addresses are used to send a packet to all hosts. Then hosts can decide to keep them or drop them if not interested. Packets should be duplicated and sent to all hosts by all routers, but their usage is normally blocked on the internet. These special addresses start with 1110 and therefore are those between 224.0.0.0 and 239.255.255.255.

4.2.4.6.2 Multiple unicast

The source send one copy of the packet for each host interested. This method is suboptimal because it may require sending the same exact packet in the same link multiple times.

4.2.4.6.3 Multicast routing

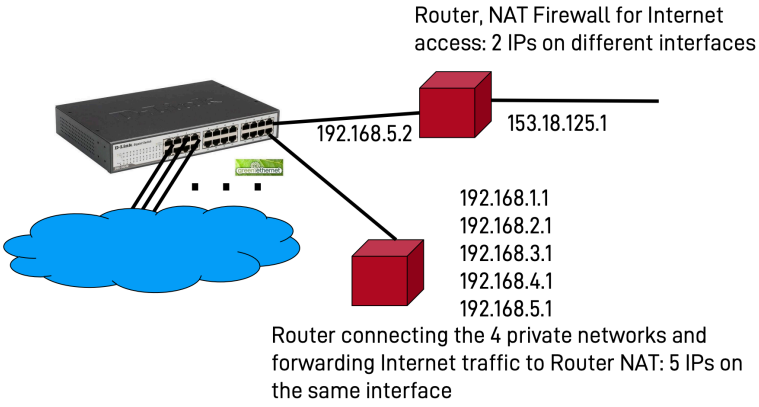
Special protocol have been developed to achieve multicast while minimising network overhead.

4.2.4.7 Link-local addresses

These exists a subnet reserved for hosts which cannot find an IP address. Hosts choose a random IP address from the 169.254.0.0/16 and use that IP address to communicate with each other.

4.2.4.8 IP addresses for routers

Each router can have two or more IP addresses, one for each logical network to which is connected. For example, a router can act as a bridge for multiple subnets, but all these subnets are connected to the router on the same network interface.

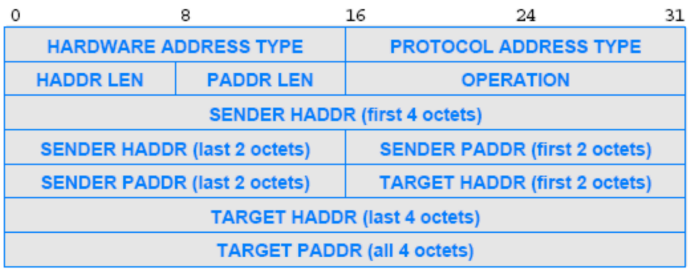


4.2.5 ARP protocol

Every IP packet gets encapsulated inside a level 2 frame and to send a level 2 frame the source and destination address of that layer are needed. These addresses are called MAC addresses and are 48-bit long numbers, which uniquely identify every network interface. While source and destination IP addresses remain unchanged during the whole trip of the packet, MAC addresses change at every hop. To go from level 3 to level 2, it is therefore needed to translate the IP address of the next hop to a MAC address. A host can discover the MAC address of another host in the network by using a level 2 protocol called ARP (Address Resolution Protocol). To do this, the host sends a broadcast request asking for the MAC address of host with IP address X. When host X receives the ARP request, it replies with its MAC address. When it travels through a physical network an ARP message is encapsulated in a hardware frame.

4.2.5.1 ARP message

The Hardware type field specifies the network link protocol type, therefore the type of address in PADDR. The message structure is the same for both request and response, so request field tells if this is a request or response (1 request, 2 response). To send an ARP request use the command arping. ARP responses are cached in an ARP table and updated periodically (every 30s).



4.2.6 ICMP

IP includes an auxiliary protocol called ICMP (Internet Control Message Protocol) which is used to notify errors and other information about the network to the sender of the packet.

ICMP generally has two message types:
- messages to report errors (ex. destination unreachable or time exceeded)
- messages to obtain information (ex. echo and reply)

During transport the ICMP message is encapsulated inside an IP packet. If an ICMP message generates an error, no error messages are sent (this is to avoid infinite loops of error messages).

4.2.6.1 Examples of ICMP messages

- Ping command: echo request and echo reply
- Traceroute: send IP packet with increasing TTL, routers on route will one by one reply with time exceeded

Number	Type	Purpose
0	Echo Reply	Used by the ping program
3	Dest. Unreachable	Datagram could not be delivered
5	Redirect	Host must change a route
8	Echo	Used by the ping program
11	Time Exceeded	TTL expired or fragments timed out
12	Parameter Problem	IP header is incorrect
30	Traceroute	Used by the traceroute program

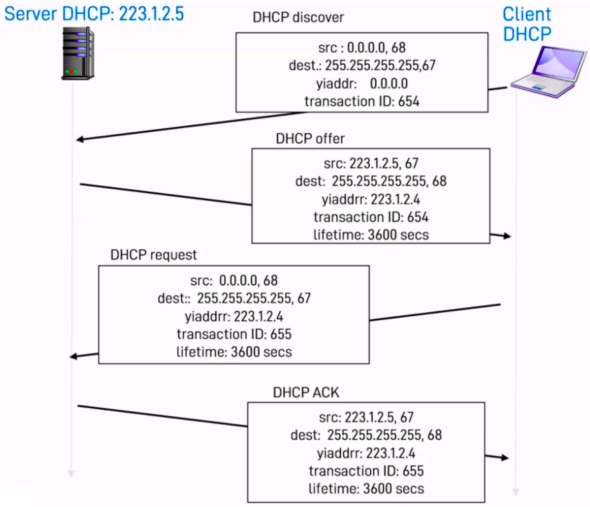
4.2.7 DHCP

The Dynamic Host Configuration Protocol deals with automatically assigning IP addresses to hosts when they join a network. DHCP has 4 main stages:
- Hosts broadcasts a discover message
- DHCP server replies with a DHCP offer
- The hosts accepts the offer
- DHCP acknowledges the assignment

Since the client doesn't have an IP address, all communications happen in broadcast. The IP address used (255.255.255.255) is limited broadcast, so it does not get forwarded by routers.

Routers can be configured to forward DHCP messages or else a DHCP router for every network is required.

DHCP is an application layer protocol (client and server have an assigned socket). The protocol uses UDP because there are no IP addresses on which to set up a TCP connection.

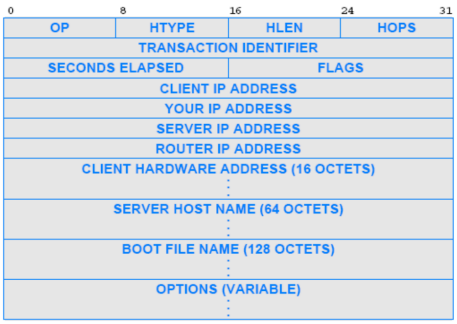


4.2.7.1 Lease management

The IP address given by DHCP is valid only for a period of time, after which the lease expires and it becomes available again for other connections. When a lease is about to expire, the hosts can ask for an extension of the lease. Normally, this lease is granted so that the client doesn't have to reopen all the connection that it has set up. If the extension is not granted the client must stop using the address.

4.2.7.2 DHCP packet

A transaction is made of a request or a response (first field). To distinguish among multiple transactions a Transaction ID field is used. The proposed IP address is put in the Your IP address field. DHCP can provide additional information to the client, such as address of first-hop router, name and IP address of DNS server and network mask.



4.2.7.3 Networks without DHCP

If there is no DHCP service offered on a network, clients can manually configure their IP or can pick a random link local address (169.254.0.0/16). In the latter case they use ARP to check if anyone has already taken that IP.

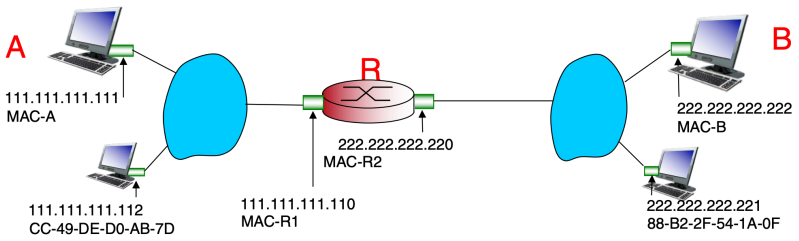
4.3 Journey of a packet

Suppose that client A wants to send a packet to host B and that A knows the IP address of B and the IP address and MAC address of the router (possible thanks to ARP and DHCP).

1. Host A creates an IP packet, setting as source their own IP and as destination the IP address of B. The it encapsulates it in a **frame** with its own MAC address and the MAC address of the router.
2. The router receives the address, does a lookup on the routing table and chooses the entry with the 222.222.222.220 interface, which has next hop as direct. Therefore the router will do an ARP request asking for B's MAC address. When it receives the response, the router will save it in its cache and

set is as the destination MAC address of the packet it has received from A. The router will also set its own MAC address as source and send the packet to B.

3. B will receive the packet and extract the contents



4.4 IPv6

The IPv6 protocol is an evolution of the IPv4 protocol and it was created to increase the quantity of available addresses, fixing the header length to increase processing speed and to improve the quality of service.

Since not all routers can be upgraded simultaneously the to protocol versions have to coexist. One solution found is tunnelling, that is encapsulating IPv6 packets inside IPv4 packets.

4.4.1 Structure of IPv6 addresses

IPv6 addresses are 128 bits long and are represented by 32 hexadecimal numbers, grouped in 8 groups by 4 ciphers each. CIDR masks works exactly the same way as in IPv4.

5 Routing protocols

5.1 General principles

This chapter will deal with routing algorithms, i.e. algorithms that determine "good" paths in the network of routers.

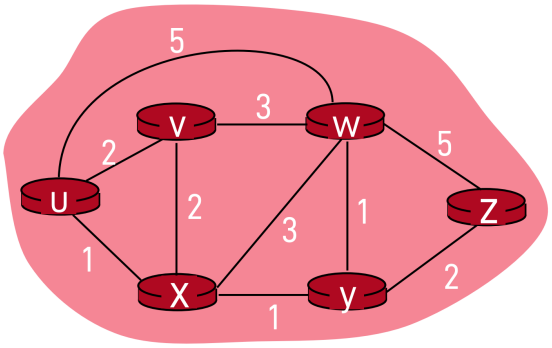
A path is a sequence of routers that a packet must travel to reach a destination. A "good" path may be the least expensive, least congested, fastest…

5.1.1 Abstraction of a network

A network can be seen as a non-oriented graph where nodes are routers and links are connections between them. A graph G is denoted by $G = (N, E)$, where N is the set of routers and E is the set of links.

$$N = \{U, V, X, W, Y, Z\}$$

$$E = \{(u, v), (u, x), (v, w), (v, x), (x, w), (x, y), (y, z), (w, z), (u, w), (w, y)\}$$



Each link has an associated cost, which is given by the cost function $c : N \times N \to \mathbb{N}$. For example for the link (w, z) the associated cost is $c(w, z) = 5$.

The cost can be determined on many parameters, such as number of hops, inverse of bandwidth (higher bandwidth -> lower cost), congestion…

5.1.2 Types of routing algorithms

We can categorise routing algorithms on 2 characteristics:

- Global or decentralised information: for global algorithms all routers know everything about the network. In the other case routers know only their neighbours and have to talk to them to gain more information. The first algorithms are called **link state** algorithms, while the second are **distance vector** algorithms
- Static or dynamic tables: if routing tables are static they are defined once by the network administrator and don't change with time, thus routing algorithms are not needed. If tables are dynamic they are periodically updated to respond to changes in the costs of the network.

5.2 Dijkstra's algorithm

Dijkstra's algorithm is a link-state algorithm. This means that all routers know all the link costs of the network. They achieve that by exchanging information with each other. The algorithm is run for every destination, so that after **k** iterations we know the cost **k** destinations. The notation used is the following:

- $c(x, y)$: link cost from node x to y (∞ if no direct neighbours)
- $D(v)$: cost of path from source to destination v

- $p(v)$: predecessor of router v in the path
- N: set of all routers
- N': set of already visited routers (routers for which the minimum cost has already been computed)

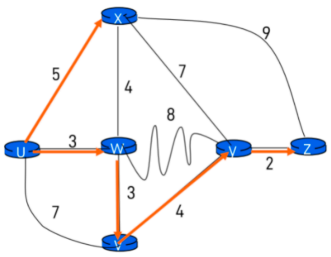
```
= Start from node u
N` = {u}
= Initialise cost table
foreach (v in U) do
  = Find adjacent nodes
  if (v in u.adj()) then
    = Add node to cost table
    D(v) = c(u, v)
  else
    D(v) = inf

do
  = From the set of unvisited routers (N - N`) take the one with the minimum associated
cost
  node w = min(N - N`)
  N`.insert(w)
  = For each adjacent node that is not in the set of already visited routers
  foreach (v in w.adj() - N`)
    = If the cost through node w is smaller than the cost in the table save this new
path
    if (D(w) + c(w, v) < D(v)) then
      D(v) = D(w) + c(w, v)
      p(v) = w
while (N`.size() < N.size()) = Repeat untill we have visited all nodes
```

Steps to compute Dijkstra's algorithm.

Passo	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

(a) Passi di costruzione della tabella di routing con *Dijkstra*



(b) Grafo di una rete

5.2.1 Complexity and problems

In a network of n nodes in the worst case all nodes except the current get checked at every iteration, therefore the time complexity is $\frac{n(n-1)}{2} = O(n^2)$. Some implementations reduce the complexity to $O(\text{nlog}(n))$.

Another problem with Dijkstra's algorithm is that if a path becomes the least expensive then all routers will choose that path, therefore making it very congested, while other paths become completely free. Then the algorithm will pick another path as the best and make that very congested, therefore inducing a lot of oscillations.

5.3 OSPF protocol

5.3.1 Autonomous system

Internet is organised in many subnetworks which are owned by some entity (ISP, corporations, network providers). Ideally, each one of these networks is autonomous and can be configured indepen-

dently of every other. These independent systems are called AS (Autonomous system) and are uniquely identified by a number. This division breaks down the problem of generating route tables in two: intra-AS routing and inter-AS routing.

OSPF is a link-state routing protocol for intra-AS routing (routing inside an autonomous system). It uses Dijkstra's algorithm to find the shortest path. Routers talk directly using IP packets (thus not using the transport layer) and using the multicast address 224.0.0.5.

5.3.2 Working of the protocol

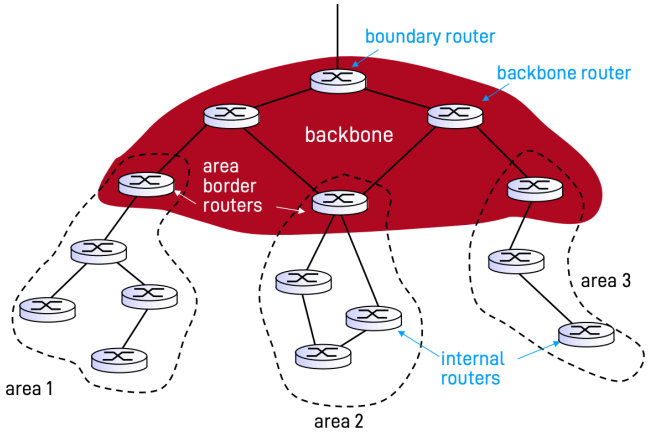
The OSPF protocol implements three procedures:

- Hello protocol: defines the exchange of messages to check which links are still active wherefore finding working neighbours
- Exchange protocol: used to exchange the known topology of the network to new neighbours
- Flooding protocol: used to inform everyone of a link change

The flooding protocol is designed so that all routers forward packets they receive on one interface to all the others. This is called **controlled flooding** end ensures that one message per link or one message per broadcast domain is sent.

5.3.3 Hierarchical OSPF

In large networks, the network gets divided in a hierarchy with two levels: a backbone network and many area networks. Messages are transmitted only inside the area network and routers inside an area network only know the topology of that area and the shortest path to all other areas. Each area network has a border router which talks to other routers and advertises its own paths. Also the backbone runs OSPF routing limited to the backbone.



5.4 Distance vector routing

In this algorithm the routers only know their neighbours and the costs to reach them. Routers learn new destinations via advertisements (distance vectors). The following notation is used:

- N_x: set of neighbours of router X
- R_x: routing table of router X
  - $R_{x[d]}$: row relative to destination d
    - $R_{x[d]}$.cost: cost to reach destination d
    - $R_{x[d]}$.nexthop: gateway to reach destination d
    - $R_{x[d]}$.time: time at which route was set (used to invalidate routes)
- D_x: distance vector of router x (vector containing all destinations reachable from x and their costs) $\left[\left(d, R_{x[d]}.\text{cost}\right) : d \in R_x\right]$

```
= Initialises routing table of router x
foreach (n in N_x) do
  R_x[n].cost = c(x, n)
  R_x[n].nexthop = n
  R_x[n].time = now()
= Every T seconds send out a distance vector to all neighbours
D_x = new distance vector
foreach (d in R_x) do
  D_x[d] = (d, D_x[d].cost)
foreach (n in N_x) do
  send(D_x, n)
= Update table when we receive a new distance vector from router y
foreach ((d, cost) in D_y) do
  if (d !in R_x or cost + c(x, y) < R_x[d] or y == R_x[d].nexthop) then
    R_x.cost = cost + c(x, y)
    R_x.nexthop = y
    R_x.time = now()
```

5.4.1 Example

At the beginning all routers know only their neighbours and the costs to reach them.

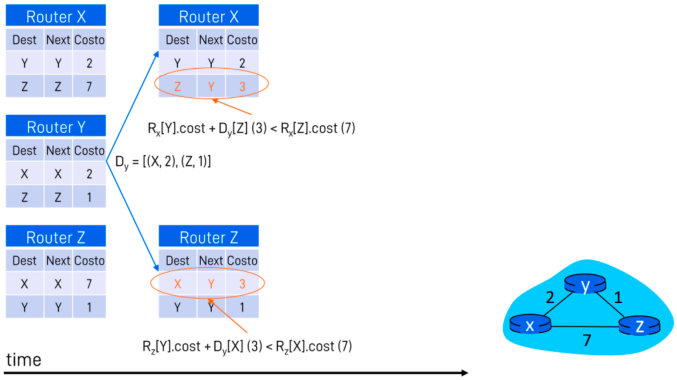


Fig. 5.5: Y trasmette il proprio *distance vector*

 After Y has sent its distance vector, routers X and Z update their routing tables. X discovers that by going through Y it can reach Z with just a cost of 3. Same happens for Z with X. X and Z transmit their vector, but nothing has to be changed in the tables.

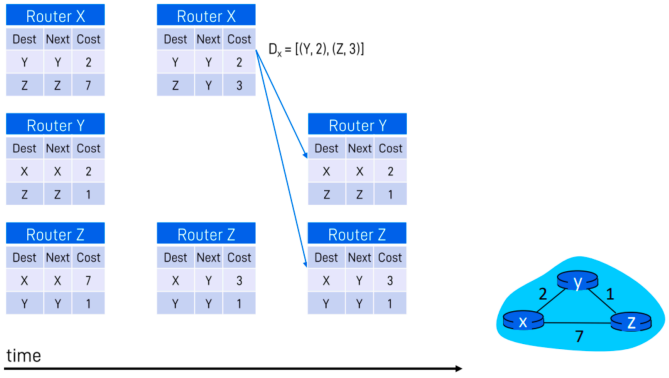


Fig. 5.6: X trasmette il proprio *distance vector*

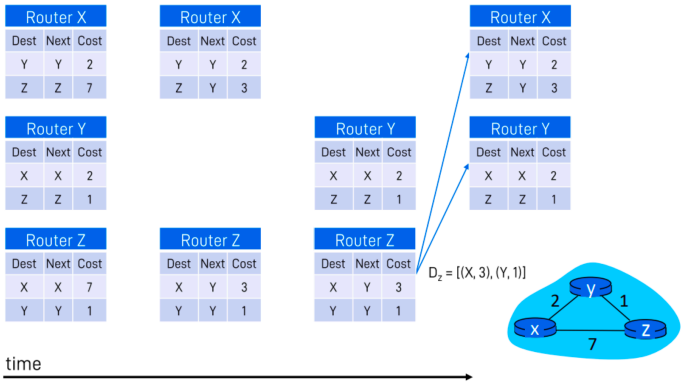
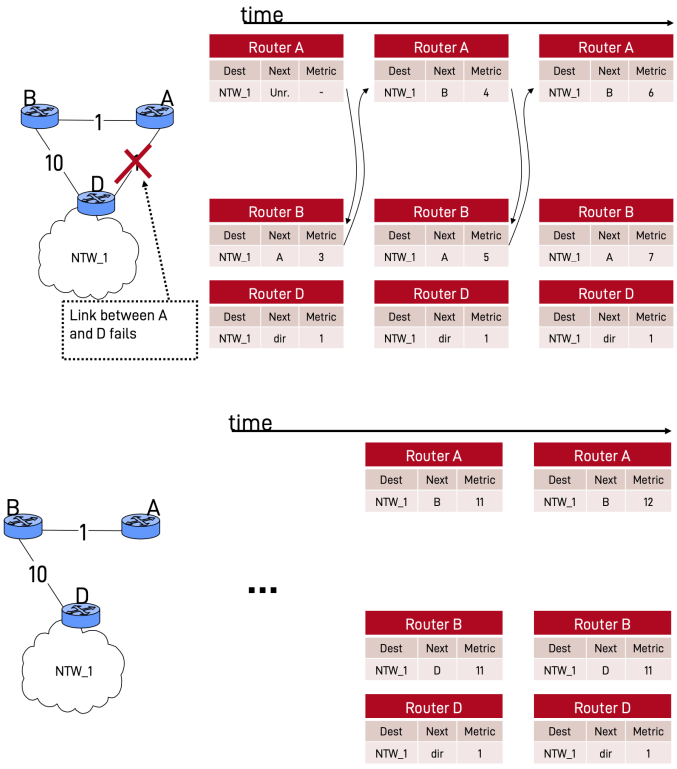


Fig. 5.7: Z trasmette il proprio *distance vector*

5.4.2 Count-to-infinity problem

The count-to-infinity problem happens when a good link breaks down. The hosts that used that link and that are not directly connected are not informed of the change, and therefore continue sending packets in that direction. The router connected to the link that broke, retransmits back the packet. The router that sent the packet updates the cost and if that path is still the cheapest it will retransmit the packet over that path. This will continue until the broken link will be more expensive than the alternatives.



There have been developed some solutions to the count-to-infinity problem:

- Limit the maximum number of hops to 15: this reduces the time to convergence
- Split horizon: when router sends a distance vector to another router, it doesn't send the destinations reachable through that router (in the previous example router B will omit send to router A the route for NTW_1)
- Poisoned reverse: same as in split horizon, but instead of not sending it send that the cost is inf

The propagation of information through the network is slow (especially when network conditions are deteriorated), because it takes time of a system to reach equilibrium. While the routers are figuring out the best route loops can happen, even with the split horizon or poisoned reverse technique.

5.5 RIP protocol

The Routing Information Protocol is a protocol for intra-AS routing (routing internal to an Autonomous system) and is implemented using distance vectors. The protocol is simple to implement and to manage, but has slow convergence and can handle a limited network size.

The cost in the RIP protocol is given by the hop count (limited to 15). Every 30 seconds (or when the routing table changes) the router sends using UDP on port 520 and as destination address the multicast address 224.0.0.9 its distance vector.

5.6 Comparison of link-state and distance vector algorithms

Number of messages:
- link-state with n nodes and e links: O(nE)
- distance vector: exchange between neighbours only

Speed of convergence:
- link-state: $O(n^2)$, but can have oscillations
- distance vector: convergence time varies, loops can happen, count-to-infinity problem

Robustness: if a router malfunctions and advertises a wrong cost, in the distance vector case the error propagates through the network

5.7 BGP protocol

The Border Gateway Protocol is a routing protocol that deals with routing among different autonomous systems. BDP can run both between two peers in the same AS (iBGP or *Interior Border Gateway Protocol*) or between different AS (eBGP or *Exterior Border Gateway Protocol*).

Two BGP routers ("peers") exchange BGP messages over a semi-permanent TCP connection. In these messages they advertise paths to different destination network prefixes (BGP is a path vector protocol).

A route message in BGP includes the prefix and some attributes. Borger routers can accept an advertisement and add it to their own table or ignore it. The can also decide if to advertise that path to other ASes.

A router can learn more that one route to a destination AS, and can pick among them base on different parameters, such as:
1. Local preference
2. Shortest AS-PATH
3. closest NEXT-HOP router (hot potato routing) - Try to reduce time of packet inside the network as mush as possible

5.8 Difference between Intra-, Inter-AS routing

There are 3 main reasons why there have been developed different protocols for Intra- (internal) and Intra- (external) AS routing, namely:
- Policy: administrators want control over how traffic is routed internally, while externally there are no policies needed
- Scale: hierarchical routing saves table size, reducing update traffic
- Performance: in internal networks performance in key, while in inter-AS routing also policies can play a role

6 Data link layer

The data link layer deals with transferring datagrams from one node to the physically adjacent node over a link. The link may be wired or wireless. The packets at layer 2 are called frames and they encapsulate the packet from the higher layer (datagram). Frames have a header and a trailer. To identify source and destination MAC addresses are used.

This division allows a datagram to be transported over different types of mediums (ex. first Ethernet, then wifi) and over connections that offer different types of services, such as reliability.

The link layer is implemented in every host inside an "adaptor" (ex. a network interface card) or directly inside the firmware of a chip (ex. Ethernet). The component is then connected to the host's system bus to provide connectivity to the system.

The communication procedure is similar to the procedure of the other layers. The sending side encapsulates the datagram in a frame, adds error checking bits and information for flow control. The receiving side collaborates in the flow control procedures, extracts the datagram, looks for errors and passes the data to the higher layers.

6.1 Link layer services

The services offered by the link layer are the following:

- Reliable delivery between adjacent nodes: can always be implemented, but usually ignored on links with a low probability of errors (ex. fiber)
- Flow control: transmission speed is adjusted based on the capabilities of the sender and the receiver
- Error correction: receiver can identify and correct single bit errors without the need of retransmission
- Link sharing: nodes can transmit at the same time (full-duplex) or at different times (half-duplex)

6.2 Error detection

Error detection and correction algorithms add some redundancy bits to the message. The receiving side then checks if the ECD bits match the received data. The error detection may not be 100% reliable, but usually increasing the EDC field size improves the accuracy.

6.2.1 Parity bit

This technique can be implemented in one or two dimensions. In the first case a single bit gets added which is 1 if the number of ones in the original string is odd. In the 2D case a parity bit gets added for every row and column.

data bits	parity bit
0111000110101011	1

(a) Controllo a una dimensione

1	0	1	0	1	1
1	1	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

(b) Controllo a due dimensioni

1	0	1	0	1	1
1	0	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

(c) Controllo a due dimensioni con correzione

6.2.2 Redundancy

This technique add redundancy to every character and reorders them. This technique works well in the case of burst errors (errors of bits in succession).

The technique goes as follows:

1. Given a string, add some redundancy to each character: HELLO -> HHH EEE LLL OOO

2. Interleave the characters (reorder them): HEL LOH ELL OHE LLO
3. The receiver receives a modified string: HEL LOH EXX XXX LLO
4. After reordering, the received string becomes: HHX EEX LXL LXL OXO
5. By choosing the most common letter for each group, the receiver is able to reconstruct the original message

6.2.3 Cyclic Redundancy Check

CRC is a more powerful but also more involved error detection technique than the ones seen so far. The components needed for this algorithm are:

- **D**: data we want to transmit in binary
- **G**: "generator" number known both to the sender and the receiver. Its length in binary is $r + 1$ bits

The technique goes as follows:

1. Add r bits to the end of **D**. Mathematically $D \cdot 2^r$ (multiplication shifts bits to the left)
2. Find the remainder **R** by dividing $D \cdot 2^r$ by **G** using modulo 2 arithmetic (instead of subtractions perform XOR operation). Mathematically $R = \text{remainder}\left(\frac{D \cdot 2^r}{G}\right)$
3. Add the remainder **R** to **D**. This can be done by the operation $(D \cdot 2^r)$ XOR R and is represented by the notation $\langle \boldsymbol{D}, \boldsymbol{R} \rangle$.
4. The number we obtained is a multiple of **G**. To check the correctness after transmission we need to divide it by **G** using modulo 2 arithmetic and check that the remainder is zero.

Example:

$$D = 101110, G = 1001 \Rightarrow R = 011$$

```
1 0 1 1 1 0 0 0 0 : 1 0 0 1 = 1 0 1 0 1 1
1 0 0 1
---------
0 0 1 0 1 0
    1 0 0 1
    -------
    0 0 1 1 0 0
        1 0 0 1
        -------
        0 1 0 1 0
          1 0 0 1
          -------
          0 0 1 1
```

$$\text{Transmitted message} : 101110\underbrace{000}_{\text{r bits}} \text{ XOR } 011 = 101110\underbrace{011}_{\text{CRC}}$$

If we receive the message without errors and we divide by the generator, we get that the remainder is zero. Otherwise it its very likely that the result is not a multiple of the generator, thus the remainder will not be zero, signalling that an error happened in the transmission.

```
1 0 1 1 1 0 0 1 1 : 1 0 0 1 = 1 0 1 0 1 1
1 0 0 1
---------
0 0 1 0 1 0
    1 0 0 1
    -------
    0 0 1 1 0 1
        1 0 0 1
        -------
        0 1 0 0 1
          1 0 0 1
          -------
          0 0 0 0
```

This technique is able to detect all burst errors up to r bits. If an appropriate generator number is chosen, it can detect also all single and two bit errors.

6.3 Multiple access protocols

We can distinguish two types of connections:

- Point-to-point: nodes are connected directly (ex. PPP - Point to point protocol)
- Broadcast: nodes share the same wire or medium (ex. Ethernet, 802.11 wireless LAN)

In broadcast connections nodes have to share the same medium, therefore to avoid interference a protocol to regulate how hosts share the channel is needed. These protocols are called multiple access protocols (MAC).

An ideal multiple access protocol with bandwidth *R* has the following characteristics:
1. When one node wants to transmit, it can transmit at rate R
2. When M nodes want to transmit, each one can transmit at an average of *R/M*
3. The protocol should be decentralised
4. The protocol should be as simple as possible

We can distinguish three types of MAC protocols:
- Channel partitioning: channel is divided into smaller pieces (time slots, frequency, code)
- Random access: nodes use the channel at random, but can detect collisions and recover
- Take turns: nodes take turns, with nodes allowed to take longer turns if they need to send more data

6.3.1 Channel partitioning protocols

In this section two techniques will be treated: TDMA and FDMA.

In TDMA (time division multiple access) access the channel in rounds. Each station is assigned a fixed time slot. If a station doesn't have to transmit, the channel stays idle during its time slot.

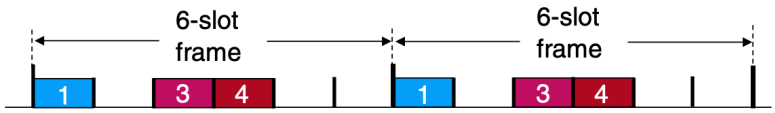
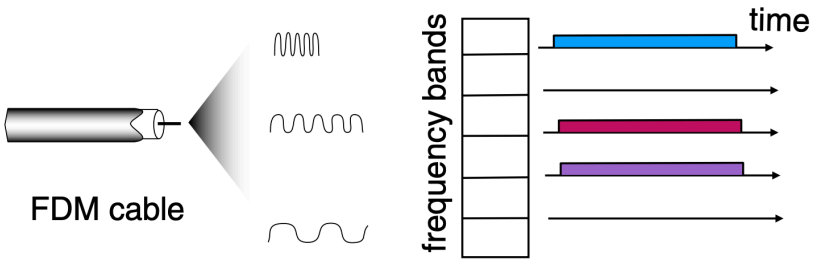


Figure 41: Example of a 6 station LAN using TDMA.

In FDMA (frequency division multiple access) the channel is dividend into frequency bands, where each station is assigned a fixed frequency band. If a station doesn't have to transmit, its assigned frequency band stays idle.

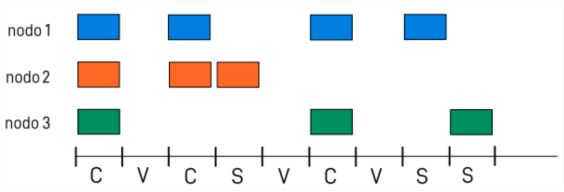


6.3.2 Random access protocols

In random access protocols when a node as a packet to send it will transmit it at the full channel data rate R, without coordinating with the other nodes. If two or more nodes try to transmit at the same time, a collision will happen. These protocols define how to detect collisions and recover from them.

6.3.2.1 Slotted ALOHA

In slotted ALOHA time is divided into slots of the same length. All hosts are synchronised and can transmit frames only at the beginning of a time slot. If a collision is detected, every node will retransmit the frame at the next slot with probability p until success.



The advantages of this technique are that a single node can transmit at full rate and that it is simple to implement. The disadvantages are that there are frequent collisions, clients are forced to wait the beginning of a slot to transmit and clock synchronisation is needed.

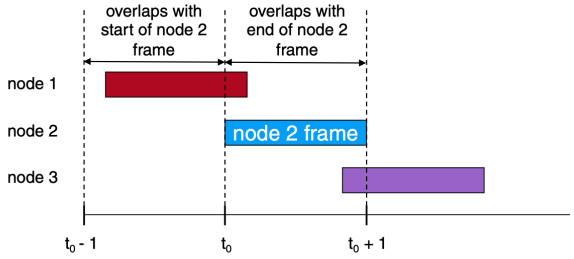
Assuming that all frames have the same size, that there are infinite stations, and that each frame is transmitted independently from the others, the probability of k frames being sent at a certain instant is poisson distributed, with probability mass function

$$P[k] = \frac{\lambda^k e^{-\lambda}}{k!}$$

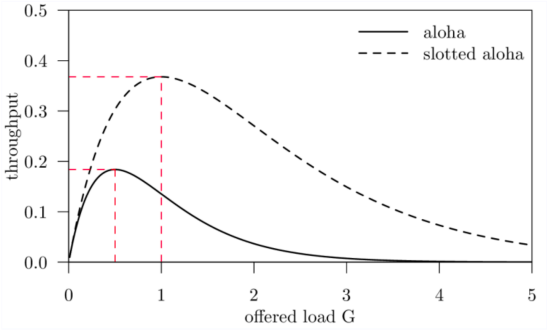
where λ is the average number of frames sent by all hosts in a defined time interval. Therefore, the probability of having just one frame being transmitted in a slot is $P[k=1] = \lambda e^{-\lambda}$. This probability is maximised when λ is 1. Plugging this value in the PMF, we obtain $P[k] = \frac{1}{ek!}$. Therefore the probability of having no collisions is $P[k=1] = \frac{1}{e} \approx 0.368$. Since this probability tells us on average how many frames will be actually sent successfully this probability is called throughput.

6.3.2.2 Pure ALOHA (unslotted)

In unslotted ALOHA frames again have all the same size, but nodes can transmit at any time. The collision probability increases because a frame sent at t_0 will collide with frames sent during $[t_0 - 1, t_0]$.



By doing the same assumptions as before, the probability of sending just one frame at time t_0 is $P[k=1] = \lambda e^{-\lambda}$. The probability that no frames have been sent in the interval $[t_0 - 1, t]$ is $P[k=0] = e^{-\lambda}$. The probability that the transmission is successful is therefore $P[k=1] \cdot P[k=0] = \lambda e^{-2\lambda}$. This probability is maximised when $\lambda = \frac{1}{2}$. Therefore the probability of having no collisions is $P[k=1] \cdot P[k=0] = \frac{1}{2e} \approx 0.184$, which is half the probability of slotted ALOHA.

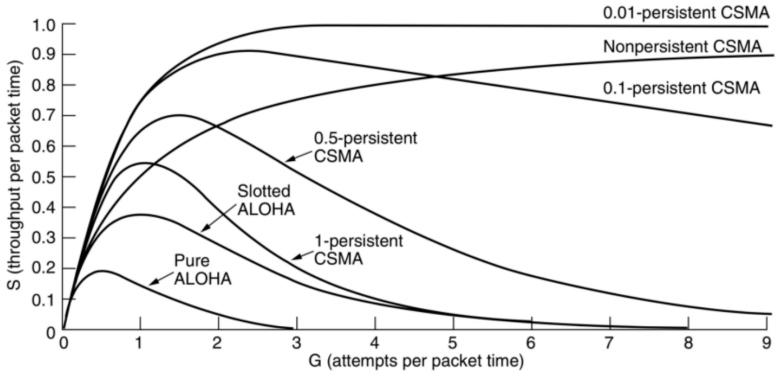


6.3.2.3 CSMA

In carrier sense multiple access nodes check if the channel is idle before transmitting. Various implementations of CSMA exist:

- non-persistent CSMA: if channel is busy defer the transmission by a random time much larger than the transmission time
- 1-persistent CSMA: if channel is busy wait for channel to become free and then transmit the frame
- p-persistent CSMA: if channel is busy wait for channel to become free and transmit the frame with probability p. If frame is not transmitted defer the transmission by a random time much larger than the transmission time

If a collision happens wait a random time and try again.



CSMA has the following problem: if a host starts to transmit but the signal has not yet reached the other hosts, the other hosts will start transmitting at the same time because they think that the channel is idle. This period is called the vulnerability period and is given by the propagation time τ and by the time T_a needed for a host to sense that the channel is busy, therefore $T_v = \tau + T_a$. In general CSMA works well when the propagation time is much smaller than the transmission time. When collisions occur, the entire transmission time is wasted. Therefore, two additional versions of CSMA have been developed: CSMA/CD and CSMA/CA.

CSMA/CD deals with collision detection: if a collision is detected, the transmission is stopped immediately, reducing the time wasted. This is easy in wired LANs because it is possible to measure the signal strength on the channel and compare it to the transmitted signal strength. In wireless LANs this is much more difficult.

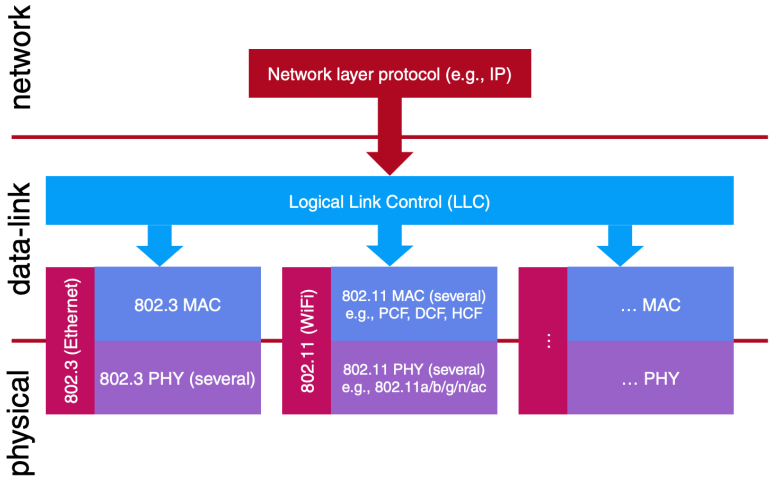
CSMA/CA deals instead with collision avoidance: in channels where is hard to detect collisions (such as wireless LANs and networks where $T \ll \tau + T_a$) p-persistent CSMA is used, where p is adapted to network conditions (p decreases at each retransmission attempt).

6.3.3 “Taking turns” protocols

Polling is a technique where a master node tells to slave nodes when to transmit. It it used when slaves have low computation power, but it introduces a single point of failure, polling messages generate overhead and some latency is introduces. Token passing is a technique where the host holding the token is allowed to transmit. Whet its done transmitting the host gives the token to the next host. Again, the passing of the token generates overhead, latency and introduces a single point of failure (the token itself).

6.4 IEEE 802 standards

In IEEE 802 standards the data link is divided in two layers: logical link control and the medium access control (MAC) of the medium. The LLC deals with multiplexing the packets it receivers from the network layer to the various mediums, such as 802.3 Ethernet or 802.11 Wireless LAN.



6.4.1 Ethernet (IEEE 802.3)

Ethernet is the dominant wired LAN technology: it is simple and cheap and nowadays it can reach speeds of 10Gbps and more.

6.4.1.1 Ethernet frame

The Ethernet frame is structured in the following way:

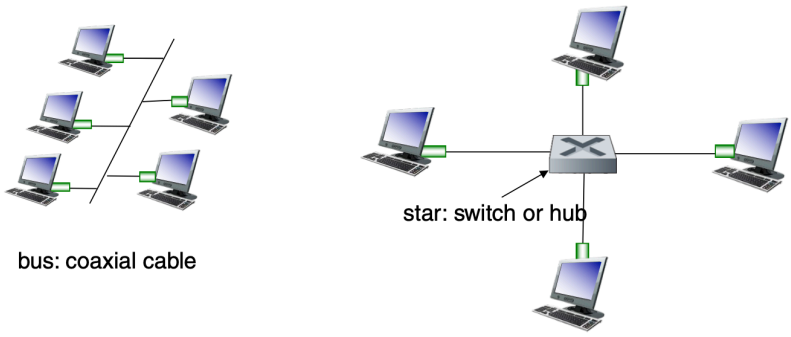
- Preamble: sequence of predefined bits used to synchronise the sender and receiver
- Source and destination MAC address
- Type: higher level protocol
- CRC: used for error detection

6.4.1.2 Ethernet characteristics

Ethernet is connectionless (does not do handshaking) and unreliable (no acks or nacks). Ethernet uses unslotted CSMA/CD. This means that if the network card (NIC) has a frame to transmit it will check the status of the channel: if it is free, it will transmit the frame immediately, otherwise it will transmit it as soon as the channel becomes free. If a collision is detected during the transmission, the NIC terminate the transmission, send a jam signal and will choose randomly a backoff value K from the set $\{0, 1, 2, ..., 2^{\{n-1\}}\}$, when n is the number of collisions. The NIC will then wait $K \cdot 512$ bit. Therefore the time grows exponentially.

6.4.1.3 Ethernet switches

Initially Ethernet was designed using the bus topology: that means that all hosts were connected to the same wire. Nowadays, the star topology is used.



There exists two types of devices to connect multiple hosts over Ethernet using this topology: hubs and switches. If hosts are connected through a hub, the hub simply retransmits everything it receives on all the other ports, thus the collision domain is the same (as in the bus topology).

The switch is more active device: it examines the incoming frames and retransmits them only to the appropriate outgoing links. This process is transparent, which means that hosts are unaware of the presence of the switch. This means that each host is on a different collision domain, therefore, if the connections are full-duplex, no collisions are possible. The switch uses internal buffers if it is receiving too many frames.

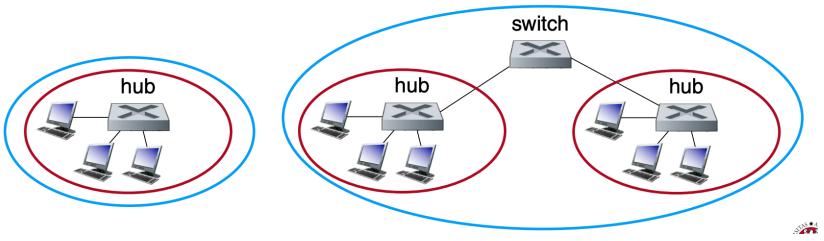
The switch a plug-and-play device and is able to automatically learn which hosts are reachable through which interfaces and build the switch table accordingly. If the switch does not know where the destination is, it will send the frame to all interfaces except from where the frame arrived (flooding). If the destination is the same as the source, the frame will be dropped. Entries in the switching table have a TTL value, similarly to the routing table.

6.4.1.4 Switches vs. routers vs. hubs

Routers are network-layer devices (examine network-layer headers), switches are link-layer devices (examine link-layer headers) and hubs simply retransmit frames on all interfaces. The forwarding table of routers is assembled using routing algorithms and contains IP addresses, switches assemble their table by flooding and contain MAC addresses, while hubs have no routing table.

6.4.1.5 Broadcast vs collision domains

The collision domain is the portion of the network in which, if two hosts transmit simultaneously, a collision will occur. The broadcast domain is the portion of the network that can be reached by a layer-2 broadcast.

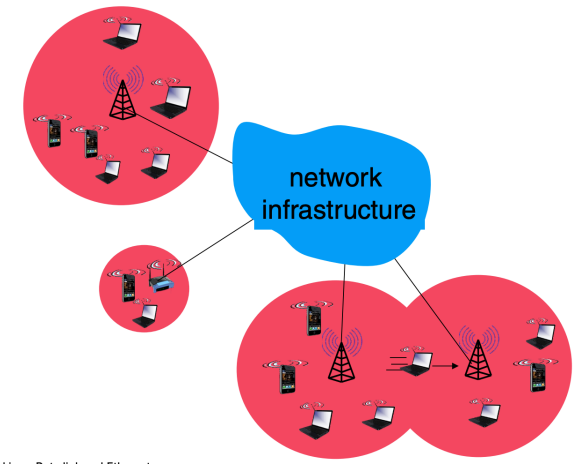


6.4.2 WiFi (IEEE 802.11)

A wireless network is composed of:

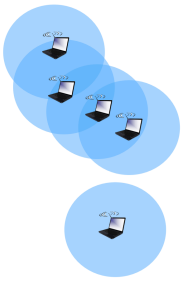
- Wireless hosts: devices that connect to the network, can be stationary or fixed (ex. smartphones, laptops)
- Base station: devices that are typically connected to a wired network and relay packets from the wired network and the wireless hosts in their area (ex. cell towers, 802.11 access points)

- Wireless link: connect wireless hosts and base stations to other base stations. Access to the medium is managed by MAC protocols



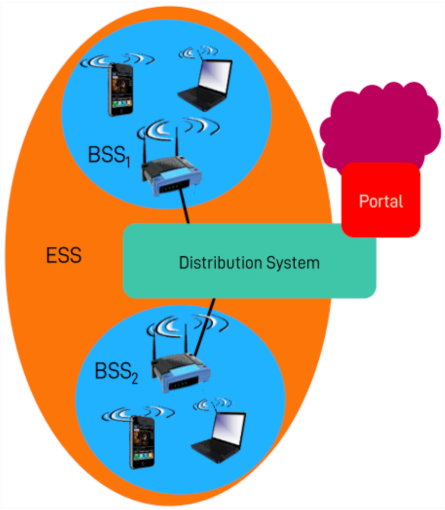
A wireless network can be structured in two ways:

- Infrastructure mode: base station connects hosts to the wired network, and hosts with a procedure named handoff can change base station
- Ad hoc mode: there are no base stations, but hosts communicate directly to their neighbours



6.4.2.1 Reference architecture of WLANs

Usually WLANs are based on a standard architecture model. In this architecture all hosts and base stations are referred to as STA (stations). All stations sharing the same MAC protocol and accessing the same medium are referred to as BSS (Basic service set). In a BSS an access point can be present, which connects the BSS to the distribution system. The distribution system is a wired backbone LAN which connects two or more BSSs. A group of connected BSSs is referred to as ESS (Extended service set). ESS appears as a single logical LAN at the logical link control (LLC) level (ex. a WiFi network is made of multiple BSSs, but if appears at the higher level as a single network).



6.4.2.2 Protocols for wireless communications

Wireless communications are regulated by the 802.11 standard, which divides the spectrum into different intervals of frequencies called channels. If two neighbouring access points use the same channel, interference may happen.

Hosts connect to access points by scanning the channels and listening for beacon frames containing the AP's name (SSID) and the MAC address. Then the hosts chooses the AP to associate with, authenticates and typically uses DHCP to obtain an IP address. There also exists an active scanning mode: the host sends a probe request using broadcast, to which the access points reply with a probe response.

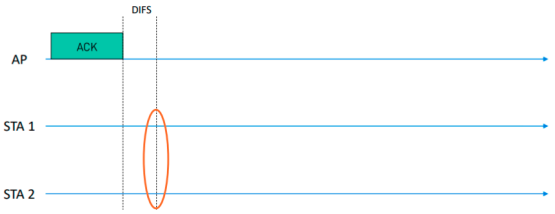
There are some key differences between wireless and wired communications: in wireless communications the signal strength is attenuated as it propagates through matter, the signal propagates in different ways over different paths (ex. reflections) and there is a lot more interference from other sources (ex. motors, lights and other access points). Moreover, collision detection in wireless network is impossible to achieve, because a transmitting antenna cannot act as a receiver at the same time. Even if we had an access point with two antennas (one transmitting and one receiving), the power of the transmitted signal will be of many order of magnitude higher than any other signal received by the received antenna (note that radio waves are attenuated quadratically). Therefore, the only option to avoid collisions is to reduce the probability of interference by using CSMA/CA.

6.4.2.3 Collision avoidance in wireless networks

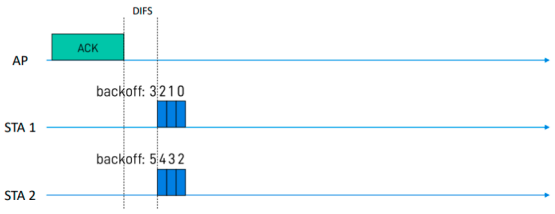
The MAC 802.11 protocol is based on CSMA with collision avoidance. The procedure goes as follows:

1. If the channel is free for a certain amount of time (DIFS - Distributed Inter-frame space) the the sender will transmit the entire frame. If the channel is busy, the sender will pick a random backoff time, wait for the channel to be free again and additionally wait for the backoff time.
2. If the receiver has received the frame successfully, it will send back an ACK after a short interval than DIFS (otherwise other frames would be transmitted before the ACK). The ACK is needed because of possible backoff collisions, channel errors (signal was to weak to reach the receiver) or hidden terminal problems. If no ACK has been received, double the backoff interval and retry sending the frame

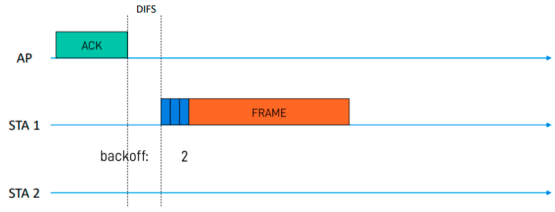
Example:



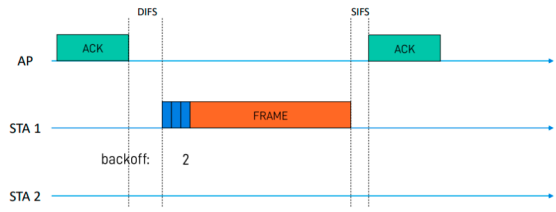
(a) Situazione iniziale



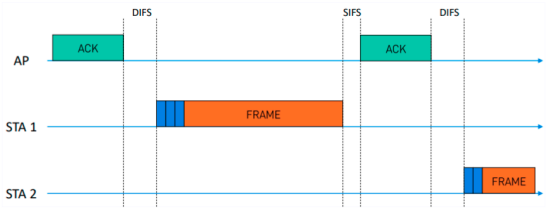
(b) Scelta dei backoff



(c) `STA 1` trasmette



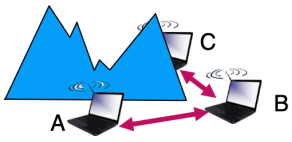
(d) L'AP risponde con un ACK

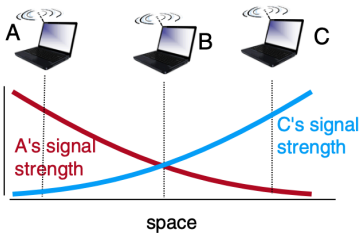


(e) `STA 2` trasmette

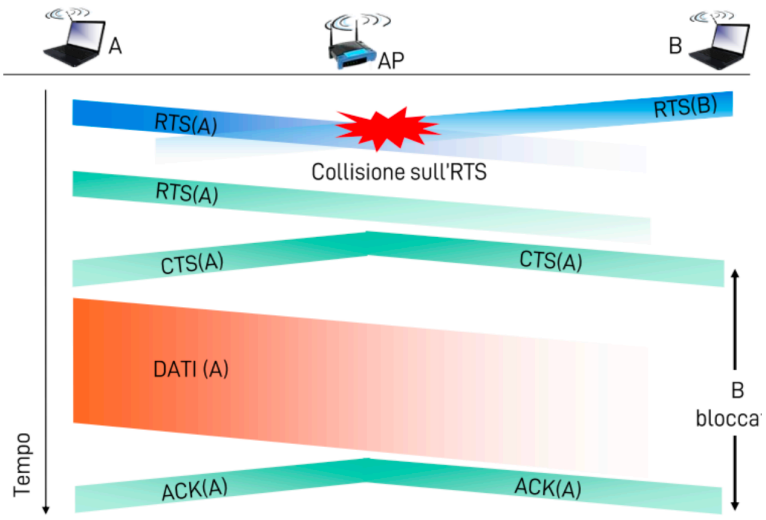
6.4.2.4 Hidden terminal problem

The hidden terminal problem arises when two stations cannot hear each other (A and C), but a third station can hear both (B). This will cause deterministic collisions, which could be avoided by using CSMA/CA with handshaking.





The handshaking requires that before transmitting the station has to send a short RTS (Request to send), to which the receiver has to reply with CTS (Clear to send). Therefore, even if a station could not hear the RTS, it will hear the CTS, thus it will know that the channel is busy for a certain amount of time. If two RTSs collide there will be no CTS, but not much time has been wasted because these packets are very short. Example:



6.4.2.5 802.11 frame

In an 802.11 frame there are four MAC addresses:

- address of the sender
- address of the destination
- address of router to which the access point is connected to
- used only when access point is connected to another access point: address of second access point