UNIT 1. INTRODUCTION

UNIT 2. IP NETWORKS

UNIT 3. LOCAL AREA NETWORKS

a. INTRODUCTION

Local networks can be classified by their range:

- Wide Area Network (WAN): have as objective to reach an extensive geographical area and with an unlimited number of stations. Are made to be scalable (e.g., *telephone network*).
- Local Area Network (LAN): they want to interconnect a limited number of stations in a small area, so scalability is not its main objective.

Or by the interconnection strategies:

- Switched network: formed by the interconnection of commutators that direct the information between both nodes that are communicating. Their object is similar to the one from the routers. Their main advantage is scalability due to the easiness to expand the network (just add more commutators). That's why WANs are always switched networks.
- Multi access network: the main handicap of switched networks is that the range escalates alongside its price, which makes them pretty expensive. They are formed with a shared medium so that when the emitting station sends a message, all the possible destinations receive it, but all dismiss it except for the expected receiver. This way we save on commutators, but we do need a protocol (Medium Access Control, *MAC*) able handle and organize all the stations. That's why LANs use multi access networks.

b. IEEE LAN ARCHITECTURE

IEEE 802.2, is a standard which defined Logical Link Control (*LLC*) as the upper portion of the data link layer of the OSI Model.

Layer			Protocol Data Unit	Function
Host Layers	7	Application	Data	Protocols from apps that use the network, (i.e., *http*, *stmp*, *ftp*, *telnet*…)
	6	Presentation		Provides protocols of data presentation.
	5	Session		Manages a session among two apps.
	4	Transport	Segment, Datagram	Establishes a channel for both apps.
Media Layers	3	Network	Packet	Guides the PDUs to the correct destination.
	2	Data Link	Frame	Connection between network level and physical.
	1	Physical	Bit, Symbol	Defines the physical and electric features.

OSI divides the Data Link layer in two sublevels:

- Logical Link Control (*LLC*): defines the interface with the upper layer. The standard has two fields; Destination SAP (*DSAP*) and Source SAP (*SSAP*). SAP identifies which upper level has to receive the content of the frame.
- Medium Access Control (*MAC*): is different for each LAN. Its objective is to regulate the access to the shared medium. The data structure (PDU) of MAC level is the frame and this is what will travel across the physical network.

c. TYPES OF MAC

In order to design MAC protocols two strategies have been used:

- Token passing: the access is regulated by a token. The station that has the token is the one that can transmit and the rest of stations have to keep silent. After the transmission of a frame is completed, the station passes the token.
- Random: the stations try to transmit and if by accident two transmissions happen simultaneously (collision), they have to wait a random amount of time (backoff time) and try again. I.E., Ethernet.

Carrier Sense Multiple Access (CSMA) is a random MAC protocol where stations listen the medium before transmission. When the medium is available the station transmits immediately, if it is occupied, then the station waits until it becomes free. If there is no confirmation, it means that there's been a collision and the frame will be retransmitted after a random backoff time.

d. ETHERNET

Ethernet is a random MAC protocol. Its frames use two different formats: Ethernet II and IEEE-802.3. Both are compatible and can be used simultaneously. Their frame fields are the following.

ETHERNET II (DIX)

Preamble (8B)	Dst (6B)	MAC@	Src (6B)	MAC@	Frame (2B)	Type	Payload (46-1500B)	CRC (4B)

IEEE 802.3

Preamble (8B)	Dst (6B)	MAC@	Src (6B)	MAC@	Frame (2B)	Length	Payload (46-1500B)	CRC (4B)

In order to make both frame formats compatible, the values of the type field of DIX frames are always higher than 1500. This way, when the driver of a station receives a frame with the value of the type field below 1500, he knows that the frame has the IEEE 802.3 format, otherwise it is the DIX format.

To solve some interoperability problems between both protocols, the standard IEEE defined an extension of LLC named Sub-Network Access Protocol (*SNAP*). When used, following the LLC header, another header is added which contains two fields: Organizationally Unique Identifier (*OUI*, 3B) represents the organism that defines the protocols and Type identifies a specific protocol (2B). It allows to encapsulate TCP/IP protocols over IEEE standard (with OUI=0x000000 and Type=RFC 1700).

The MAC protocol used by Ethernet is known as CSMA with Collision Detection (*CSMA/CD*). It is similar to CSMA, but now the station continuously listens the medium while transmitting the frame and stops transmitting when detects a collision. If no collisions are detected during the transmission it is assumed that no collision has occurred and it's not necessary for the receiver station to send a confirmation.

- Transmission: between frames, the medium doesn't receive signals during a time known as Inter Packet Gap (*IPG*), fixed in 12B. So, if a station wants to transmit consecutive frames, it has to wait an IPG after each transmission.
- Collision: when a collision occurs, the station stops transmitting immediately and a *jam signal* (32b that produce an erroneous CRC) is sent. Then the station waits a backoff time and continues transmitting.
- Backoff time is equal to $n \cdot T_{512}$ where T_{512} is the transmission time of 512b (i.e., 51,2 μs at 10Mbps) and n is a random number between 0 and $2^{\min(N,10)}-1$ where N is the number of retransmissions of the same frame ($N \geq 1$).

For example, at 10Mbps if there's a collision the backoff might be equal to 0 or 51,2 μs. If another collision occurs, the backoff might be equal to 0; 51,2; 102,4 or 153,6 μs. If this process is repeated 16 times, the frame is discarded.

The Ethernet standards that use different lines for transmission and reception allow both processes simultaneously. There are multiple modes of operation:

- Full Duplex: when two Ethernet NICs (Network Interface Card) are connected point-to-point, some Ethernet standards allow a full-duplex transmission. NICs deactivate CSMA/CD (no collisions can occur).
- Half Duplex: using CSMA/CD only one NIC can be simultaneously transmitting into the medium.

Ethernet NICs have an auto-negotiation mechanism to detect the full-duplex availability. There are many physical level Ethernet standards. The following are just a few of them.

Speed	Common Name	Informal IEEE Name	Formal IEEE Name	Cable Type, Max Length
10 Mbps	Ethernet	10BASE-T	802.3	Copper, 100m
100 Mbps	Fast Ethernet	100BASE-T	802.3u	Copper, 100m
1000 Mbps	Gigabit Ethernet	1000BASE-LX	802.3z	Fibre, 5000m
1000 Mbps	Gigabit Ethernet	1000BASE-T	802.3ab	Copper, 100m
10 Gbps	10 Gib Ethernet	10GBASE-T	802.3an	Copper, 100m

There are also Ethernet standards with optic fibre, that cover major distances. In the denomination "*XBaseY*", 'X' stands for the transmission speed in Mbps (bitrate), "Base" means that the codification is base band signal (though it can also be Broad (translated band signal)) and 'Y' has multiple meanings (maximum segment distant in hundreds of m, reference to the medium type (T: UTP, F: optic fibre, and others…).

The more standardized and economic one is 10BaseT. In this standard the stations are connected to a repeater (hub), which decodes the signal received in a port and transmits it with a delay of few bits through all the other ports. All standards use UTP or OF (except 10GBaseCX4 that operates only in full-duplex mode).

- Fast Ethernet (1995). 100BaseTX: UTP-cat 5.
- Gigabit Ethernet (1998). 1000BaseT: UTP-cat 5e.
- 10Gigabit Ethernet (2002). Uses optical fibre.

e. ETHERNET SWITCHES

If there are many stations connected simultaneously in a hub, it may be inefficient due to collisions.

With the objective to segment the collision domain of an Ethernet Network in an economic manner (not router) the bridges are created. A bridge is a device with a limited number of ports and each one with its own NIC. The process is as it follows:

The bridge has a MAC table, so that when a frame is received, the bridge knows through which port it must be sent. But before starting the transmission, the frame is stored in the transmission queue of the corresponding port. So, each port of the bridge is a different collision domain. Whenever a frame arrives, the bridge checks whether its source address is in the MAC table, and adds it if it is not found. Then checks the destination address, and if it is unknown, duplicates the frame in all transmission queues of the ports, so that the frame will be transmitted through all ports to make sure that it reaches its destination. The entries of the MAC table have a *time-out* that triggers the deletion of the address when it expires.

And then came the switches, which have the same functionality as the bridges but with more ports and a major capability of communicating frames among the ports. It is capable of commutate frames simultaneously between different ports. Each port is a different collision domain and can have distinct bitrates. They may be full-duplex (if only one host is connected) and there can be ports simultaneously in half or full duplex mode. Bitrate can be increased by aggregating several links, which behave as a single one. Stations can only capture traffic of their collision domain, which increases the security.

Thanks to the segmenting of the collision domain, switches can increase the scalability of the LAN. Once the switches have the MAC tables initialized, they direct the frames so that they cross just the necessary links to get to their destination. But when a switch receives a broadcast frame (their objective is to arrive to all stations of the network and their destination address is FF: FF: FF: FF: FF: FF), it is sent through all ports except the one it came from. Due to this, all Ethernet stations interconnected with level 1-2 devices create a "broadcast domain". The routers do segment the broadcast domain.

Suppose a port with a station that transmits at 100Mbps to another station with a NIC of 10Mbps. The transmission queue of the port configured at 10Mbps will quickly overflow and the switch will start losing frames. That is where flow control appears. It is an element of the switch which consists of adapting the rate at which the switch receives the frames, and the rate at which the switch can send them. There are two techniques:

- Jabber signal (half duplex): the switch sends a signal into the port which needs to be throttled down, such that CSMA see the medium busy.
- Pause frames (full duplex): the switch sends *special pause frames*. These frames have an integer indicating the number of slow-times (512b) that the NICs receiving the frame must be silent.

But when two stations are receiving frames through the switch at different bitrates, the slow link may trigger the flow control and send pause frames towards the server, causing under-utilization of the switch-server link (which has higher bitrates).

If the hub is the bottleneck for all the active ports, the capacity is equally shared between all ports where frames are transmitted. But if one congested port is the bottleneck for all ports sending traffic to it, the port bit rate is equally shared between all ports sending traffic to it.

In a network formed by Ethernet switches, when the MAC tables are initialized, the frames go from switch to switch from the source to the destination. The switches do not admit an arbitrary topology (routers do), so when a broadcast frame is sent, each switch will transmit the frame through all ports, and some of these frames will loop indefinitely, which will saturate the network. So even though loop can sometimes come in handy, they are not allowed. To solve this problem there is the standard STP (*Spanning Tree Protocol*), which builds a loop free topology with optimal paths. The ports that do not belong to the STP tree are blocked and discard all incoming frames, so they are not in the initialization process of the MAC tables.

An Ethernet switch constitutes a broadcast domain. Sometimes is convenient due to efficiency and security motives to have servers and hosts related in different broadcast domains, each one identified by a subnetwork. With Virtual LAN (*VLAN*) we can achieve a logical distribution of the broadcast domains that do not belong to the distribution and physical connection of the commutators (switches).

Each switch port belongs to a different VLAN and all hosts connected to that port belong to the VLAN associated. For every VLAN the switch has a different MAC table. If a broadcast frame is received in a port, the switch will just retransmit it through the rest of the ports belonging to the VLAN. So, in order to go from one VLAN to another it is necessary to go through the router. This allows a greater flexibility of the physical placement of the devices, facilitates the network growth. A different STP tree is built in each VLAN.

If a port belongs to several VLANs (maybe all) it is configured as trunk (connection between two switches), so the traffic sent in one VLAN is also sent to the trunk the VLAN belongs to. A tagging mechanism is used in the trunk to discriminate the traffic from different VLANs. There are two trunking protocols, Inter-Switch Link (*ISL*) and the IEEE-802.1Q standard, which adds 4B between the source address and the Type/Lengh fields. The field Tag Protocol Identifier (*TPID*) has the hex value 0x8100 when the tag has been added to an Ethernet frame, and the field Tag Control Information (*TCI*) contains several fields, the most important is the VLAN ID (12b), which identifies the VLAN.

f. WIRELESS LANs

The wireless LANs have various advantages respecting the wired networks. No expenses in the wiring, flexibility in the deployment of the network (the network can be easily installed/uninstalled), the stations can move freely in the network.

But they also have some handicaps. High frequency modulations are needed in order to make possible the transmission through the area. When the signal is propagated though the space it is attenuated and the usable power of the received signal is really weak. The feebleness of the received signal doesn't help to get rid of interferences and noise.

The IEEE-802.11 protocol, also known as Wireless Fidelity (*WIFI*) is one of the most standardized. It uses the frequency bands Industrial, Scientific and Medical (*ISM*). There are various standards at physical level:

Standard	802.11	802.11b	802.11a	802.11g	802.11n
Bitrate	1, 2 Mbps	up to 11Mpbs	up to 54Mbps	up to 54Mbps	up to 600Mbps
ISM Band	2,4 GHz	2,4 GHz	5 GHz	2,4 GHz	2,4 or 5 GHz

802.11 has two operating modes:

- Infrastructure: all transmissions have to go through a special station known as Access Point (*AP*). AP sends beacons (special signalization) to make known their presence. The stations have to find and associate with an AP in order to access the WLAN.
- Ad-hoc: all stations access the medium the same way (no Aps).

To reduce to the maximum the number of collisions uses Carrier Sense Multiple Access with Collision Avoidance (*CSMA/CA*), which in contrast to CSMA/CD, always waits a random backoff before starting transmitting and *Acks* are needed to detect whether a transmission frame collided.

802.11 addresses are designed to be compatible with ethernet. Use non overlapping ranges with ethernet. The frame may have up to 4 addresses, and their meaning is specified by the bits to-DS and from-DS of the control. The BSSID is always present to identify frames belonging to the BSS.

Generic format of an 802.11 frame

Control (2B)	Duration (2B)	Address 1 (6B)	Address 2 (6B)	Address 3 (6B)	Seq-Ctrl (2B)	Address 4 (6B)	Payload (0-2312B)	CRC (4B)

An important characteristic of WLAN is that uses no ISM regulated frequency band, which allows multiple independent WLANs to be inside the same radius of range. So, in order to avoid receiving unpleasant frames from another WLAN a filtering mechanism is needed. To identify stations that belong to different networks, 802.11 defines the known Basic Service Set (*BSS*), identified by a number of 48b named BSS Identifier (*BSSID*). The frames the carry a different BSSID from the one of the NIC are discarded. When a station still hasn't accessed its network and doesn't known the BSSID, it can access using the BSSID broadcast, which coincides with the address 802.11 broadcast (FF: FF: FF: FF: FF: FF).

If an 802.11 network is constituted by a single BSS, then it becomes an Independent BSS (*IBBS*). A BSS can be in infrastructure mode, where each AP forms a different BSS and the BSSID is the address 802.11 of the AP, or in ad-hoc mode (no AP). If a network has more than one BSS, then it is an Extended Service Set (*ESS*) and the part of the network that allows the interconnection of the different BSS is known as Distribution System (*DS*).

In a network with 802.11 and Ethernet, the uniqueness of the address is guaranteed. This allows 802.11 NICs to communicate with Ethernet NICs in a transparent way. But in order to make this possible, the 802.11 header has 4 address fields, the meaning of which depend on the scenario.

Scenario	Usage	to-DS	from-DS	Address 1	Address 2	Address 3	Address 4
STA > STA	Ad-hoc	0	0	DA	SA	BSSID	-
STA > AP	Infrastructure	1	0	BSSID	SA	DA	-
AP > STA	Infrastructure	0	1	DA	BSSID	SA	-
AP > AP	WDS	1	1	RA	TA	DA	SA

In Ad-hoc mode only the STA > STA addressing mode is used. In Infrastructure mode both STA > AP and AP > STA addressing modes are used. AP > AP is only used when the DS is also wireless, this scenario is known as Wireless Distribution System (*WDS*).

UNIT 4. TRANSPORT PROTOCOLS

g. BASIC ARQ PROTOCOLS

Their objective is to transmit the information to the destination without errors, nor duplications and in the same order they are sent. The entity that sends the package is named "Primary" and the one how receives it "Secondary". In order for primary to know if the retransmission is successful, secondary must send back acknowledgments. The primary has a "transmission buffer" that stores all the information that has been sent but no confirmed. As long as there's space in the buffer, primary will continue sending data to the secondary. In the case of an error, primary will retransmit the failed package because is stored in the buffer. The secondary has a "reception buffer" where received information is stored until the upper level reads it.

To be able to link the information messages and their confirmations we need to use "sequence numbers".

There are three ARQ basic algorisms:

- Stop and wait: transmit a PDU (Protocol Data Unit) and wait until its confirmation before sending another. The complete process is as it follows: when the sender is ready, writing from the upper layer is allowed, I_k (Information PDU k) is built and sent down to data-link layer to be transmitted. When I_k completely arrives to the receiver, it is read by the upper layer, A_k (Acknowledge PDU k) is generated and sent down to data-link layer and transmission starts. When A_k completely arrives to the sender, transmission is completed. Each time that the sender transmits a PDU, a retransmission timeout is started. If the information PDU do not arrives, or arrives with errors, no *ack* is sent, so the timer will run out and the sender will retransmit the PDU. The sequence numbers are stored in the header that the protocol adds. Thanks to this we avoid duplicated PDUs. This protocol can be really inefficient if the propagation time (t_p) is not so much smaller than the PDU transmission time (t_t).

The continuous transmission protocols aim to allow a high efficiency by not limiting the number of PDUs that can be sent continuously.

- Go back N: if the sender receives an error, it will go back and start transmitting from that PDU. A_k confirms all information PDUs with sequence numbers $< k$. If the secondary receives an I_k PDU with errors or out of order, it will stop sending acknowledgments until it receives the pending PDU and will discard all transmission with sequence number $\neq k$. When the primary timer runs out, it will retransmit the I_k PDU, followed by I_{k+1} …
- Selective retransmit: the secondary never discards any PDUs that arrive correctly (even if they are out of order). This improves the efficiency respecting the "Go back N" algorism. Out of order received

PDUs will have to be stored and reordered. A_k confirms all information PDUs with sequence numbers < k. If the secondary receives an I_k PDU with errors or out of order, it will stop sending acknowledgments until it receives the pending PDU and will store all transmission with sequence number ≠ k. When the primary timer runs out, it will only retransmit those for which the timer ran out. When the secondary receives a transmission, it sends back an acknowledgment confirming all previous in-order PDUs received.

ARQ are also used for flow control, which consists on avoiding the sender to transmit at higher PDU rate than can be consumed by the receiver.

- Stop and wait: if the receiver is slower, *acks* are delayed and the sender reduces the throughput (effective speed). Stop and wait is a window protocol with transmission window equal to one PDU.
- Continuous transmission protocols: a transmission window is used. The window is the maximum number of non-*ack* PDUs that can be transmitted. If the transmission window is exhausted, the sender stales.

Transmission window allows dimensioning the transmission buffer, and the receiver buffer for selective retransmission (no more the transmission window PDUs need to be stored). The optimal window is the minimum window that allows the maximum throughput.

h. INTERNET'S TRANSPORT LAYER

Transport layer implements a point-to-point protocol between both hosts that are communicating, that multiplexes the information transmitted by the processes. Transport layer offers a communication channel between applications. Transport layer access points (applications) are identified by a 16b port number. TCP/UDP use the client/server paradigm, where the "server" (usually a daemon in a UNIX machine) waits the requests from the "clients". That's way the server listens the requests addressed to a well-known port. These ports have a value in the interval the goes from 0 to 1023, which is assigned by IANA. The client is always who starts the connection towards the server and has a port assigned by the O.S. in the interval [1024, ..., 2^{16}-1], which is fleeting because only identifies the process of the client while there's a connection.

i. UPD PROTOCOL

UDP (*User Datagram Protocol*) offers a non-reliable datagram service. Basically, what it does is add a header to the information that receives from the application in order to build a "UDP datagram" and passes it to IP level to be transmitted. Is not a connection-oriented service (*connectionless*) and has no error recovery protocol, so if a datagram is lost, UDP won't retransmit it. UDP doesn't have a transmission buffer, which means that each application write operation generates a UDP datagram. UDP is typically used in applications where short messages are exchanged (e.g., *DHCP*, *DNS*, *RIP*...) and real time applications (e.g., *Voice over IP*, *videoconferencing*) that don't tolerate large delay variations, which would occur using an ARQ.

If a datagram is late, it will be discarded because it is useless to the application.

UDP DATAGRAM HEADER

Src Port	Dst Port	Length	Checksum

UDP PSEUDO- HEADER

Src @	Dst @	Zero	Protocol	UDP Length

Fixed size of 8 bytes. The checksum is computed using the header, a pseudo-header and the payload. A drawback is that because of the pseudo-header, the UDP checksum needs to be updated if PAT is used.

j. TCP PROTOCOL

It's the protocol of transport layer used in Internet due to its reliability transmitting information. It is an end-to-end protocol, ARQ, connection oriented and has the following objectives:

- Error recovery to guarantee a reliable transmission.
- Flow control to block the primary to send segments at a higher speed than what the secondary can process. In order to do so, TCP uses an advertised window that will always be sent in the TCP header. When TCP receives a segment, the value of the window is stored in a variable named *awnd*. The primary cannot send more bytes without confirmation than the ones indicated by *awnd*. This way,

there will always be enough space to store the bytes sent in the receiver buffer. If this buffer is full, the secondary will send an advertised window equal to zero in order for the primary to get blocked until the received sends back a larger window.

- Congestion control to avoid the primary from sending segments at a higher speed of what the network can handle. Otherwise, there would be segments that could be lost because of the bottleneck. Losses due to congestion indicate TCP that the speed used is above the one from the bottleneck. TCP uses a congestion window and *cwnd* will keep its value. This variable is incremented if no losses are detected, and decremented otherwise.
- Segments of optimal size: unlike UDP, TCP gets the bytes from the applications to generate segments of optimal size. A variable named Maximum Segment Size (*MSS*) stores the size of whatever size is considered optimal regarding the payload (usually the optimal size is largest one that doesn't implies using fragmentation). In order to be prepared to the worst-case scenario, TCP always uses a transmission window (*wnd*) equal to min {awnd, cwnd}.

TCP connections can be classified as:

- Bulk transfer: during the transmission the applications always has data ready to be sent. In this case the transmission buffer is always full and TCP sends maximum size segments. The transmission window restricts the effective speed of the connection. (e.g., web, ftp, mail…).
- Interactive: the information is sent in messages of few bytes and discontinuous. No restrictions are needed because of the low amount of data that is being sent. (e.g., telnet, ssh…).

In bulk connections sending an *ack* every data segment can unnecessarily send too many small segments, so delayed *acks* are used. Consists of sending 1 *ack* each 2 MSS segments, or 200ms. *Acks* are always sent I case if receiving out of order segments.

In Interactive connections the user is who interacts with the remote host, and if a lot of time passes between key presses, a segment and its confirmation would be generated for each press. To solve this problem, the Nagle Algorithm is used. Each time new data gets to the transmission buffer and the window allows sending the next segment, datagrams will only be transmitted if there are enough bytes to send a maximum size segment or there are no pending of confirmation bytes.

TCP HEADER

Src Port	Dst Port	Seq. Num.	*Ack* Num.	Header length	TCP Flags	Advertised window
Checksum	Urgent Pnt.	Options	Padding			

TCP PSEUDO-HEADER

Src @	Dst @	Zero	Protocol	TCP length

Size of the header in words of 32 bits. Minimum size is 20B, maximum size is 15·4 = 60B. Like UDP, the checksum is computed using the header, a pseudo-header, the payload and needs to be updated if PAT is used. The possible flags are these:

- *URG* (urgent): the urgent pointer is used. It points to the first urgent byte. Rarely used.
- *ACK* (acknowledgment): the *ack* field is used. Always set except for the first fragment sent by the client.
- *PSH* (push): the sender indicated to "push" all buffered data to the receiving application.
- *RST* (reset): abort the connection.
- *SYN* (synchronize): used in the connection setup.
- *FIN* (finalize): used in the connection termination.

The field options allow adding options to the header. These are some of them:

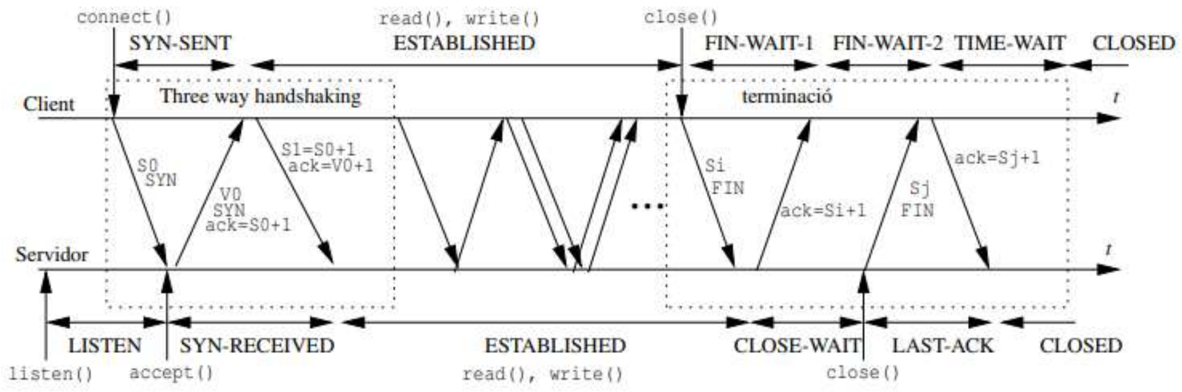
- Maximum segment size (MSS): used during the connection setup to initialize the MSS. In IPv4 it is set to MTU-40.
- Window scale factor: used in the connection setup to indicate that the value of the adverted window has to be multiplied by this factor ($2^{\text{Window scale factor}}$). Allows using awnd larger than 2^{16} bytes.
- Timestamp: used to compute the Round-Trip Time (*RTT*). Is a 10B option, with the timestamp clock of the TCP sender, and an echo of the timestamp of the TCP segment being *ack*.
- *SACK*: in case of errors, indicate blocks of consecutive correctly received segments for Selective retransmit.

The sequence number of the header identifies the first byte of data of the segment. Initially, the window is equal to MSS and is incremented each time that an *ack* confirms new data. The *acks* have a value equal to the sequence number of the segment that is being confirmed. So *ack's* value is the sequence number of the following data segment awaiting to be received by the secondary. When the primary receives the *ack* confirms all bytes with sequence numbers < than the one received and removes them from the transmission buffer.

Establishment of a TCP connection is known as "*Three-way-handshaking*" and consists of the exchange of three segments that only carry the TCP header:

- *SYN*: first segment, always sent by the client. *Well Known* port (<1024) as destination and a short-lived port (≥1024) as source. Has the *SYN* flag activated and *ACK* deactivated (nothing is to be confirmed). Carries the initial sequence number (random number of 32 bits).
- *SYN* + *ack*: second segment, sent by the server. Also has the *SYN* flag activated and carries the initial sequence number to identify the information bytes sent by the server. Even though *SYN* segments don't have information bytes, they carry a sequence number. The *ack* of this segment is equal to the initial sequence number plus 1.
- *ack*: the client confirms the reception of *SYN* + *ack* transmitting the confirmation with the server's initial sequence number plus 1.

The connection termination happens when exchange of 3/4 segments occur. *FIN/ack* in a direction and *FIN/ack* in the opposite direction. *FIN* segment is sent when the application calls *close()*. Just like *SYN*, *FIN* has a sequence number. But *FIN* can also carry data. The first segment can be sent either by the client or the server.



In order to fulfill congestion control's objective, the *congestion window*'s (*cwnd*) value is reduced rapidly when there's congestion and incremented slowly otherwise. The basic Congestion Control algorithm is named Slow Start / Congestion Avoidance (*SS/CA*).

SS/CA uses a the following variables; *cnwd* (congestion window), *snd_una* (*unacknowledgment*, first not confirmed segment), *ssthresh* (*slow start threshold*, threshold between the *slow start* and *congestion avoidance* phases). During SS *cwnd* is rapidly increased to the "operational point". Duraing CA *cwnd* is slowly increased looking for more available brandwidth.

```
Initialization:
        cwnd = MSS;
        ssthresh = infinite;
Each time an ack confirming new data is received:
        if (cwnd < ssthresh) cwnd += MSS;                 /* SS */
        else cwnd += MSS * MSS / cwnd;                    /* CA */
When there is a time-out:
        Retransmit snd_una;
        ssthresh = max (min (awnd, cwnd) / 2, 2 MSS);
        cwnd = MSS;
```

TCP sends the entire window, W (in several segments). The segments accumulate in the queues of the interfaces where there are bottlenecks. Steady state : the TCP connection started time ago. In general, we can assume that, on the average, is fulfilled vef = W / RTT. If there are no losses, W will be awdn, otherwise W follows a "saw tooth".

TCP has a timer to control retransmissions (*Retransmission TimeOut*, *RTO*). This timer is active when there are pending *akcs*. When *RTO* is active, it is continously decreased, and a retransmission occurs when *RTO* reaches zero. Each time an *ack* confirming new data arrieves *RTO* is computed and restarted if there are pending *acks*, or stopped otherwise.

Whenever *RTO* is computed, the TCP sender measures the RTT mean (srtt) and variance (rttvar). The retransmission time-out is given by *RTO* = srtt + 4·rttvar. *RTO* is duplicated each retransmitted segment. RTT measurements are calculated using "slow-timer tics" or the TCP timestamp option.

UNIT 5. NETWORK APPLICATIONS

k. DNS

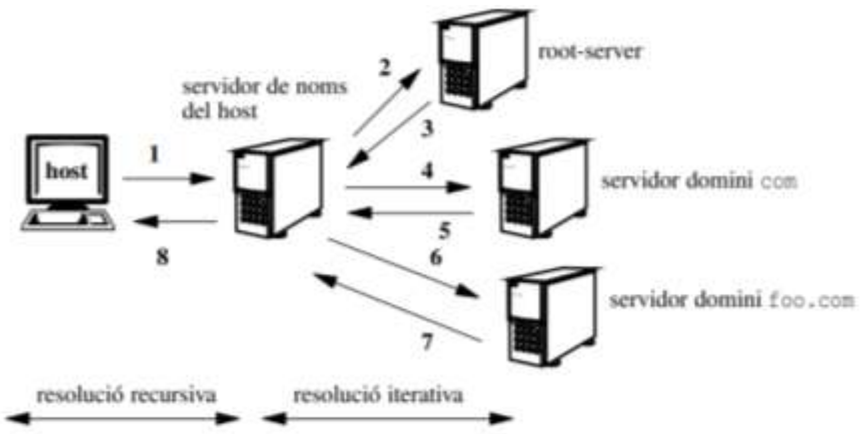
DNS (*Domain Name System*) allows users to use names instead of IP addresses (e.g. rogent.ac.upc.edu instead of 147.83...). These names consist of a node-name (rogent.) and a domain-name (ac.upc.edu). DNS consists of a worldwide distributed data base, where its entries are referred to as Resource Records (*RR*). The information associated with a name is composed of 1 or more RRs.

Access to DNS data base is done using Name Servers (*NS*), which may hold permanent and cached RRs (removed after a timeout). Each subdomain has an authority which consists of a primary and backup NSs. An authority has the complete information of a zone (subdomain); names and addresses of all nodes within the zone and of all subzone authorities).

Uses the client-server paradigm with TCP/UDP transport level and well-known port 53.

In order for a host to access a data base, it must first resolve the address:

- The host sends the name that is to be resolved to the server.
- The server transmits the request to a root-server, that will return the name server address of the TLD.
- Then the server addresses the domain server, which returns the name-server address of the second level domain.
- The server now accesses the server of the domain and gets back the requested address.



- Finally, the server returns the address to the host.

All DNS messages have the same format:

Header (12B)	*Question (var)*	*Answer (var)*	*Authority (var)*	*Additional (var)*

The header has the following format:

Identification	*Flags*	*#Questions*	*#Answers*	*#Authorities*	*#Additional*

l. Email

Uses transport layer, TCP and well-known port 25. Its application layer protocol is known as Simple Mail Transfer Protocol (*SMTP*). Retrieval protocols are *IMAP*, *POP* and *HTTP*.

STMP is designed as a simple and text-based protocol.

- Client basic commands: *HELO* (identify SMTP client), *MAIL FROM* (identify sender mailbox), *RCPT TO* (identify recipient mailbox), *DATA* (mail message), *QUIT* (close transaction).
- Server replies: Three digit number (identifies what state the client to enter next), and a message.

Multipurpose Internet Mail Extensions (*MIME*) is used in mail, web, etc…

There're multiple retrieval protocols:

- Post Office Protocol (*POP*): listens on well-known port 110, user normally deleted messages upon retrieval.
- Internet Message Access Protocol (*IMAP*): listens on well-known port 143, messages remain on the server until deleted by the user. Provides commands to create folders, move messages…
- Web based Email (*HTTP*): a web server handles users mailboxes. User agent is a web browser, thus, using HTTP to send and retrieve email messages.

m. Web

Uniform Resource Identifier (*URI*) is a generic syntex to identify a resource.

Uniform Resource Locator (*URL*) is a subset of URIs identifying the location of a resource in the Internat. URL's general syntex is: *scheme://username:password@domain:port/path?query_string#fragment_id*, *scheme* (purpose and the syntex of the remaining part (http. file…), domain (name or IP address that gives the destination location), *query_string* (contains data to be passed to the server), *fragment_id* (specifies a position in the html page).

There are two types of connections:

- Non persistent: default in HTTP/1.0, the server closes the TCP connection after every object (e.g. for an html page with 10 jpeg images, 11 TCP connections are sequentially opened).
- Persistent: default in HTTP/1.1, the server maintains the TCP connections opened until an inactivity time. All 11 objects would be sent over the same TCP connection.
- Persistent connections with pipelining: supported only in HTTP/1.1, the client issues new requests as soon as it encounters new references, even if the objects have been not completely downloaded.

There are multiple HTTP messages:

- GET: most used, requests an object.
- POST: most used, requests an object qualified by the data in the body (form fields of the HTML). Uses MIME types: application/octet-stream to send raw binary data, and application/x-www-form-urlencoded to send name-value pairs.
- HEAD: safe and mandatory, the server returns only the header.

- OPTIONS: requests communication options.
- PUT: stores an entity.
- PATCH: modifies an existing resource.
- DELETE: deletes an entity.
- TRACE: final recipient echoes the received message back.
- CONNECT: used with a proxy.

Caching: the client stores downloaded pages in a local cache. Conditional get requests are used to download pages if necessary.

Proxy servers act as an intermediary for requests from clients. Adds security (the proxy may reject the access to unauthorized servers), logs, caching, saves public IP addresses (only the proxy may have access to the Internet).

n. HTML

Hyper-Text Markup Language (*HTML*) design mail goal was displaying formated text documents with hyperlinks in web browsers. Its features are the next: Hyperlinks (click on a link and jump to another document), Forms (the document accept the user inputs that are sent to the server), Scripting (allows adding programs, the program is executed on the client's machine when the document is loaded), Cascading Style Sheets (*CSS*, allows describing the physical layout in a separate document.

o. Charsets

Interaction between agents in different languages and cultures implies multiple alphabets and characters sets. So, characters are encoded following several conventions: repertoire (a set of characters), code (correspondence between repertoire and natural numbers), encoding (method to convert code numbers into a sequence of octets).

Universal Transformation Format (*UTF*) helps to encode characters. There is UTF-8 (1-4 8bit code units, most common), UTF-16 (1-2 16bit code units), UTF-32 (fixed-length 32bit code units).