1. A protocol defines what / how / when data is communicated
2. The **internet API** is a set of rules that a sending program must follow so that Internet can deliver the data to the destination
3. Networks can be classified on the types of transmission as <u>circuit</u> switching and <u>packet</u> switching
4. **OSI** = open system interconnection
5. OSI layers: application, presentation, session, transport, network, data link, physical
6. TCP/IP layers:
    a. Application = application + presentation + session from OSI
    b. Transport = transport
    c. Internet = network
    d. Link = data link + physical
7. Services of the data link layer: framing and link access, flow control, error correction
8. O placa de retea poate avea mai multe adrese IP; un calculator poate avea mai multe placi de retea si mai multe IP-uri; placa de retea poate fi si externa; pentru conectarea unui calculator la o retea LAN se foloseste placa de retea
9. MAC addresses - physically embedded into the network card, guaranteed to be unique; made of 6 bytes; **MAC** = media access control; in the data link layer; it can be changed
10. Broadcast MAC address = FF.FF.FF.FF.FF.FF = logical address used to identify all the computers within a network
11. A switch keeps a record of the MAC addresses of all the devices connected to it
12. A computer can have multiple IP addresses
13. **127.0.0.1 = localhost** cannot be a network address nor a broadcast addr; it can be set as default gateway but not as DNS
14. 0.0.0.0 is a valid netmask
15. To find the MAC address on windows - *ipconfig /all*; to check if the data link layer works(?) - *arp /a*
16. The optical fiber cable theoretically has unlimited bandwidth; but there is a limit to how much the end devices can send and receive + the ISP contract limits the bandwidth
17. The <u>bandwidth</u> is the physical property of the transmission medium, **throughput** represents the amount of data which we transmit over a quantity of time
18. Optical fiber -> light waves; wireless -> electro-magnetic waves

19. Routing is done at the network level, is based on the destination IP - **there is no routing based on MAC addresses**
20. IPv6 = 16 bytes, IPv4 = 4 bytes
21. In **class-full addressing** - 5 addr classes
    a. **A** - network part = 1 byte, host = 3 bytes; 1.0.0.0 -> 127.255.255.255; 2^7? networks (most significant bit is fixed => only 7 left)
    b. **B** - network = 2 bytes, host = 2 bytes; 128.0.0.0 -> 191.255.255.255; 2^14 networks (first 2 bits are fixed..)
    c. **C** - network = 3 bytes, host = 1 bytes; 192.0.0.0 -> 223.255.255.255; 2^21 networks (first 3 bits..)
    d. D; 224.0.0.0 -> 239.255.255.255; used for multicast
    e. E; 240.0.0.0 -> 255.255.255.255; never used
        i. On some implementations, 255.255.255.255 is not part of class E, it's the *universal broadcast*
    f. Natural mask = masks of A, B, C
22. Class-full addressing => size of the largest routing table = 2^7 + 2^14 + 2^21
23. **CIDR** = Classless InterDomain Routing
    a. Network part (= *prefix*) + host part, but the sizes are no longer multiples of 8bits
    b. Host part = "/x", x = number of bits in the host part
        i. /24 ⇔ class C
        ii. /16 ⇔ class B
        iii. /8 ⇔ class A
    c. The number of hosts is always a power of 2 (this includes the network addr + broadcast)
    d. The beginning host address in each block must be divisible by the no. of hosts in that block; ex. a block of 16 addresses cannot start at a.b.c.36, but can start at a.b.c.64, or 48 etc.
24. A network addr is correct if by ANDing it with the mask, we obtain the net addr; conversely, if we AND a random IP addr with its mask, we get its network addr
25. To get the broadcast addr from the network -> OR it with the negated mask
26. **Supernetting** = multiple routing entries become a single one (so the opposite of subnetting)
    a. Ex: a.b.c.0 / 30 and a.b.c.4 / 30 can be seen as a.b.c.0 / 29
27. A router has as many interfaces as the number of networks it connects

28. In a routing table, if the gateway = 0 => the packet doesn't need to be sent to another network, its destination network can be directly accessed by the router
    a. If the destination and the mask = 0 => all addresses that don't match any other entries are "caught" here
29. Routing table can also be named forwarding table; it contains interface, netmask, destination, address, gateway
30. All the hosts from the same network can physically reach each other without an intervening router
31. **Metric** = the no. of routers that have to be passed in order to reach the destination; 1 actually means that no routers are passed (0 has another meaning)
32. **Private networks**:
    a. 10.0.0.0 / 8 (10.0.0.0 -> 10.255.255.255)
    b. 172.16.0.0 / 12 (172.16.0.0 -> 172.31.255.255)
    c. 192.168.0.0 / 16 (192.168.0.0 -> 192.168.255.255)
33. Proxy server = intermediary for requests from clients seeking resources from other clients
34. **DHCP** = dynamic host configuration protocol
35. A DHCP server can relay IP addr to clients on a network segment separated from the server's location when the router that separates them acts as a relay agent
36. It's possible to have +1 DHCP server on the same subnet if each has its own, disjoint pool of addresses
37. DHCP uses UDP at the transport layer
38. **CLI** = command line interface
39. **SCTP** = Stream Control Transmission Protocol; in the transport layer
40. **DSL** = digital subscriber line; used to give access to internet through telephone lines
41. **Crossover cable**: connect switch -> switch, switch -> hub, hub -> hub, router -> router, router -> pc, pc -> pc
42. **Straight through**: connect router -> switch, switch -> pc / server, hub -> pc / server
    a. Obviously the reverse work as well
43. The switch sends a packet specifically to an endpoint (or more), the hub broadcasts the message to all the network
44. A switch can transport UDP / IP / TCP packets
45. A hub does **not** understand a MAC address; a switch does
46. **Big endian** = most significant byte first; little endian = least …
    a. The network standard is big endian, that's why we use ntoh and hton

47. 2 PCs might not be able to ping each other if both firewalls are enabled
48. Gateways are used for providing connectivity between +2 network segments
49. A computer cannot have 2 gateways
50. **RIP** = routing information protocol
51. RIPv1 doesn't support classless routing protocols (but v2 does); it has the same timers as v2
52. Maximum metric of RIPv2 = 15; any further is considered unreachable
53. Float uses 4 bytes in 32bit systems; 8 in 64bit systems
54. **SSH** = secure shell; used for remote connection to a terminal / command line; in the application layer
55. **P2P** = peer to peer
56. Exchange units at each layer:
    a. App layer - data structure
    b. TCP - segment, UDP - datagram
    c. IP - datagram / packet
    d. MAC layer - frame
57. IP protocol = best effort; doesn't guarantee successful delivery, unicity, data integrity
58. **IP Datagram header = 20 bytes**
    a. Version - 4 bits, usually 4 / 6, if it's ipv4, ipv6
    b. Header length - 4 bits; it counts how many 32-bit entities there are
    c. Type of service - 8 bits; type of content (file, voice, video..), related to priorities
    d. Length - 16 bits; of the entire datagram in bytes => **max IP datagram size = 64kb**
    e. 16-bit identifier; given to each datagram that leaves the host
    f. Flags - 3 bits
        i. DF = "don't fragment"; if it's set to 1 and a packet doesn't fit => the packet is not sent, the sender is notified
        ii. MF = "more fragments", set to 1 whenever a packet is split
    g. 13-bit fragment offset; if a packet doesn't fit on a connection, it will be fragmented and it won't be reassembled until it reaches the destination
        i. The split packets all have the same 16bit ID
        ii. Beware of the 20 byte header size
    h. **TTL = time to live** - 8 bits; = nr. of routers that the datagram can pass before it's discarded; decremented when passing through a router, when it reaches 0 => dg is discarded and a signal is given (*TTL expired*) to the host of this dg

i. Upper layer - 8 bits; which protocols are transported inside this dg (ex. TCP = 6, UDP = 17)
j. **Header internet checksum** - 16 bits; = the 16bit one's complement of the one's complement sum of all 16bit words in the header; while computing this, we consider the value of the checksum inside the header to be 0
k. Source IP - 32 bits
l. Destination IP - 32 bits

59. **MTU** = maximum transfer unit (through a connection)
60. **ARP** = address resolution protocol; basically determines the destination MAC address given its IP; it's not used for determining the IP when the MAC is known; uses broadcast
61. **NAT** = network address translation
    a. About 64k simultaneous connections for TCP and 64k for UDP with a single LAN address (bc the port-number field is 16bits)
    b. The outside world only sees 1 IP addr
    c. **Port forwarding** is an application of it
    d. Increases flexibility when connecting to the internet
    e. Causes loss of end-to-end IP traceability
    f. Reduces address overlap occurrence
62. **ICMP** = internet control messaging protocol
    a. Signaling protocol - error reporting, pings
        i. **Dest network unreachable** - the router cannot route to a network, it has no idea about it
        ii. **Dest host unreachable** - the (final) destination cannot be reached, maybe it's protected by firewall or sth
        iii. **Dest protocol unreachable** - an IP connection can be established, but not a TCP / UDP (perhaps a firewall again)
        iv. **Bad ip header** - checksum doesn't correspond
    b. its data is mostly useless, it's there to ensure that there's been no modifications to the packet in transit
    c. Encapsulated within IP datagrams
63. **Traceroute**
    a. is a diagnostic tool for displaying the path and measuring transit delays of packets across an IP network
    b. It first sends a packet (usually ICMP) with TTL = 1 => we get the first router, TTL = 2 => second … and so on
64. **UDP = user datagram protocol**
    a. Process to process comunication

b. Header = 8 bytes (just for it, there are also 20 bytes from IP, and others from the app layer)
    i. Source port + dest port (16 bits each)
    ii. Length = 16 bits, of the entire datagram
    iii. Checksum = 16 bits, computed over the header + data of UDP + IP
c. The datagrams' integrity is only checked when reaching the (final) destination
d. No congestion control (as opposed to TCP), so it can overflow
e. Datagram delivery is not guaranteed
f. Uses the socket **SOCK_DGRAM**

65. **TCP = transmission control protocol**
    a. Ordered data transfer, retransmission of lost packets => error-free
    b. It has flow control (as opposed to UDP), so no party can overflow the other; traffic is controlled by the OS
    c. It writes to a **stream** of bytes, while UDP writes **packets**
    d. Uses the socket **SOCK_STREAM**
    e. Header = 20 bytes (+ the 20 from IP and others from the app layer)
        i. Source port + dest port (16 bits each)
        ii. Sequence number = 32bits, counts the amount of bytes exchanged over that connection
        iii. Acknowledgement number = 32bits, the index of the next expected byte
        iv. Header length = 4bits, it counts how many 32bit entities are there; also called **data offset**
        v. Flags = 6 bits
            1. ACK; 1 if the acknowledgement number is ok
            2. SYN - synchronize when creating the connection
            3. FIN - final, when closing a connection
        vi. Window size = 16 bits, for flow control (how much space left in the buffer of the receiver)
        vii. Checksum = 16bits, same algorithm as IP, also 'uses' header + data from IP and TCP
        viii. Urgent pointer = 16bits, not frequently used
    f. When initialised, a TCP connection needs a **state**, at the kernel level for both sender / receiver
        i. Starting sequence number (sent)
        ii. Received sequence number
        iii. 2 buffers (queues), one for sending, one for receiving
    g. **Receiver window -**

h. A segment is retransmitted if a timer expires; the timer is changed according to previous results
    i. **RTT** = round-trip delay time;
    ii. **estimatedRTT** = (1 - a) * previous_estiamatedRTT + a * sampleRTT; a = 0.125 usually
    iii. **devRTT** = (1 - b) * previous_devRTT + b * abs(sampleRTT - estimatedRTT)
    iv. **Timeout interval** = estimatedRTT + 4 * devRTT

i. **Congestion** = too much data is sent too fast for the network to handle; symptoms = lost packets, long delays
    i. <u>Network-assisted congestion control</u> = the router provides info about the current situation to its end points, rarely used
    ii. <u>End-end congestion control</u> - the congestion status is inferred
    iii. **Congestion window** - starts from 1; when below a threshold => grows exponentially (slow-start phase); when above => grows linearly (congestion-avoidance); it's imposed by the sender
    iv. Triple duplicate ACK occurs => threshold = cw / 2, cw = threshold
    v. Timeout occurs => threshold = cw / 2, cw = 1

j. **Fairness**: k TCP sessions => each gets R / K of the bandwidth

66. connect() is normally called by the client in order to connect to a server
67. bind() is not mandatory for the client when establishing a TCP connection (when the socket is created, is gets a random IP + port by default, so it works even if we don't specify them ourselves);
68.

Mandatory calls	**TCP**	**UDP**
client	connect(), socket()	socket()
server	accept(), listen(), bind(), socket()	bind(), socket()

69. Application layer protocols: DNS, SMTP, FTP, SSH
70. **DNS** = domain name system
    a. A distributed database of domain names and IPs
    b. Uses UDP, port 53
    c. **TLD** = top level domain (.com, .ro, …); **ROTLD** = Romanian TLD; subdomain = www.plm.cs.**ubbcluj**.com
    d. **FQDN** = fully qualified domain name = hostname + domain name (www.site.com)

e. **Root DNS servers** = global, serve domains like .com, .org etc.; about 16-64 of them; they are known by all the OS which allow DNS
f. **Resolver** = the software component that translates IP to hostname
g. **Iterative website query**: local DNS server -> a root DNS s -> local DNS -> the DNS s for the domain obtained above -> local DNS -> the DNS s of the website **-sends its IP to->** local DNS -> access the website by its IP
h. **Recursive website query**: instead of returning info back to the local DNS at every step, that server goes directly to the next one (ex. from root DNS to the domain DNS)
   i. Not frequent in practice, bc those servers basically do the work of our local DNS s

71. A DNS server can be a default gateway
72. The dot in www.site.com is unrelated to the dot in an IP address
73. **ICANN** = Internet Corporation for Assigned Names and Numbers - supervises all the *domain registrars* = companies that register domain names (ex. rnc.ro)
74. **Whois** - query used for checking if a domain name is already bought
75. It's recommended that we have at least 2 DNS servers for a domain name, in case one fails (if possible not in the same location, bc earthquakes / fires / tzanca uraganu)
    a. There's a **primary / master** one, the others are **secondary / slaves** and are periodically synchronized with the master
76. **Resource records** = (domain name, TTL, class, type, value);
    a. Types:
       i. **A** = hostname for an IP add
       ii. **NS** = domain
       iii. **CNAME** = alias for (canonical) real name
       iv. **MX** = name of mailserver for that domain
       v. **SOA** = start of authority
77. **Query DNS packet** = QName (hostname) + QType (A, MX…) + QClass (type of addressing, IN = internet)
78. **FTP** = file transfer protocol
    a. Exchange files between 2 machines
    b. It's a text protocol
    c. 2 modes: active and passive
    d. SMB or SAMBA protocols
    e. 2 channels - control (command) and data
    f. Not encrypted
79. **SMTP** = simple mail transfer protocol

a. Text protocol, implemented by MX servers
b. Uses TCP
c. Works on port = 25
d. Allows for **offline** msg exchanging
e. When configuring email clients, an Internet address for an SMTP server must be entered
f. Used for reading mails:
   i. **POP3** = post office protocol
   ii. **IMAP** = internet mail access protocol

80. **HTTP** = hypertext transfer protocol
    a. Works on port = 80
    b. Is human readable
    c. <u>Also involved in sending email</u>