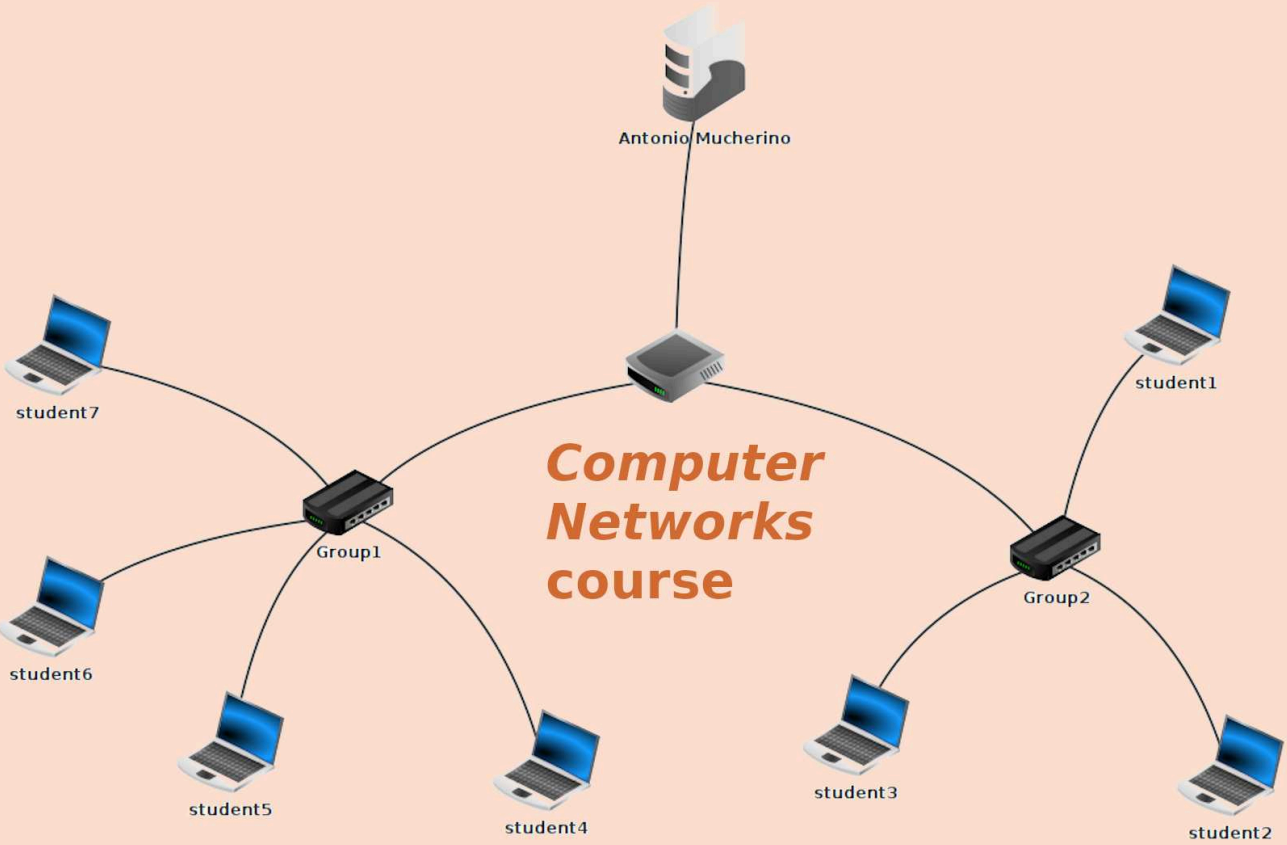
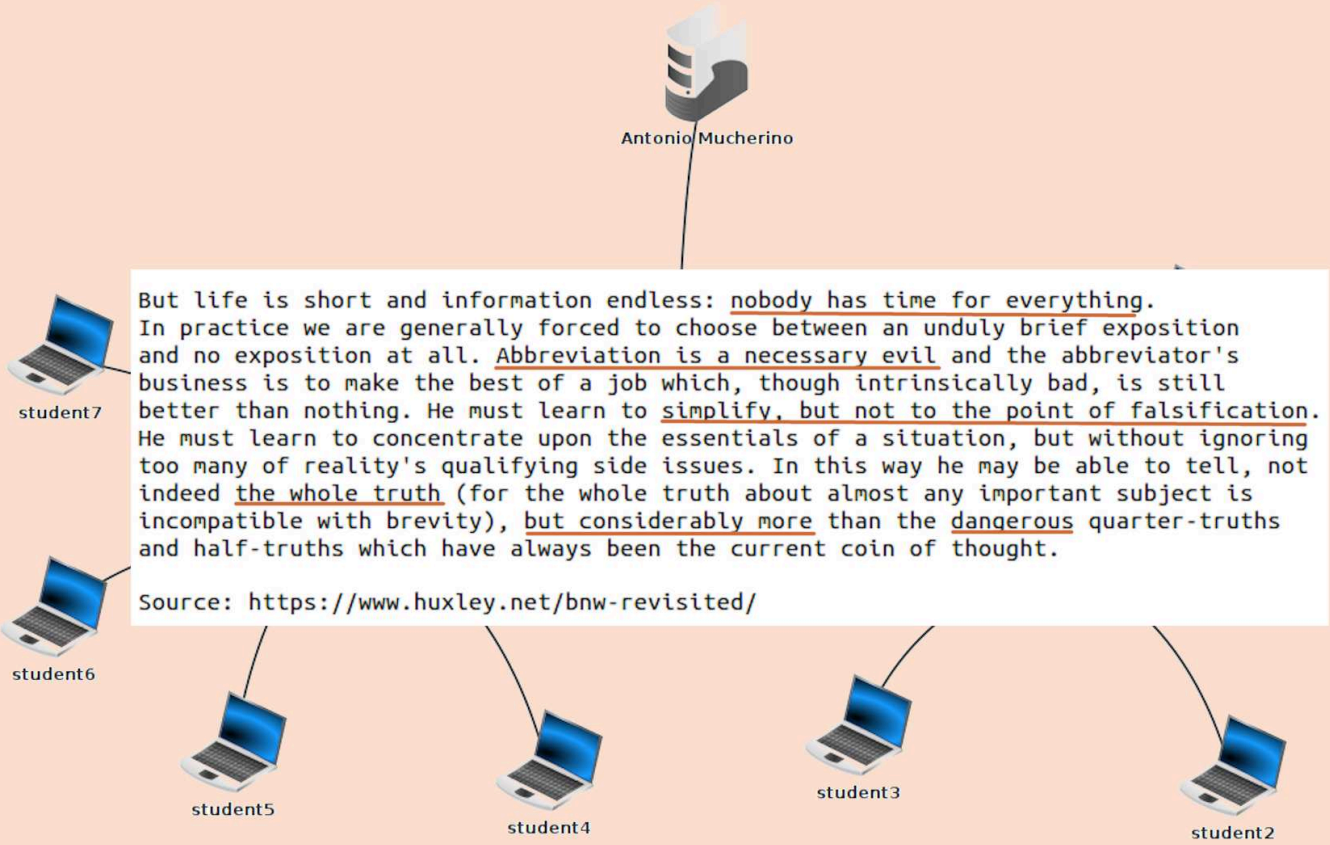
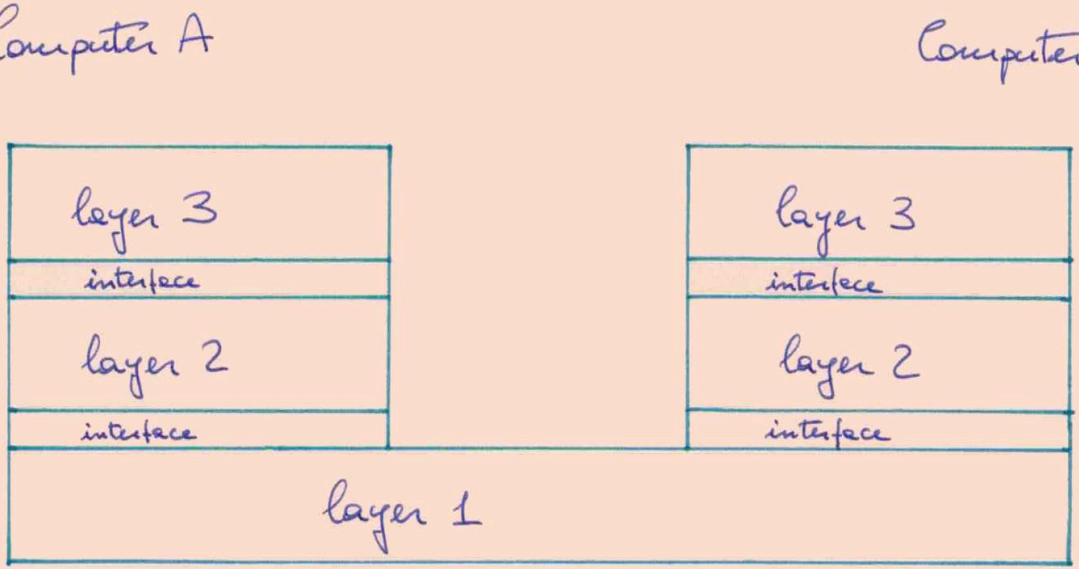


... with some drawings generated with the software ***Filius***



... with some drawings generated with the software ***Filius***

Network layers



The lowest layer is the physical layer

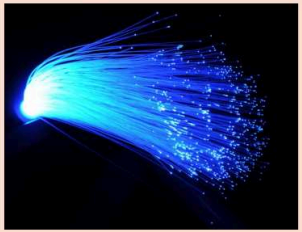
How to make a message pass through, from a physical point of view ?

What physical supports can we use nowadays ?

Cable

Radio waves

Light



What kind of signal these three physical devices are able to carry ?

A wave-like signal

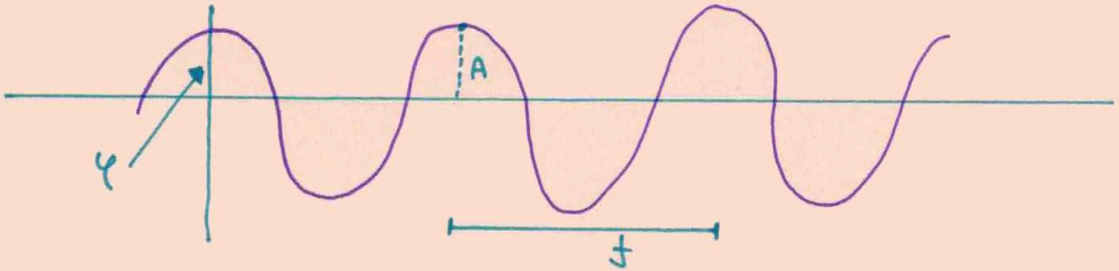
The sinusoidal wave

It's a mathematical function describing a smooth periodic oscillation:

$$y(t) = A \sin(2\pi f t + \varphi)$$

where:

- A is the amplitude
- f is the frequency (number of oscillations per unit of time)
- φ phase (position for $t = 0$)

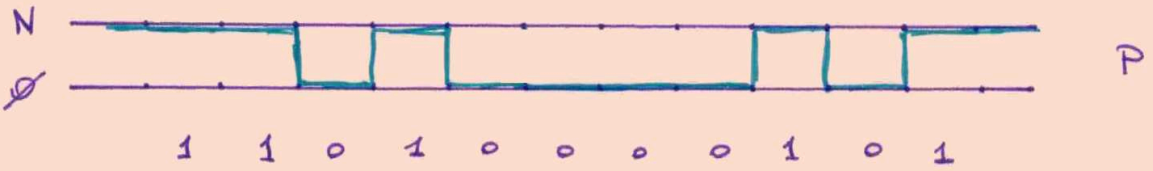


For simplicity, we'll focus in the following on the square wave.

In the digital world, a piece of information is basically a sequence of bits.

How to convert this piece of information in a signal ?

The simplest code :



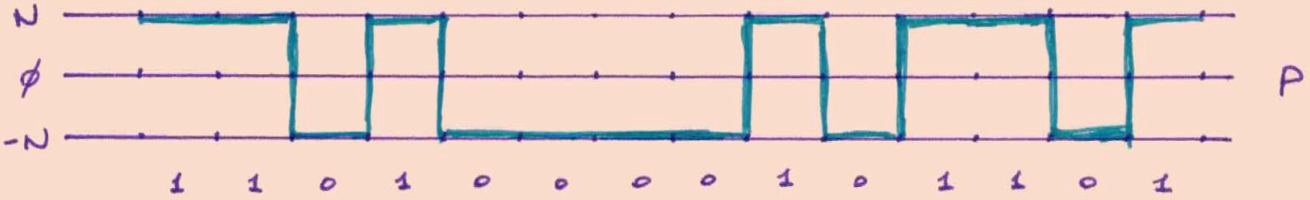
where Ø and N are the two possible values for the property P.

But this code is too simple :

> How to distinguish between the bit Ø and the absence of signal ?

Le code NRZ : Non-Return-to-Zero

The bit 0 does not correspond to $P = \phi$, but rather to $P = -N$.



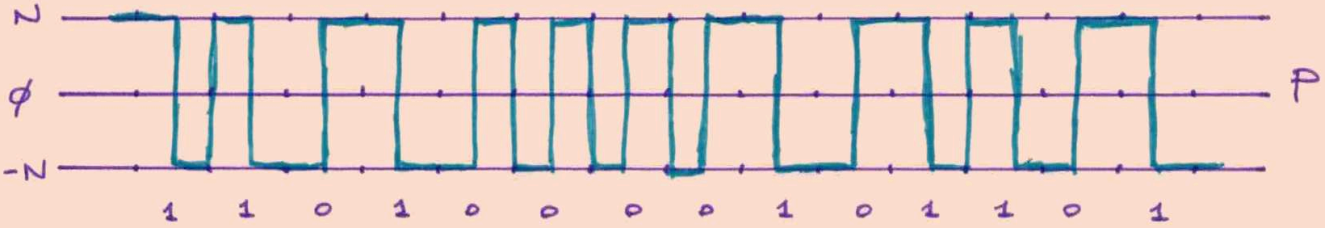
This solves the problem of distinguishing with an absence of signal.

However, when the sequence contains several bits of the same kind, how to correctly count the number of bits ?

The synchronization between sender and receiver becomes crucial !

The Manchester code

It's not the value of P that defines the bit, but rather the variation in value.

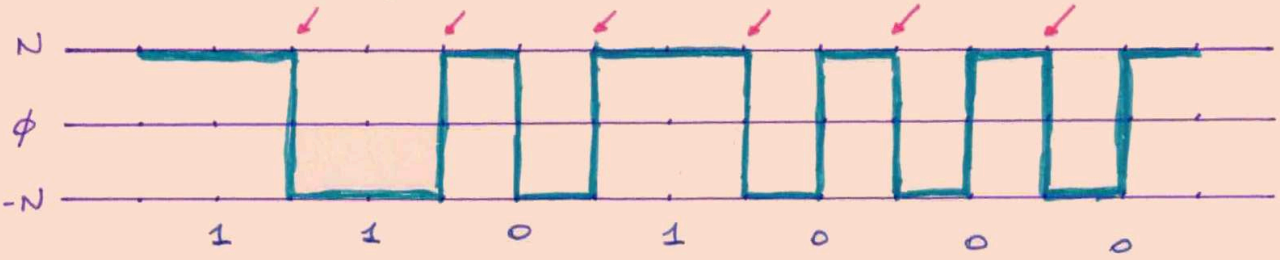


A variation between N and -N indicates 1;
a variation between -N and N indicates 0.

The "differential" Manchester code is a further improvement to the synchronization.

The differential Manchester code

In order to improve the synchronization, one bit covers two clock ticks, and we make sure that there is a variation of value at least every two ticks.

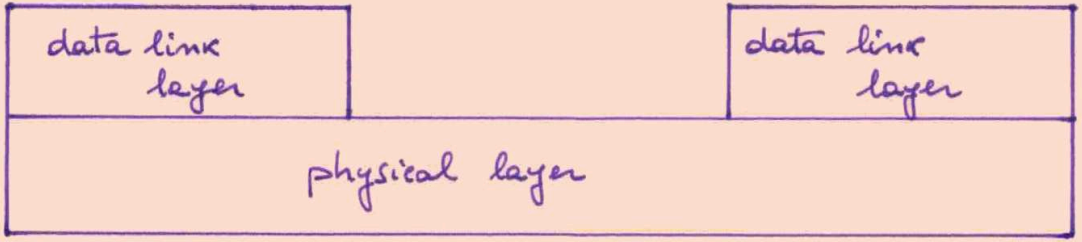


No variation during two consecutive clock ticks indicates 1; a variation (from -N to N or from N to -N), indicates 0.

Notice that the mirror image of the drawing (with symmetry axis ϕ), represents exactly the same sequence of bits.

Network layers: the lowest

The data link layer establishes the "link" between two directly connected (via the physical layer) nodes. It can detect, and possibly correct, errors that may occur in the physical layer.



The physical layer is responsible for the transmission and reception of raw data between a device and a physical trasmission medium.

Building an interface

First question: what about the length of the messages?

Some messages may be short, others too long...

If we separate the messages in a sequence of frames, what about the length of the frames?

Possibility 1

All frames have the same length.

Possibility 2

The header of every frame contains its length.

Possibility 3

Use flags to delimit the beginning and the end of the frame.

Typical flag: 01111110.

Building an interface

Second question: what if an error occurs when the frames are sent out via the physical layer?

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

How to detect that an error has occurred?

A simple strategy: the Parity Bit

We add an extra bit to all frames, which we name "the parity bit", so that the number of 1s in the frame is always even.

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

With the 3 errors marked in red above, the number of 1s would be 9, which is odd, so that a transmission error would be detected.

The 1-bit parity check is not very efficient...

- it can help in finding out that there is an error in the bit sequence, but where?
- if the number of errors is even, no errors are detected;
- there is no mechanism to verify if the parity bit itself was flipped during the transmission...

A more advanced parity check

For every 16-bit sequence,

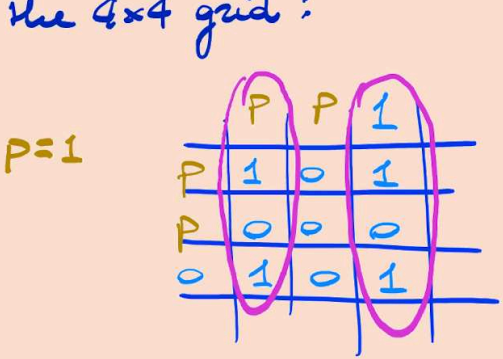
we use:

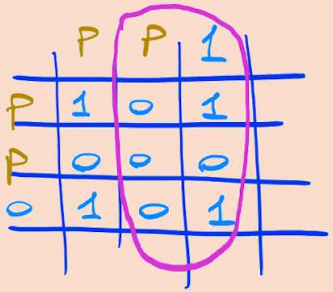
- 11 bits from original message
- 5 parity bits P

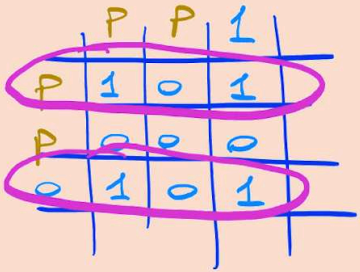
P	P	P	1
P	1	0	1
P	0	0	0
0	1	0	1

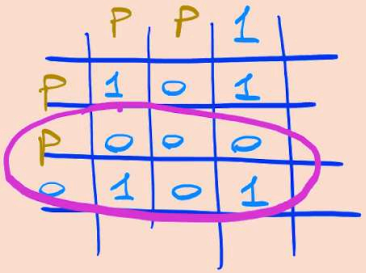
Parity checks on a 4×4 grid

Every parity bit is in charge to verify a specific region in the 4×4 grid:

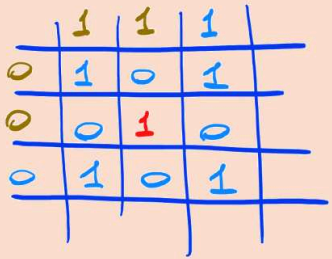




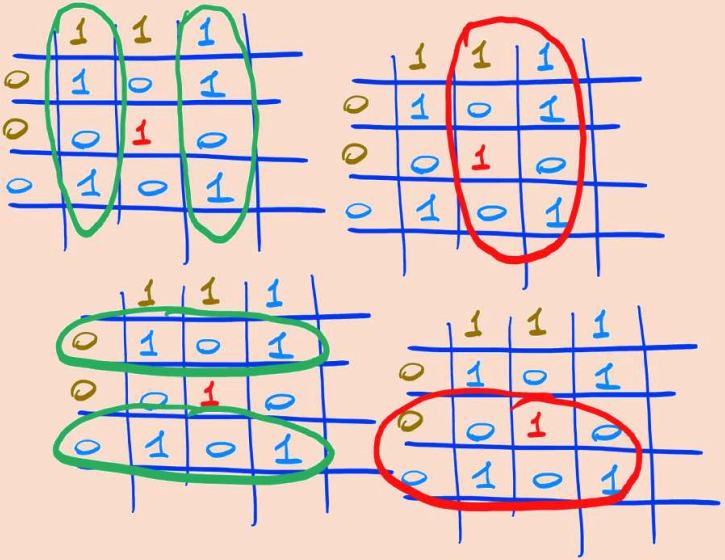




Let's introduce an error in the sequence!



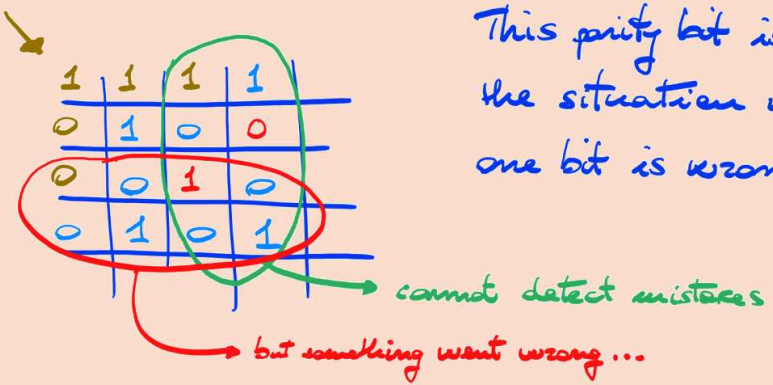
We perform the following 4 checks



This parity check is able to tell us which is the wrong bit!

What about the top-left parity bit?

It is exploited as a "basic" parity bit over the entire grid.



This parity bit is useful to detect the situation when more than one bit is wrong.

Since the top-left parity check is even, we know that there must be at least two mistakes in the sequence of bits.

However, in this situation, we cannot correct these mistakes.

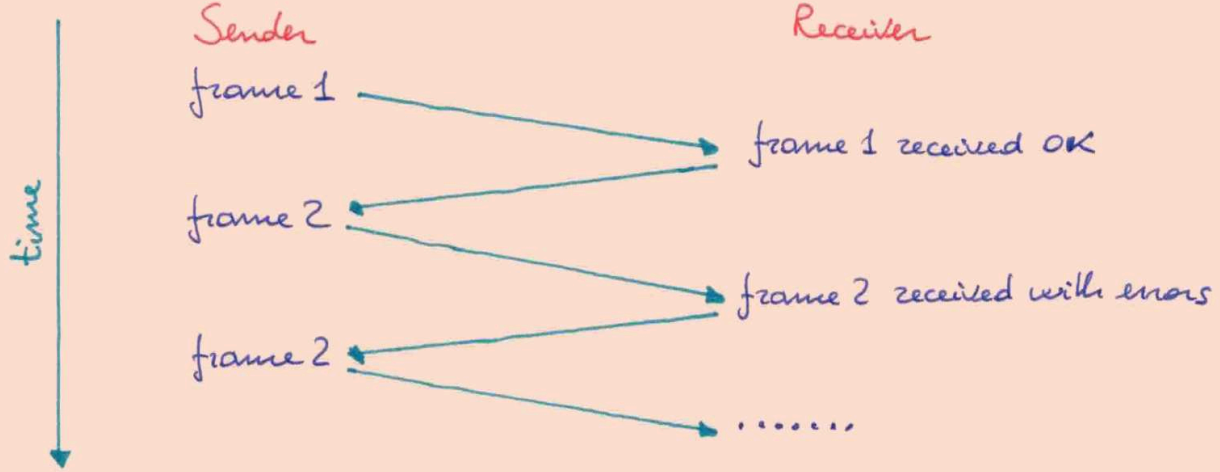
THIS PARITY CHECK CAN BE IMPLEMENTED IN HARDWARE.

Building an interface

Another question: what to do when an error is detected ?

The receiver may try to correct the errors, but this can be complex, and with too many errors, even impossible...

Another simple approach : retransmit frames received with errors

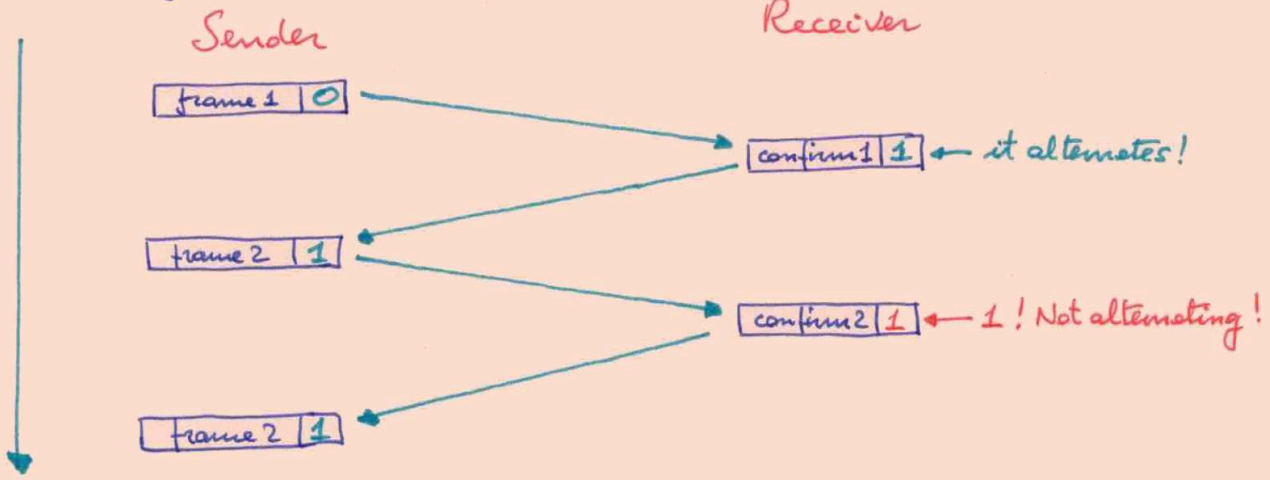


Building an interface

And how to notify the sender that the frames are well-received, or received with errors?

The alternating bit protocol.

Again, we add an extra bit to the frame, which is supposed to keep alternating as far as no errors are detected.



Physical / data link layers : summing up

Physical → Data link

- receives frames coming from another device
- verifies whether they contain errors
- reorders the frames transmitted out of sequence
- may attempt the correction of frames with errors

Data link → Physical

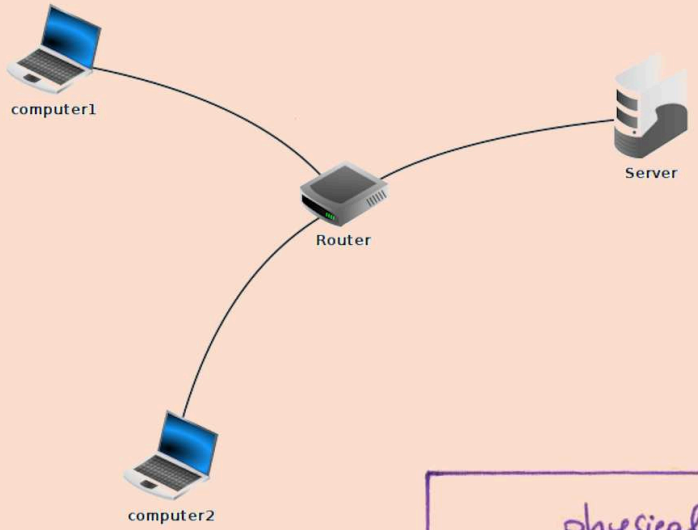
- splits the messages in several frames
- adds information to frames for error detection (parity bit, alternating bit)
- generates confirmation messages (alternating bit)

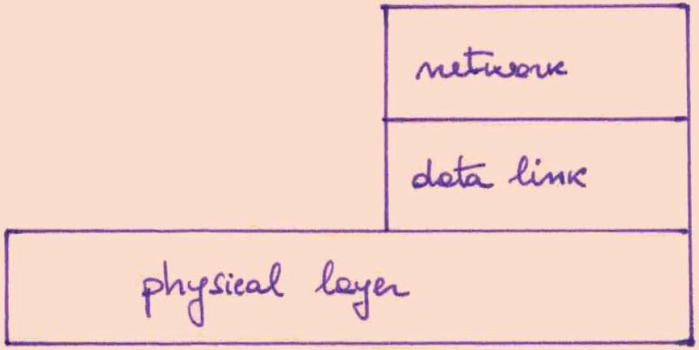
The Point-to-Point Protocol (PPP) works at the level of the data link layer, because it is capable to directly connect two devices without any host in between.

"The Network Layer"

This is the first layer that is "aware" of the network.

The network is a graph.





THE GRAPH

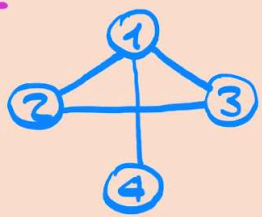
A graph is a pair $G = (V, E)$ consisting of

- a vertex set V; if $v \in V$, we say that v is a vertex of the graph
- an edge set E
  - the maximal cardinality of E is $|V \times V|$;
  - the elements of E are generally represented by $\{u, v\} \in E$

Numerical values can be associated to both vertices and edges.

In this case, we refer to such values as weights (either on vertices, or on edges)

Example



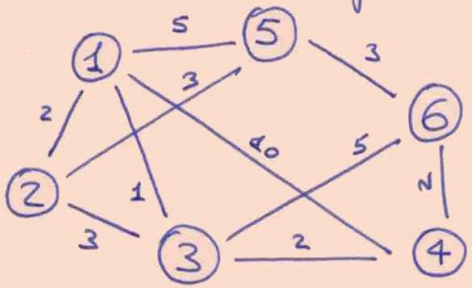
$V = \{1, 2, 3, 4\}$

$E = \{\{1,2\}, \{1,3\}, \{1,4\}, \{2,3\}\}$

The Network Layer and the Graphs

When two computers need to exchange data through the network, what is the best route in the network for such data?

This corresponds to finding the shortest path between two vertices u and v on the graph G representing the network:



Which is shorter?
① - ⑤ - ⑥ or ① - ③ - ⑥ ?
any other possibility?

An important point: what are the weights representing in the graph?
These weights are supposed to represent the "cost" of passing through the corresponding edge.

Moore's Algorithm A

We want to find the best path, the shortest path, from the vertex a to the vertex b of a given graph $G = (V, E, w)$.

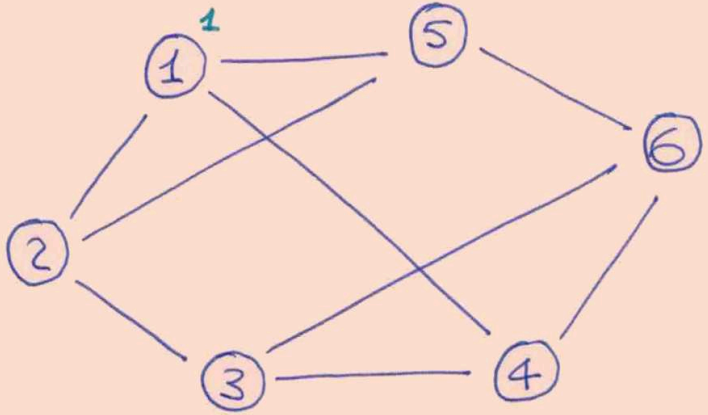
We suppose that all edges have the same weight, so that the cost of the path can be computed as the number of "hops" necessary to go from a to b.

Moore's proposition:

- assign the label 1 to vertex a; let $l = 1$
- while (vertex b is not labeled)
  - let $l = l + 1$;
  - assign label l to all vertices with labeled adjacent vertex;
- backtrack to a by following the path given by decreasing labels.

Running the algorithm

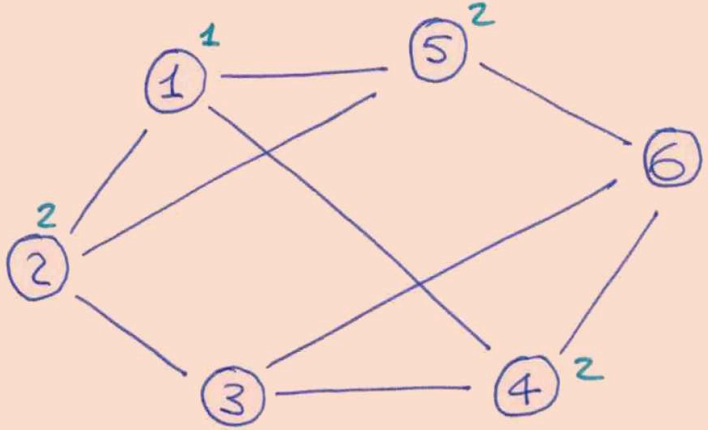
We consider our previous example, but without weights on the edges :



What is the best path between (1) and (6) ?

Running the algorithm

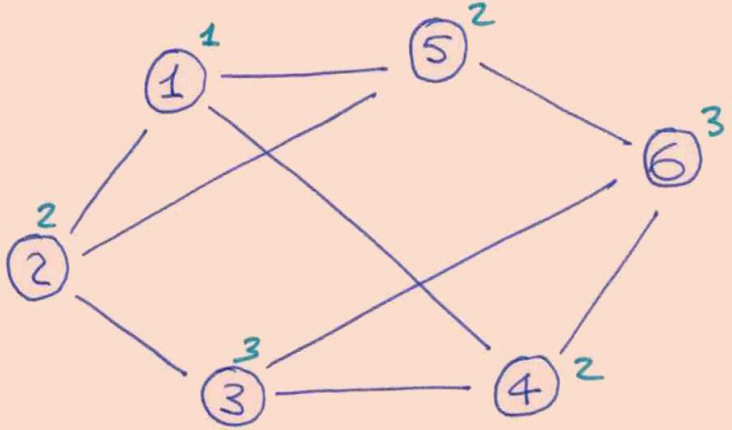
We consider our previous example, but without weights on the edges:



What is the best path between (1) and (6) ?

Running the algorithm

We consider our previous example, but without weights on the edges:



What is the best path between (1) and (6) ?

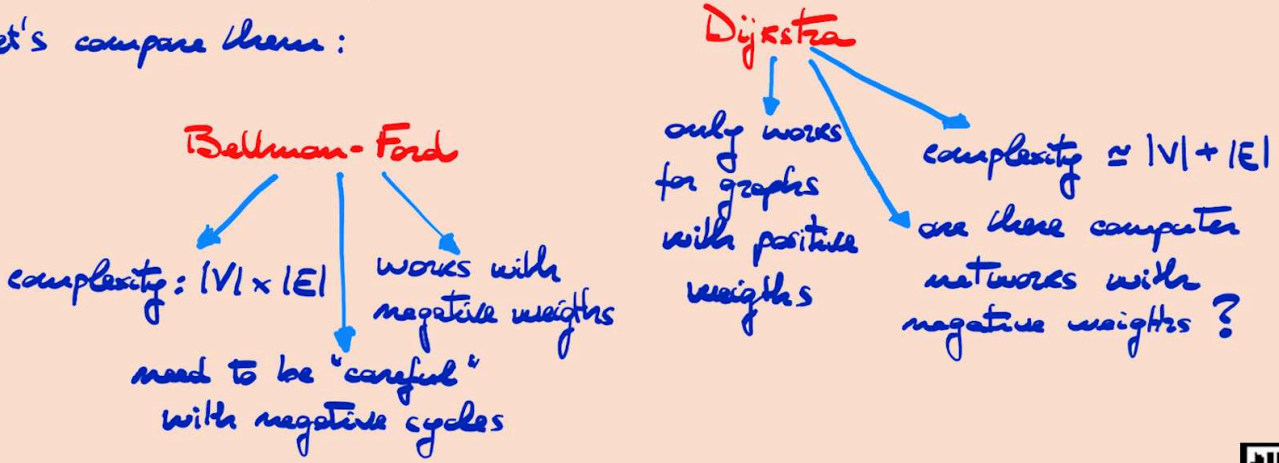
Other possible algorithms for shortest paths on graphs

Moore's algorithm A only counts the number of "hops" (number of crossed edges), but what if our edges are weighted ?

The very important algorithms that we can use are :

- the Bellman-Ford algorithm
- and Dijkstra algorithm

Let's compare them :



Moving to real Networks

In computer networks, we need to distinguish among the following devices:

Generic computer

Server

Switch

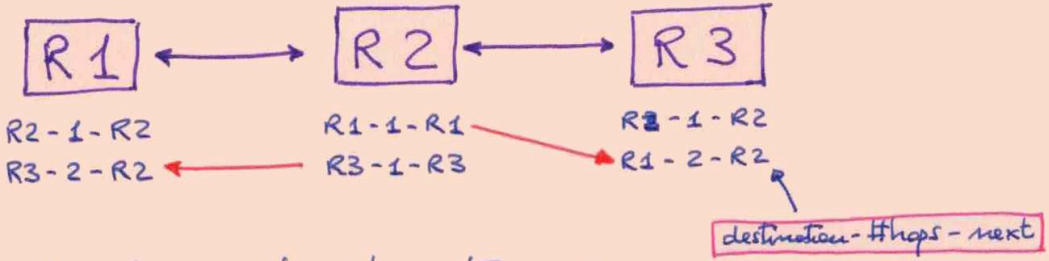
Router

The shortest paths are generally searched between two routers of the network.

The flooding approach is instead used for devices connected to a common switch.

Routing Information Protocol (RIP)

RIP is based on the idea of counting the number of "hops" between two routers (just like Moore's algorithm), but it is not static.



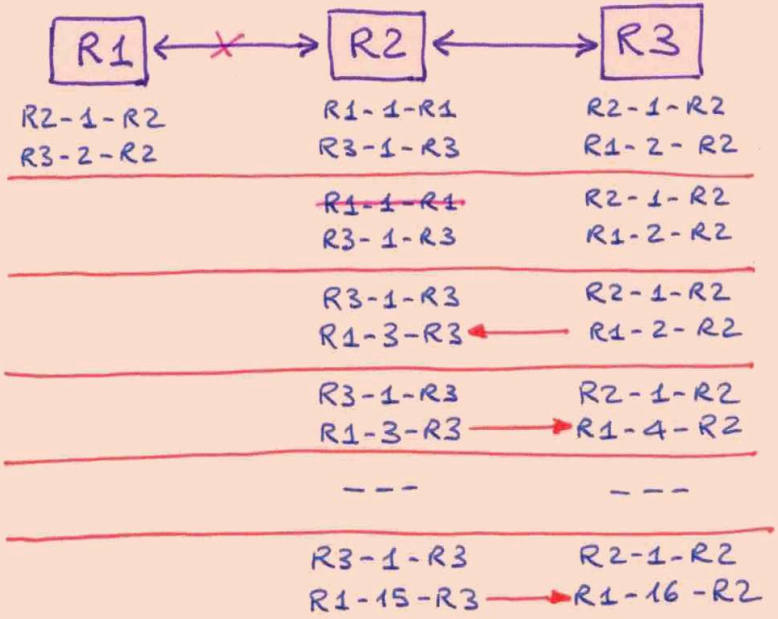
By a frequent exchange of information over the network, every router identifies the other routers that it can reach with 1 hop. This information is subsequently shared with all its neighbors, that can update accordingly their routing tables.

Records in routing tables have a limited life.

The maximum number of hops is generally fixed to 15.

RIP: Counting to infinity

What if R1 crashes?



16! Infinity...

It takes actually too much for the network to "realise" that the router R1 is actually not reachable.

How to better evaluate the path length ?

The physical layer uses different kinds of media for establishing the communications. The speed of the communication depends on the medium and on its physical properties.

We use as a metric the "bit rate".

The actual distance value is derived from the bit rate.

bit rate	distance value	
100 Mbps	1	(reference)
10 Mbps	10	
1.544 Mbps	65	
64 kbps	1562	
9.6 kbps	10416	

distance value = reference bit rate / actual bit rate.

Open Shortest Path First: the OSPF protocol

The advantages of this protocol w.r.t. RIP:

- the distance value between two routers reflects the bit rate;
- each router sends its tables to the entire local network (and not only to the neighbors);
- every router collects the connectivity information and runs a shortest path algorithm (generally Dijkstra) to find the best routes;
- when a router sends out information, a message confirming the reception is awaited: when waiting is too long, the information is sent out again;
- when a record in the router table is "dead", then this details is communicated to the network, so that the router can receive the updated information.

The Internet Protocol IP

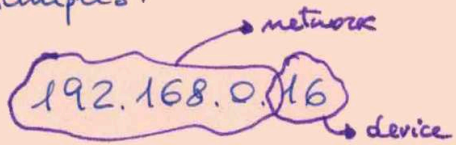
It defines the format of packets (our messages) and provides an addressing system.

As a unique identifier of a computer device over the Internet, this protocol introduces the IP address.

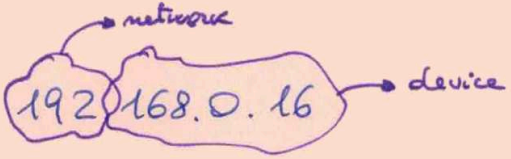
The IP address gives information about the local network where the computer machine is located.

192.168.0.16 IPv4 : 4 bytes (range 0-255 each)

Examples:



Several networks with a few devices.
Mask: 255.255.255.0



A few networks with a lot of devices.
Mask: 255.0.0.0

How to configure the IP addresses of a computer network?

Possibility 1: manually

However, this can be a very hard work, especially for large networks. Moreover, errors may be introduced in the configurations.

Possibility 2: use the Dynamic Host Configuration Protocol (aka DHCP)

Once a machine of the network is set up as DHCP server, it can automatically assign an IP address to every machine in the network.

Moreover, it also takes care of the sub-network mask, as well as of the DNS server.

The DHCP server

The server has a pool of possible IP addresses that it is allowed to assign to the computers in the network.

For example: 192.1.1.1/100

Notice that:

- the address 192.1.1.0 is reserved to the network
- for very large networks, where the pool of addresses may not be sufficient to cover the entire network, IP addresses are only "leased" and not permanently assigned (with some exceptions)
- the server gives the same DNS and gateway addresses to all its machines

→ in our example, the network mask is 255.255.255.0

The DNS server

Every network has its own DNS server.
This server basically works as a "cache memory" for I.P. addresses.
When one of the machines in the network asks to connect to a given website (say, www.google.com), the DNS server checks its cache, and if the website was never visited before, it redirects the requesting machine to the

AUTHORITATIVE NAME SERVER

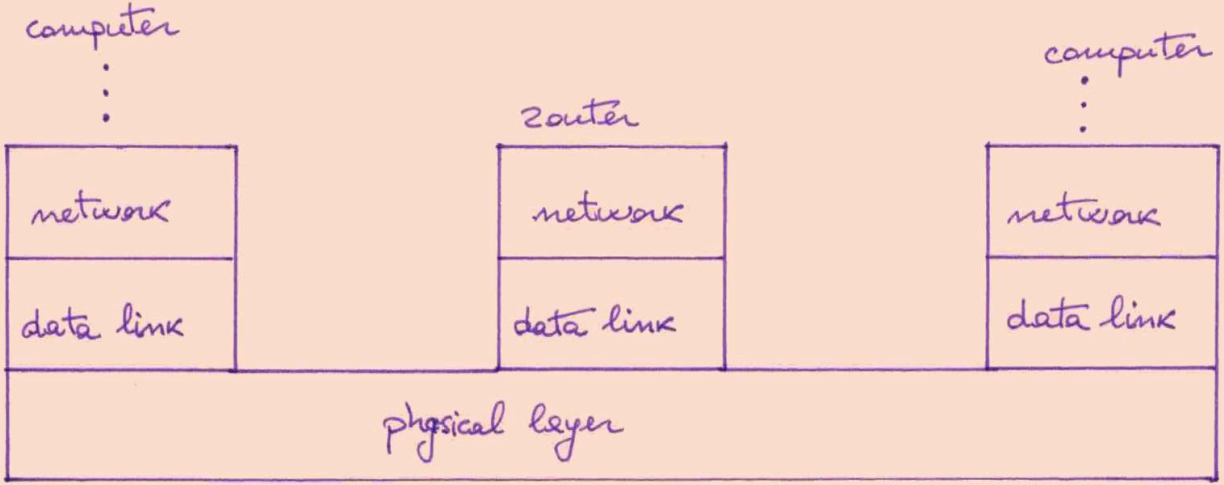
which is a public machine.

By the way, DNS = "Domain Name Server".

The interface with the Network layer

This interface allows the messages to find the best route in the network.

Routers implement layers up to the network layer, they also have a unique address in the network, they have a fundamental role.



The Transport layer

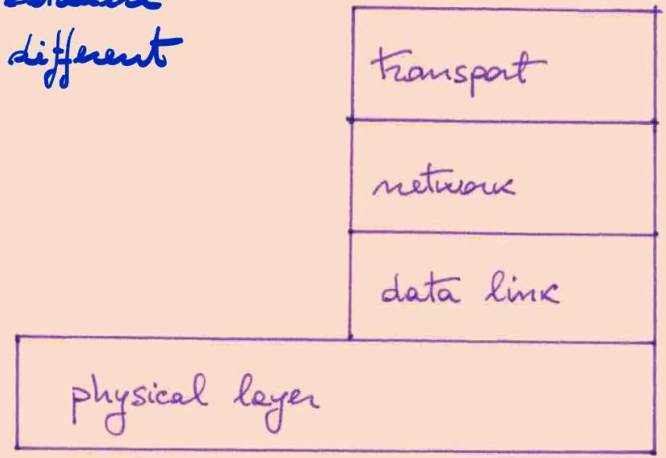
The main role of this layer is to improve the quality of the connection, independently from the different paths that the messages can take over the network.

It introduces the concept of PORT, allowing multiple communications between the same machine pair but for different purposes.

The two main protocols are

UDP TCP

This is a software-based layer!



UDP is

- connectionless: messages between two computers can be sent out (and received) without any preliminary exchange of information
- light: it requires a short header

but

- it doesn't systematically verify possible transmission errors
- it has no congestion control over the network
- it doesn't verify the correct order of the received messages, before passing them to the upper layers

UDP = User Datagram Protocol.

TCP is

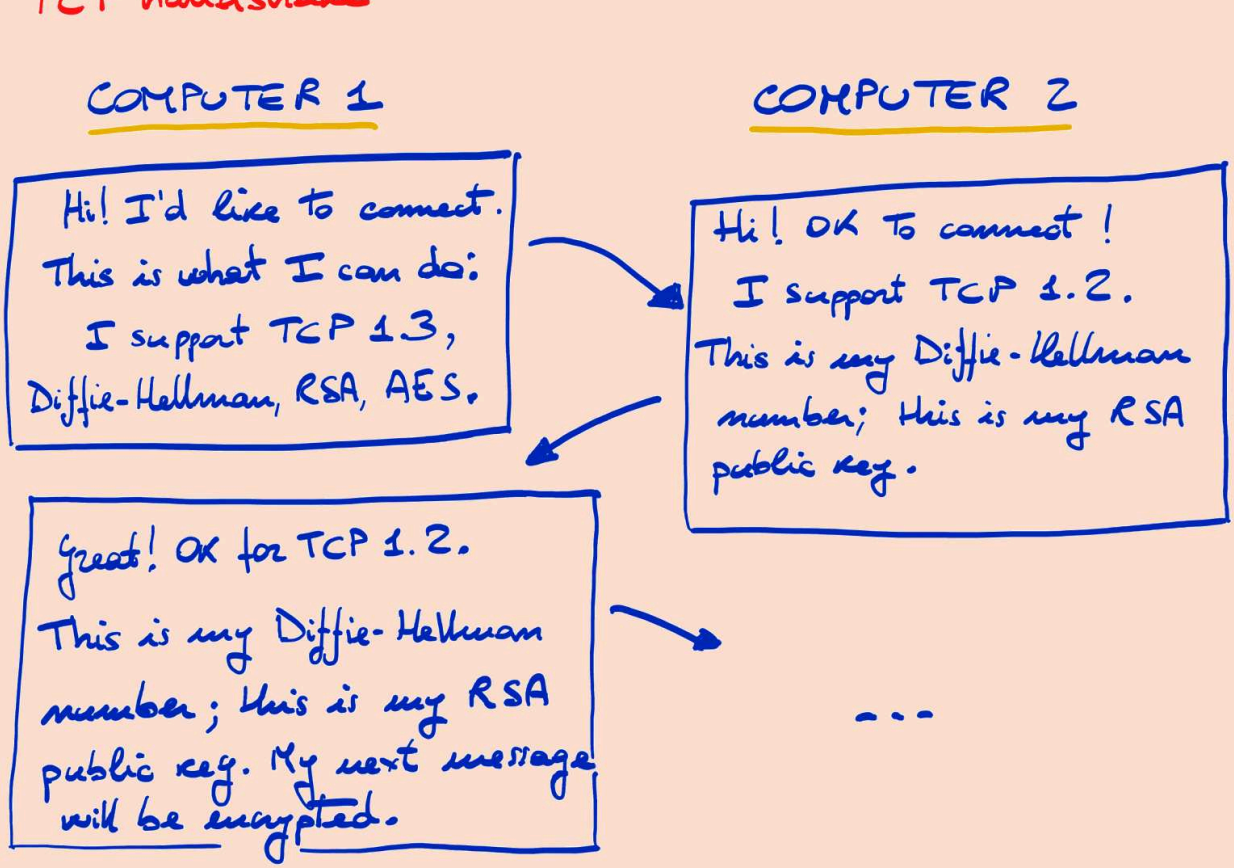
- connection-based: the three-step handshake is required; once the connection established, it is possible for receiving machines to notify the reception of messages, and in absence of such notifications, it is possible for sending machines to retransmit.
- reliable: the checksum verification is applied on the data; a failed verification implies a request of retransmission
- network friendly: it avoids contributing to network congestion
- application friendly: it reorders the received messages before handing them to the upper layers

but:

- it is "heavier"

Transport Communication Protocol.

TCP handshake



TCP has not always used encryption

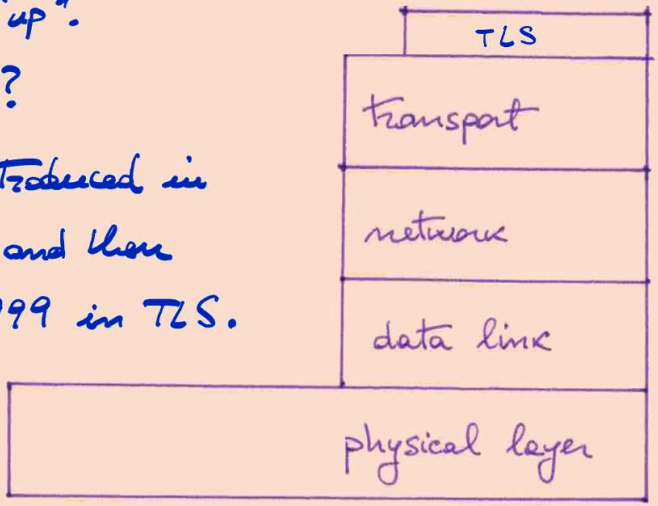
Encryption is managed by the sub-layer named

TLS Transport Layer Security

TLS is positioned "at the borders" with the upper layer, in order to provide encryption for everything that is "up".

Maybe you heard about SSL ?

The secure socket layer was introduced in the Netscape browser in 1995, and then extended and renamed in 1999 in TLS.



The other layers

Session

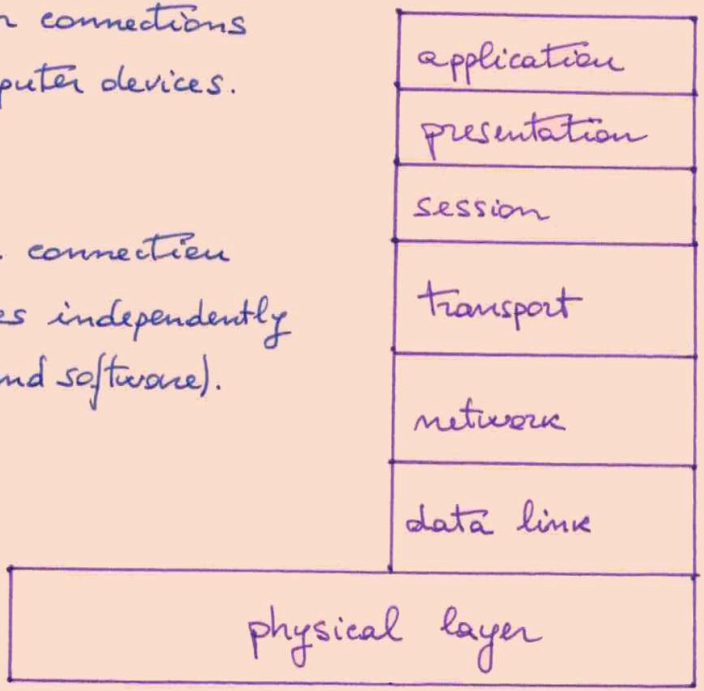
It is in charge to establish connections between "users", not computer devices.

Presentation

It allows for establishing a connection between two computer devises independently from their features (hardware and software).

Application

It provides the interface to the software for performing operations on the network.

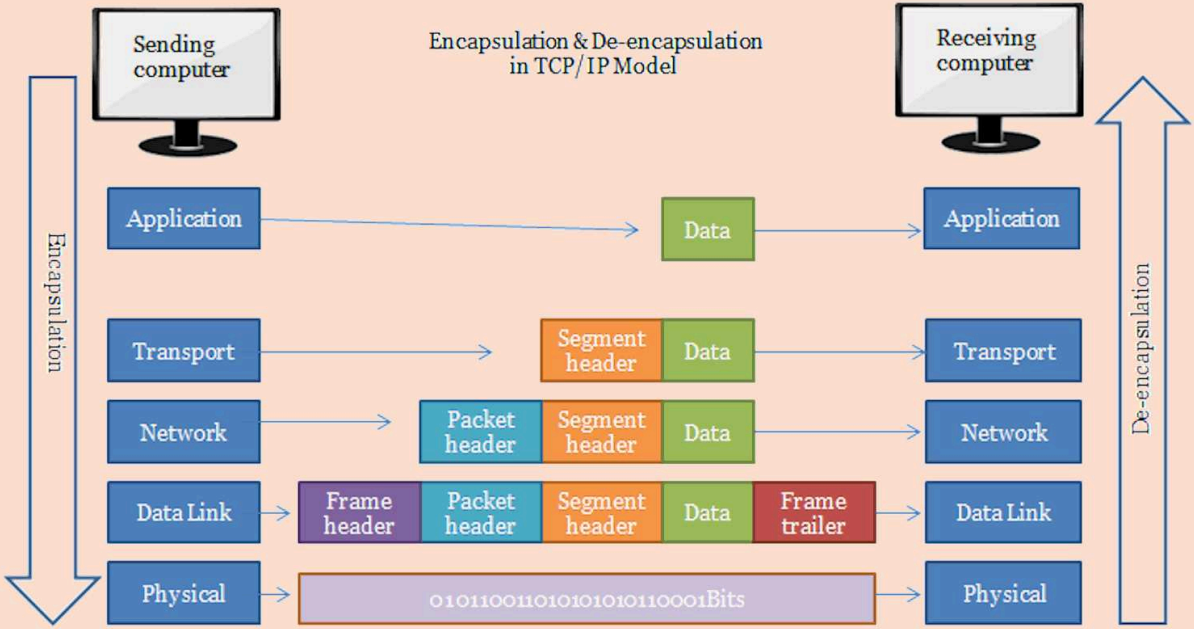


The OSI model

The famous protocols of the application layer

- HTTP(S) hypertext Transfer protocol (secured)
- (S)FTP (secured) file Transfer protocol
- NFS network file system
- POP3 post office protocol (version 3)
- SMTP simple mail Transfer protocol
- IMAP internet message access protocol
- VoIP voice over internet protocol
- ... ...

Data encapsulation



picture taken from a Forum on Stack Overflow.

THE END

Part of these slides has been handcrafted in 2021 after an entire semester where the covid-19 sanitary crises had forced all teachers to teach online...
I really had enough of staring at computer screens.
The other slides have beed added one year later, still handcrafted, but on a tablet device this time.