ECE 4309

# Basics of computer networks and systems - part 2

Dr. Valerio Formicola

CalPolyPomona

# Reference about this material

A. Tanenbaum – Computer networks:
https://www.pearson.com/en-us/subject-catalog/p/computer-networks/P200000003188/9780137523214

https://www.geeksforgeeks.org/computer-network-tutorials/?ref=lbp

And others…

# Network layer

# Network Layer

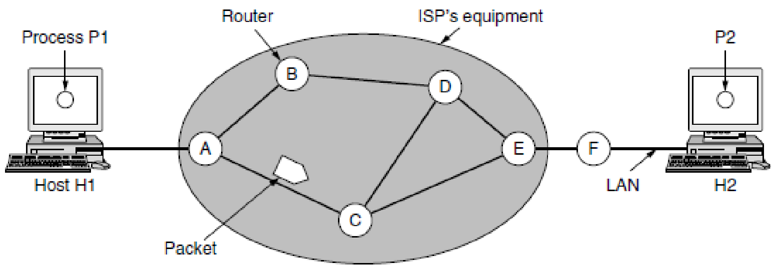
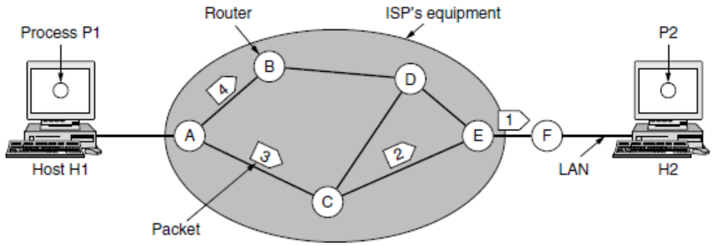



Figure 5-1. The environment of the network layer protocols.




- Connection-less
  - with primitives SEND PACKET and RECEIVE PACKET and little else
  - no packet ordering and flow control done in this layer
- Exchange *packets* between two hosts *logically* visible to each other (logical connection of two hosts)
  - they might not be physically connected to each other, but they are sending and receiving messages sent to each other
- Provides **best-effort** to send a ***packet*** from source to destination
  - Packets might follow different paths and arrive in different order, or might be corrupted or lost on the route
  - Packets are also known as ***datagrams***
- Each intermediary node implements a **routing algorithm** to know where to forward the packets arriving in input
  - Routing algorithms exist to find an optimal path: e.g., distance vector, shortest path first (Dijkstra), link state, open shortest path, etc.

# Network Layer protocol: Internet Protocol (IP)

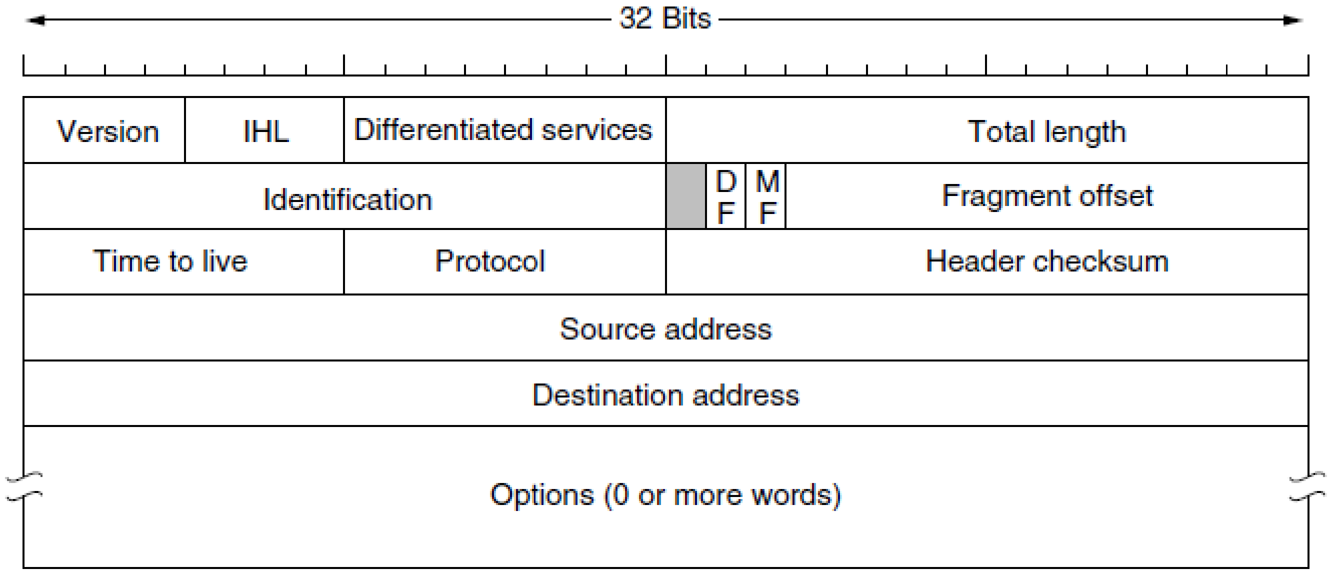



**Figure 5-46.** The IPv4 (Internet Protocol) header.

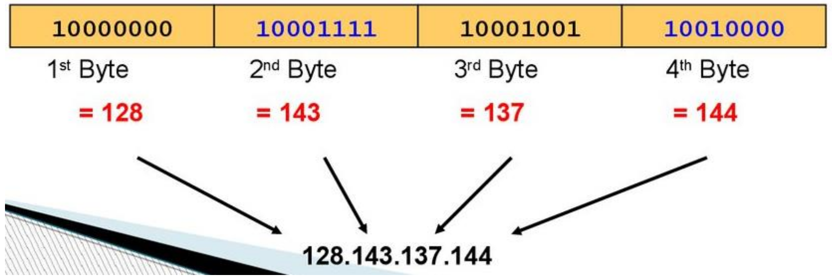
## Notable fields:

- Version: IPv4 (there is also an IPv6, but less common for user devices)
- Total length: max length in bytes $2^{16} = 65{,}535$ bytes
- Fragment offset: 13 bits to represent where the fragment goes into the whole packet (max $2^{13} = 8{,}192$)
- Time to live (TTL): 8 bits used to count numbers of hops traversed by the packet, i.e., countdown from $2^{13} = 255$.
  At 0, the packet is too old and it's dropped.
  Why? It might be a lost packet going around for long time without reaching the destination
- Protocol: 8 bit to represent the protocol above at Transmission level or some Network level protocols for operations.
  Most common: 6 (TCP), 17 (UDP), 1 (ICMP)
- Header checksum: a zero checkum to find out if the packet is corrupted (if not zero, there is a problem).
  Observation, at each hop, the TTL changes, so?

# IP Address

- It is a unique identifier of source and destination nodes within a network
  - Every network reachable node has an IP address, including the router only used to interconnect different networks
- 4 units of 1 byte each: each number in 0-255
- Dotted decimal notation: for example, 192.168.0.1
- IP is divided in two parts: **Subnetwork Prefix + Host number**
- The Subnet Prefix and the Host number can have a variable number of bits dedicated, but the total has to be always 32 (4*8 bits)

- We need the address to decide where to send the packet
  - If the packet is in the same subnet, the computer will transmit the message for all the nodes



# IP subnets

- Since the number of bits dedicated to the subnetwork address can vary, we indicate the number of bits for the network address like this:

128.208.0.0/24 here 24 is saying that the first 3 bytes (3*8 = 24) are used for the network address, the last 8 bits are used for the host number

- For the network address, the host number is replaced with a 0
- The number of bits indicated for the subnet is used to calculate the Subnet Mask
  - /24 subnet mask is 255.255.255.0
  - /26 subnet mask is 255.255.255.192

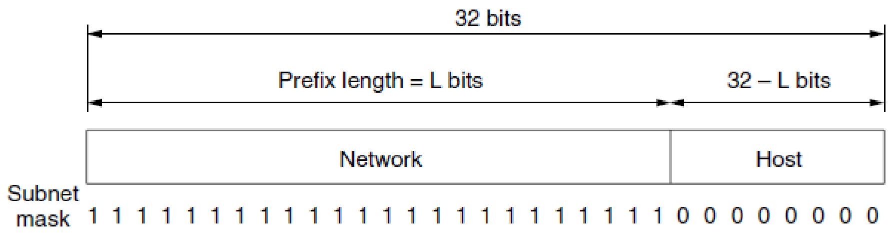
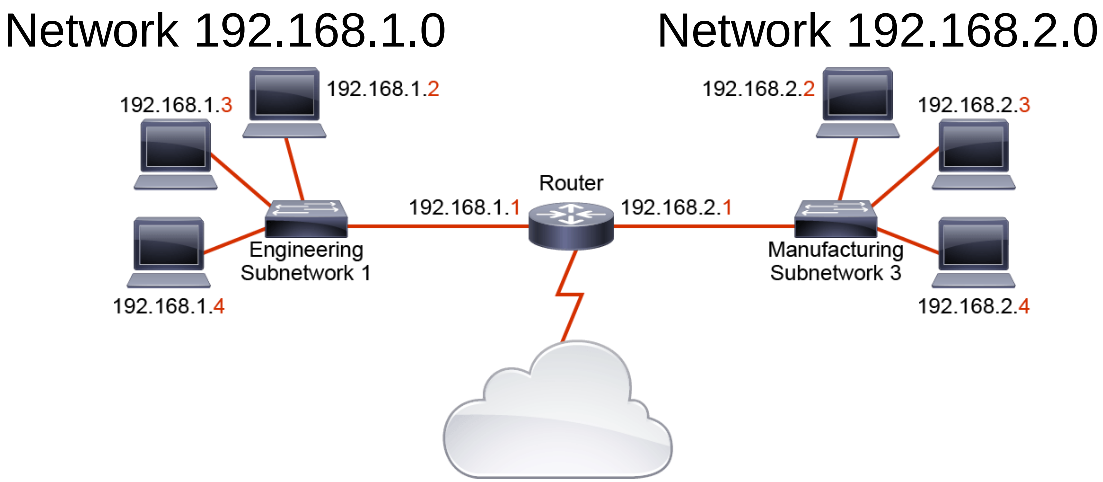



**Figure 5-48.** An IP prefix and a subnet mask.



# How to use the subnet mask

| | |
|---|---|
| IPv4 address | 192.168.50.112 |
| subnet mask | 255.255.255.0 |
| IPv4 address (binary) | 11000000.10101000.00110010.01110000 |
| subnet mask (binary) | 11111111.11111111.11111111.00000000 |
| Bitwise AND | TTFFFFFF.TFTFTFFF.FFTTFFTF.FFFFFFFF |
| Network address | 11000000.10101000.00110010.00000000 |

| | | | | |
|---|---|---|---|---|
| Network address | 192. | 168. | 50. | 0 |

In this example, the subnetwork mask is 24 bits (255.255.255.0) and the Bitwise AND gives as a result 192.168.50.0
What does it mean?
Suppose **192.168.50.1** wants to send a message to **192.168.50.2**, (or any other machine on the same subnetwork)
Your system calculates the mask: if the Subnet address of receiver is the same Subnet sender, they are both in the same subnet and there is no need to find a route to the destination.
Hence, the packet is sent to all the machines on the same subnetwork and only the receiver host id will receive the message.
If the Subnetwork address of receiver is different than the sender, then the sender has to forward the packet to the **router**, which will find the next step towards the destination

# Subnet classes and classless subnetting

- Subnetworks are divided in classes originally
- Each class has a different purpose

| Class | 1st Octet Decimal Range | Network/Host portion (N=Network, H=Host) | Default Subnet Mask | Hosts per Network (Usable Addresses) |
|---|---|---|---|---|
| A | 1 – 126 | N.H.H.H | 255.0.0.0 | 16,777,214 ($2^{24}$ – 2) |
| B | 128 – 191 | N.N.H.H | 255.255.0.0 | 65,534 ($2^{16}$ – 2) |
| C | 192 – 223 | N.N.N.H | 255.255.255.0 | 254 ($2^{8}$ – 2) |
| D | 224 – 239 | Reserved for Multicasting | | |
| E | 240 – 254 | Experimental; used for research | | |

- Some ranges of addresses are used to create a local subnetwork (private IP addresses)

| Class | Private IP address range | Subnet mask |
|---|---|---|
| A | 10.0.0.0 – 10.255.255.255 | 255.0.0.0 |
| B | 172.16.0.0 – 172.16.31.255 | 255.255.0.0 |
| C | 192.168.0.0 – 192.168.255.255 | 255.255.255.0 |

| Prefix size | Network mask | Usable hosts per subnet |
|---|---|---|
| /1 | 128.0.0.0 | 2,147,483,646 |
| /2 | 192.0.0.0 | 1,073,741,822 |
| /3 | 224.0.0.0 | 536,870,910 |
| /4 | 240.0.0.0 | 268,435,454 |
| /5 | 248.0.0.0 | 134,217,726 |
| /6 | 252.0.0.0 | 67,108,862 |
| /7 | 254.0.0.0 | 33,554,430 |
| Class A | | |
| /8 | 255.0.0.0 | 16,777,214 |
| /9 | 255.128.0.0 | 8,388,606 |
| /10 | 255.192.0.0 | 4,194,302 |
| /11 | 255.224.0.0 | 2,097,150 |
| /12 | 255.240.0.0 | 1,048,574 |
| /13 | 255.248.0.0 | 524,286 |
| /14 | 255.252.0.0 | 262,142 |
| /15 | 255.254.0.0 | 131,070 |
| Class B | | |
| /16 | 255.255.0.0 | 65,534 |
| /17 | 255.255.128.0 | 32,766 |
| /18 | 255.255.192.0 | 16,382 |
| /19 | 255.255.224.0 | 8,190 |
| /20 | 255.255.240.0 | 4,094 |
| /21 | 255.255.248.0 | 2,046 |
| /22 | 255.255.252.0 | 1,022 |
| /23 | 255.255.254.0 | 510 |
| Class C | | |
| /24 | 255.255.255.0 | 254 |
| /25 | 255.255.255.128 | 126 |
| /26 | 255.255.255.192 | 62 |
| /27 | 255.255.255.224 | 30 |
| /28 | 255.255.255.240 | 14 |
| /29 | 255.255.255.248 | 6 |
| /30 | 255.255.255.252 | 2 |
| /31 | 255.255.255.254 | 0 |
| /32 | 255.255.255.255 | 0 |

# Special addresses

- Any address terminating with all 1s in the host part, is said to be a broadcast for all the hosts in the same Subnetwork

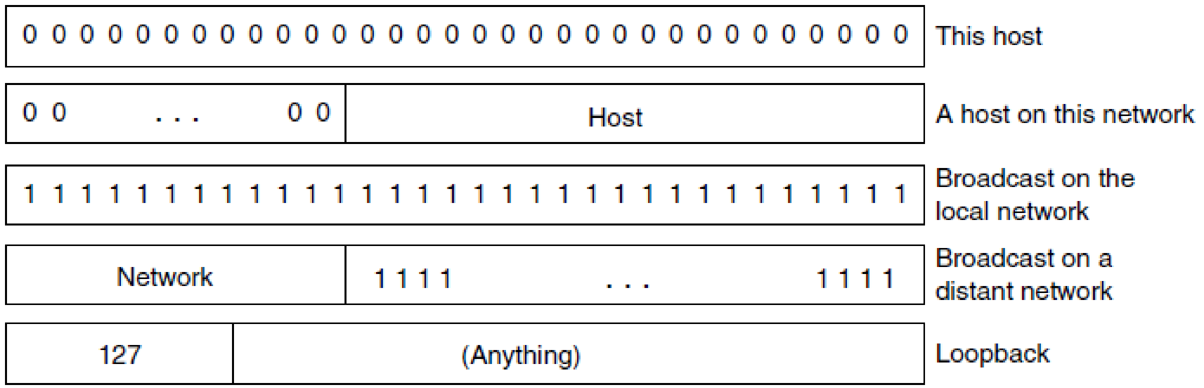



**Figure 5-54.** Special IP addresses.

For example, 192.168.50.255 is the broadcast for the subnet 192.168.50.0/24

# Private IP addresses and NAT (Network Address Translation)

- Private IP addresses are reusable in any private network (e.g., LAN)

| | | |
|---|---|---|
| 10.0.0.0 | – 10.255.255.255/8 | (16,777,216 hosts) |
| 172.16.0.0 | – 172.31.255.255/12 | (1,048,576 hosts) |
| 192.168.0.0 | – 192.168.255.255/16 | (65,536 hosts) |

- Question: if every private network uses same private IP addresses, how to machines can talk to each other if they are in different networks (because they are not anymore uniquely identified)?
- Answer: NAT, Network Address Translation

NAT is an intermediary device that connects a private network with a public netw

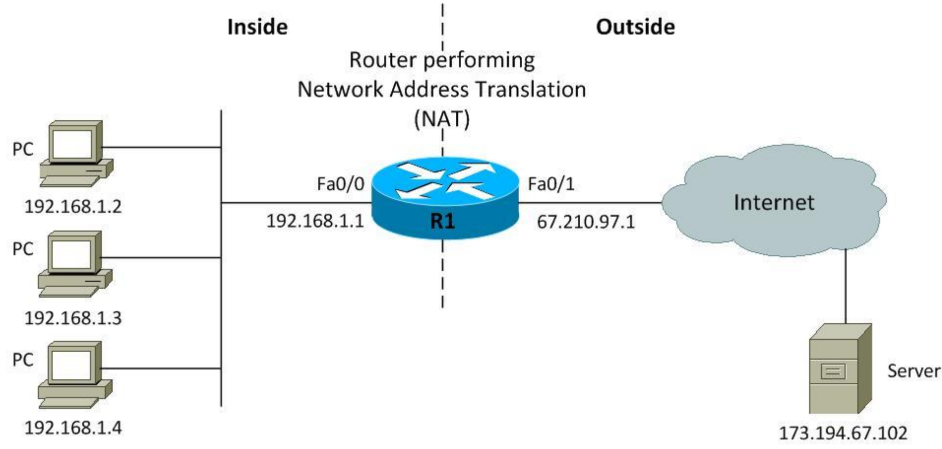
- It performs this translation of addresses in both directions (requests and replies

Local (private) IP + Transport Level Port Number

is translated to/from

Global (public) IP + Transport Level Port Number

Sometime, translation of local IP + port
to global IP + port is also called NAT overload
or PAT (Port Address Translation)




**NAT Translation Table**

| Protocol | Inside Local IP : Port | Inside Global IP : Port |
|---|---|---|
| ICMP | 192.168.1.2 : 18 | 67.210.97.1 : 18 |
| ICMP | 192.168.1.3 : 19 | 67.210.97.1 : 19 |
| ICMP | 192.168.1.4 : 20 | 67.210.97.1 : 20 |

# Network Protocol ICMP - The Internet Control Message Protocol

- A protocol to communicate problems on the network or to check if a destination is reachable

| | Bits 0–7 | Bits 8–15 | Bits 16–23 | Bits 24–31 |
|---|---|---|---|---|
| Header (20 bytes) | Version/IHL | Type of service (ToS) | Length | |
| | Identification | | *flags* and *offset* | |
| | Time to live (TTL) | Protocol | Header checksum | |
| | Source IP address | | | |
| | Destination IP address | | | |
| ICMP header (8 bytes) | Type of message | Code | Checksum | |
| | Header data | | | |
| ICMP payload (*optional*) | Payload data | | | |

| Message type | Description |
|---|---|
| Destination unreachable | Packet could not be delivered |
| Time exceeded | Time to live field hit 0 |
| Parameter problem | Invalid header field |
| Source quench | Choke packet |
| Redirect | Teach a router about geography |
| Echo and echo reply | Check if a machine is alive |
| Timestamp request/reply | Same as Echo, but with timestamp |
| Router advertisement/solicitation | Find a nearby router |

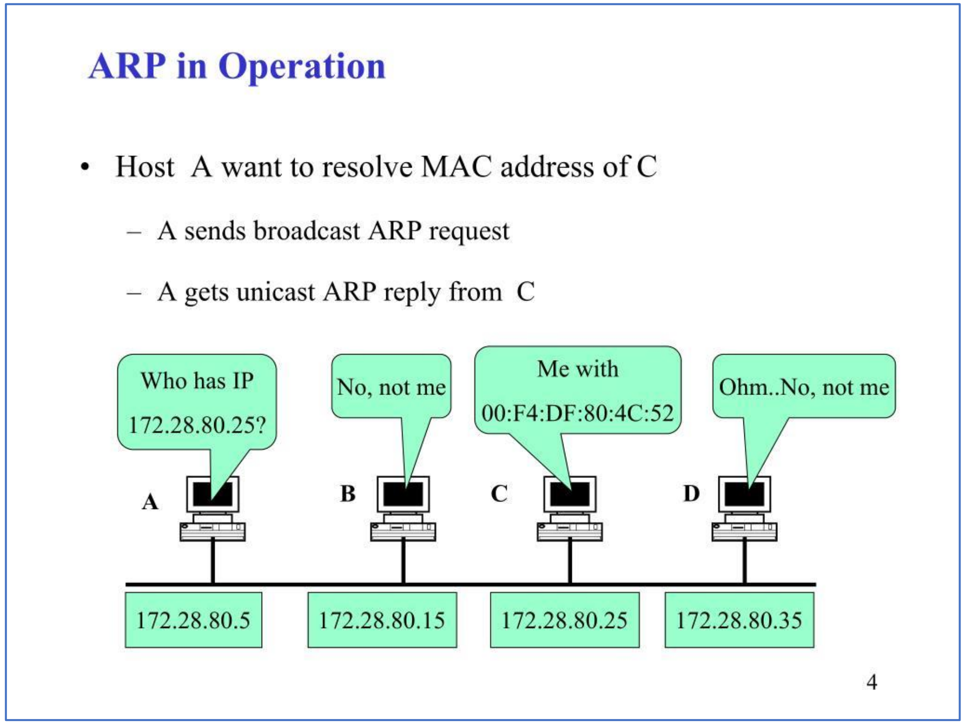
**Figure 5-60.** The principal ICMP message types.

```
$ ping -c 5 www.example.com

PING www.example.com (93.184.216.34): 56 data bytes
64 bytes from 93.184.216.34: icmp_seq=0 ttl=56 time=11.632 ms
64 bytes from 93.184.216.34: icmp_seq=1 ttl=56 time=11.726 ms
64 bytes from 93.184.216.34: icmp_seq=2 ttl=56 time=10.683 ms
64 bytes from 93.184.216.34: icmp_seq=3 ttl=56 time=9.674 ms
64 bytes from 93.184.216.34: icmp_seq=4 ttl=56 time=11.127 ms

--- www.example.com ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 9.674/10.968/11.726/0.748 ms
```

Most famous ICMP message?
**Ping (echo request)** is ICMP “Type of message” 8
**Ping (echo reply)** is ICMP “Type of message” 0

# ARP – Address Resolution Protocol (1/2)

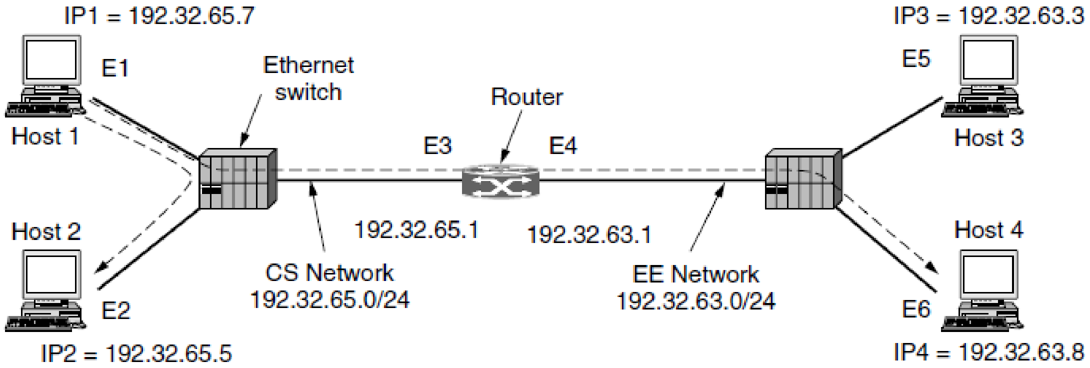
- Used in local networks to associate an IP address (layer 3) to the physical MAC address (layer 2) that should receive the message
  - ARP request and ARP response



# ARP – Address Resolution Protocol (2/2)

- If a machine is on the same physical network, sender will associate receiver IP to physical MAC address
- But if sender and receiver are not physically connected or on the same physical network, the receiver IP is associated to the router MAC address, like in the figure
- The MAC address for all the receivers NOT on the same network is the router MAC address, which is also known as **Default Gateway**
- Visualize the cache ARP in Linux with:

```
cat /proc/net/arp
```

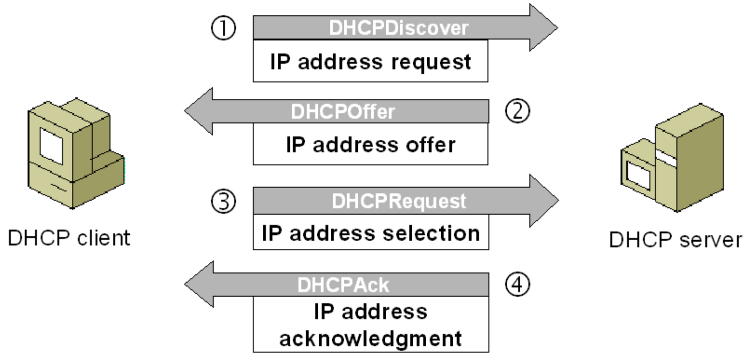



| Frame | Source IP | Source Eth. | Destination IP | Destination Eth. |
|---|---|---|---|---|
| Host 1 to 2, on CS net | IP1 | E1 | IP2 | E2 |
| Host 1 to 4, on CS net | IP1 | E1 | IP4 | E3 |
| Host 1 to 4, on EE net | IP1 | E4 | IP4 | E6 |

**Figure 5-61.** Two switched Ethernet LANs joined by a router.

# DHCP—The Dynamic Host Configuration Protocol

- How do you assign an IP address to a host, considering that you don't have to make a mistake in choosing one that is already assigned?
- Two options:
  1. Manually: you have to know if the IP address you are choosing in your network is already assigned to another node
     - If by mistake you assign to a host an already assigned IP address, a lot of problems because packets are received by two nodes and are dropped at higher layers
  2. Automatically: you use the DHCP protocol which uses a server answering requests for IP addresses
     - In addition to an offer of IP address to "lease", a DHCP server will also provide info. About the **default gateway** to send all messages in the future
     - Usually, your home router has also a DHCP server, that provides IP addresses to the computers connected in your home network



# Transport Layer

# Transport Layer

- The ultimate goal of the transport layer is to provide efficient, reliable, and cost-effective data transmission service to its users, normally processes in the application layer.
- There are many protocols at this level: most notably, **TCP** and **UDP**
- Protocols at this layer provide idea that there is a *reliable and dedicated data channel* between two applications in two different computers **(logical connection of processes/apps)**
  - Acknowledgement: confirmation of data received correctly
  - Multiplexing: making many applications send data from the same computer (with use of **port numbers**)
  - Segmentation/reassembly: divide larger pieces of data into **segments**
  - Flow control: avoid congestion of data
  - Error control: errors in transmission for corrupted data
  - Sequence control: check the order of messages

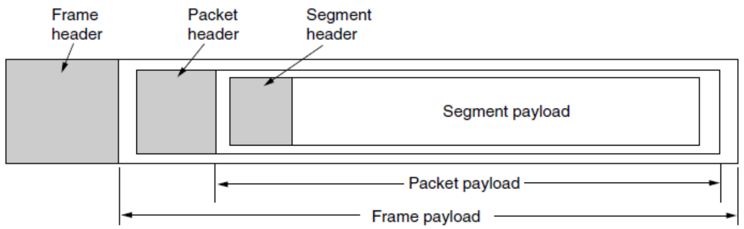
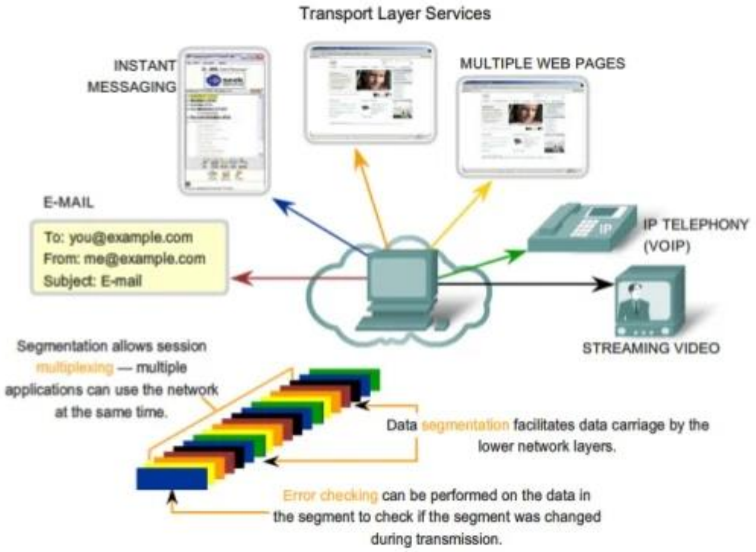
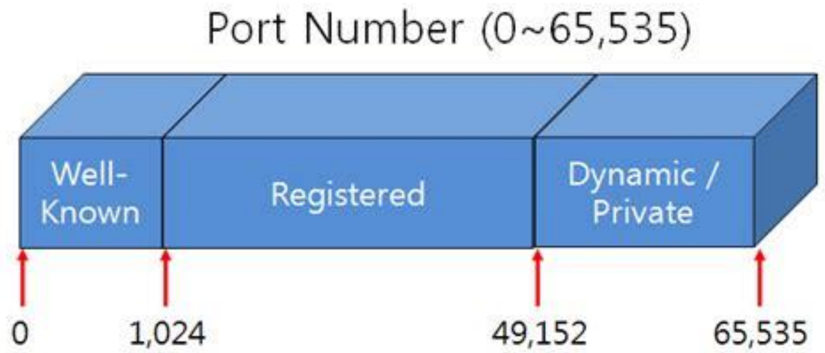



**Figure 6-3.** Nesting of segments, packets, and frames.



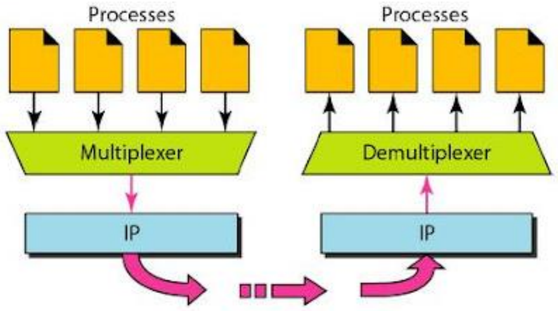
# Port numbers

- Ports are used to implement multiplexing/demultiplexing in TCP and UDP protocols:
  - I.e., multiple applications on the same “data channel”
- There are 65,535 port numbers
- First 1,024 dedicated to well-known application **servers** (like default)
  - for example: a server on port TCP 80 is typically providing web-pages
  - Remember that each machine has also a unique identifier which is the IP address to be reached




Well-known ports

| Port number | Protocol | Application |
|---|---|---|
| 20 | TCP | FTP data |
| 21 | TCP | FTP Control |
| 23 | TCP | Telnet |
| 25 | TCP | SMTP |
| 53 | TCP, UDP | DNS |
| 69 | UDP | TFTP |
| 80 | TCP | HTTP (web) |
| 110 | TCP | POP3 |
| 161 | UDP | SNMP |
| 520 | UDP | RIP |



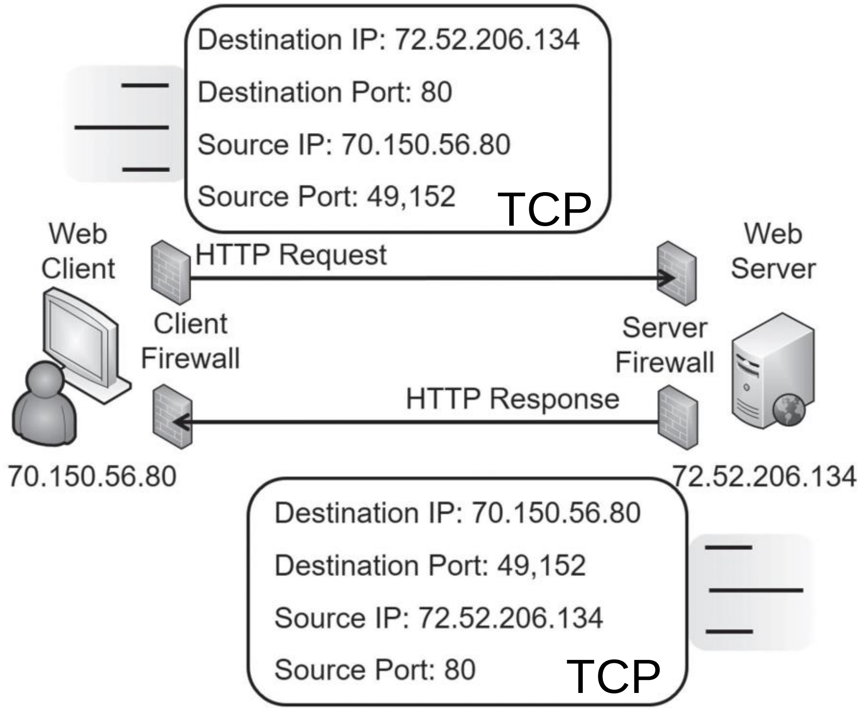

Each port is associated to one app in multiplex

On a single host, each port corresponds to an application that wants to communicate

If the port is associated to **a server application**, Then there are some very common numbers as above

# Network Flow as a unique data channel

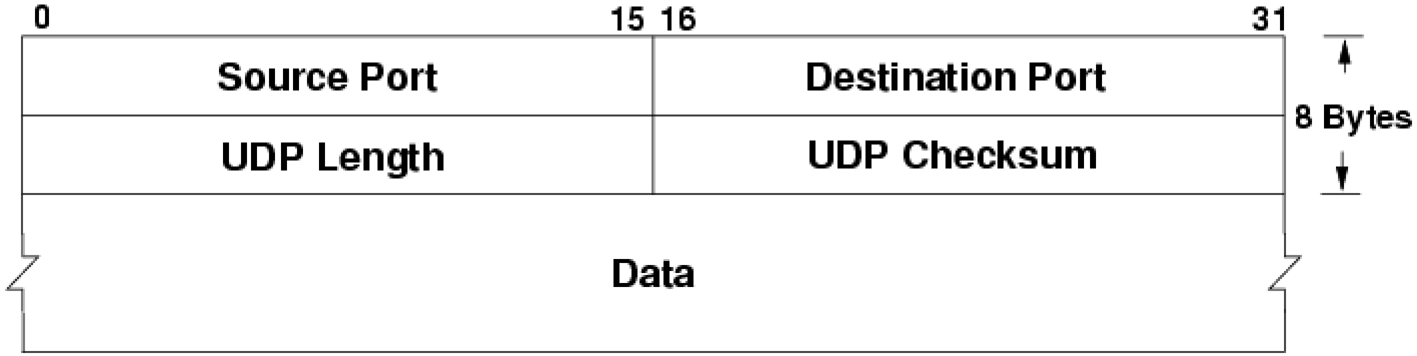
Unique network flow: *Connection protocol (UDP or TCP), Source Port, Source IP, Destination Port, Destination IP*



# A Simple Transport Layer protocol: UDP (Unreliable Datagram Protocol)

- Connection-less, like the underlying IP protocol
- It doesn't do much, just add Source Port, Destination Port, Length and a checksum for errors.
- Used if your application doesn't need a reliable transmission
    - E.g., DNS (Domain Name Service), VoIP (Voice over IP), SMTP (Simple Mail Transport Protocol), some videogames or video streaming applications

App data here



# A reliable Transport Layer protocol: TCP (Transmission Control Protocol)

- Connection-oriented:
  - Establish a communication channel before starting sending data
  - Performs congestion control, sequence order, error control, and other services for quality of transmission
- Notably fields:
  - Sequence number: order of segment in the sequence
  - Ack number: number of message ack-ed since received correctly
  - Window size: size of receiver window in bytes to avoid overload
  - Checksum: to find errors in transmission

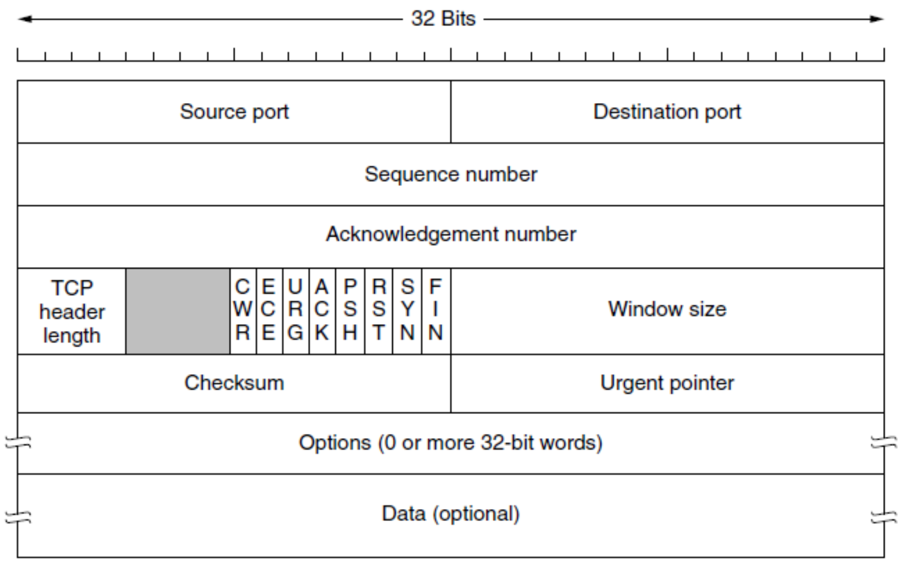
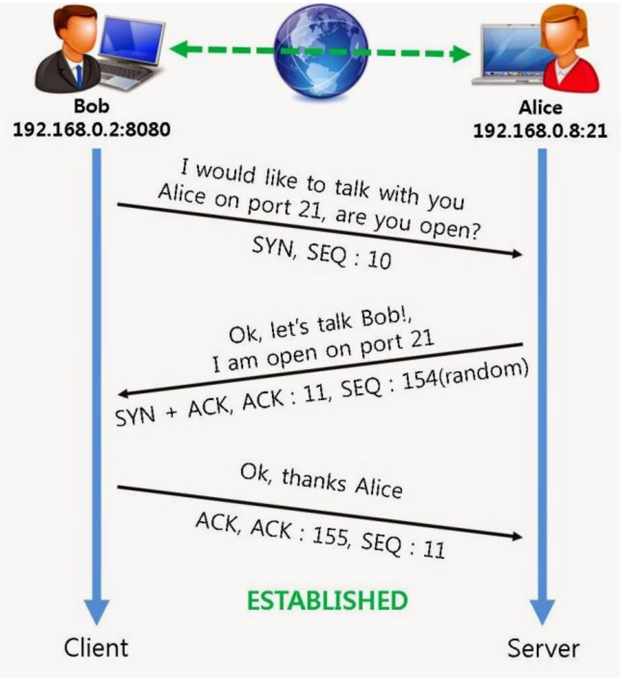


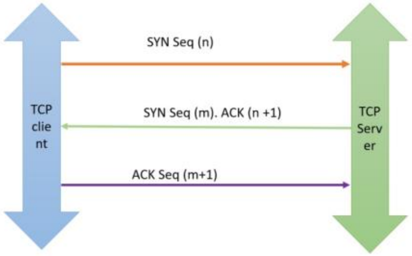

**Figure 6-36.** The TCP header.

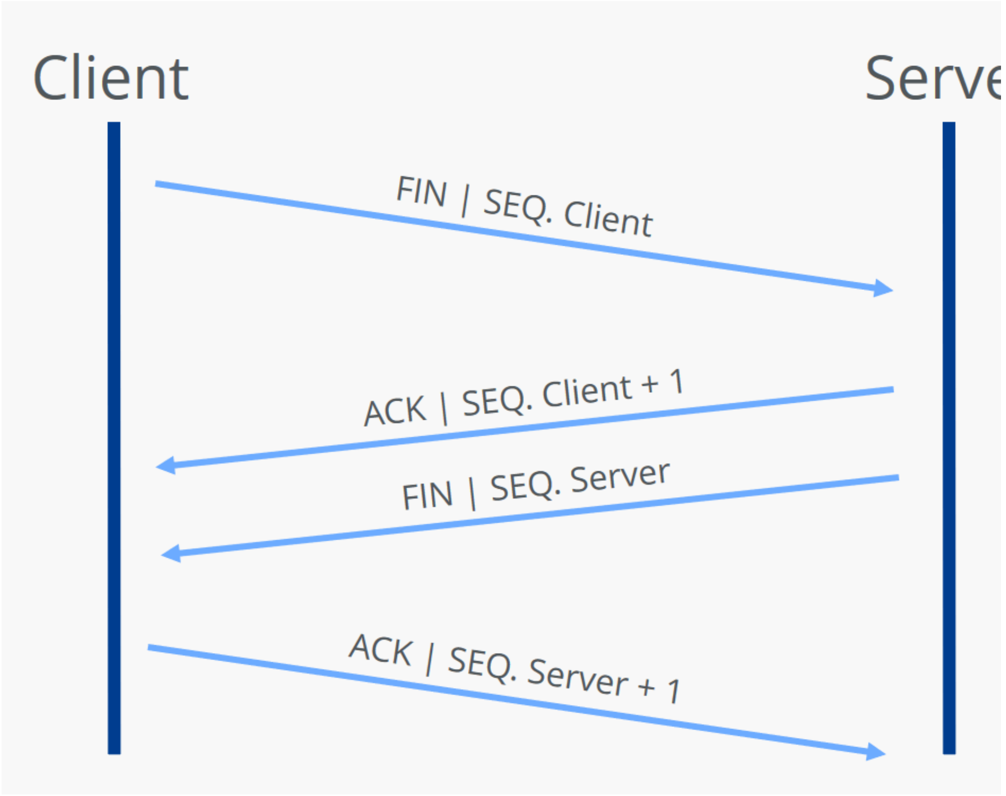
# TCP connection establishment and termination

- Establish (Three-way handshake): SYN, SYN+ ACK, ACK

Termination:
FIN, ACK, FIN, ACK

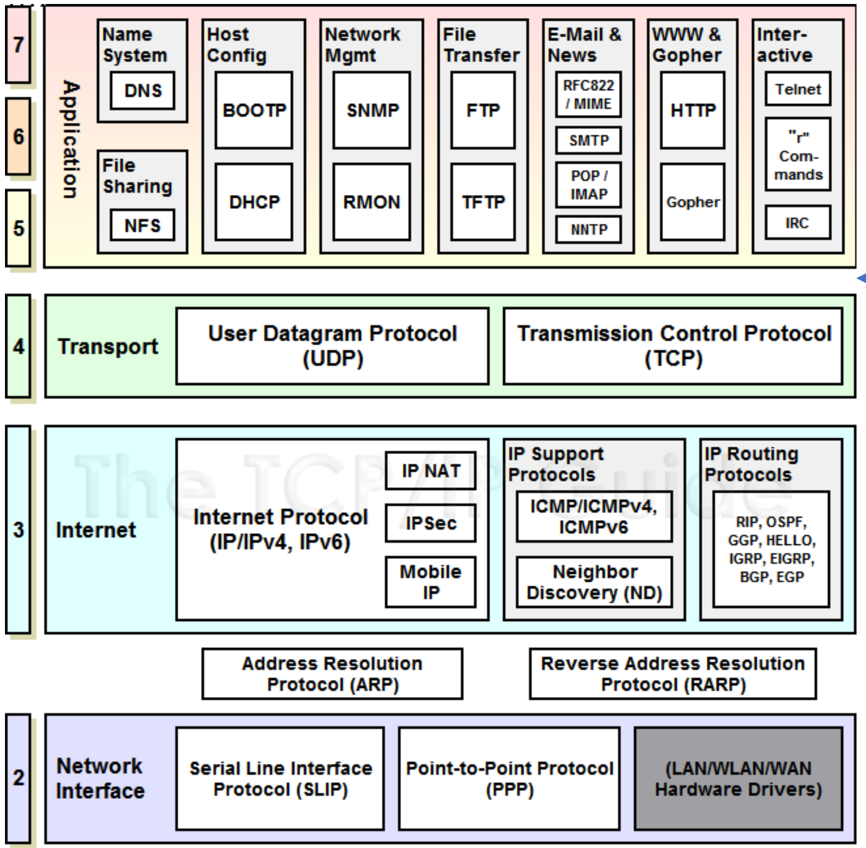


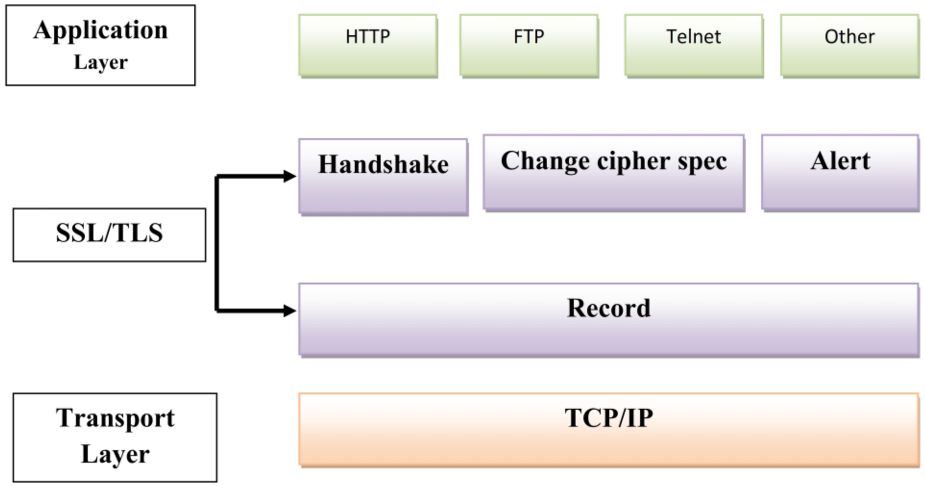
# Most common protocols for each layer and use of security

Without *Transport level security*



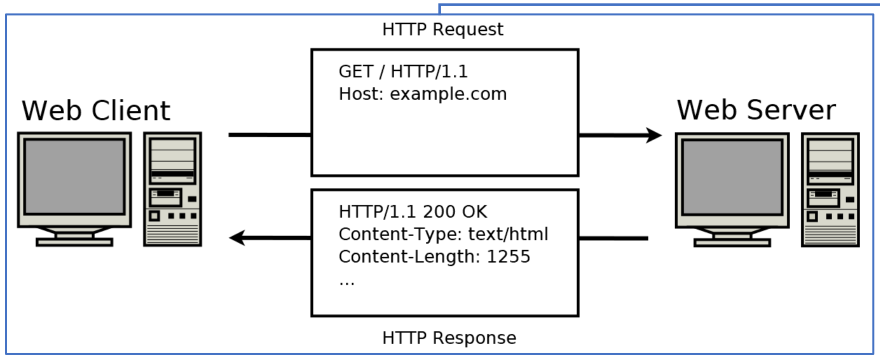

In between App and Transport layers

With *Transport level security*

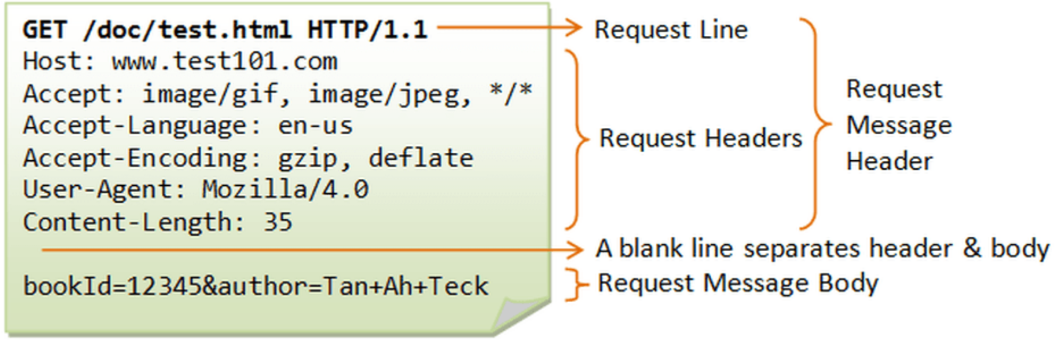


# Application Layer

# HTTP – Hypertext Transfer Protocol




HTTP header




Connection protocol and port for server: TCP on 80 or 8080

HTTP Methods

| Method | Description |
|---|---|
| GET | Read a Web page |
| HEAD | Read a Web page's header |
| POST | Append to a Web page |
| PUT | Store a Web page |
| DELETE | Remove the Web page |
| TRACE | Echo the incoming request |
| CONNECT | Connect through a proxy |
| OPTIONS | Query options for a page |

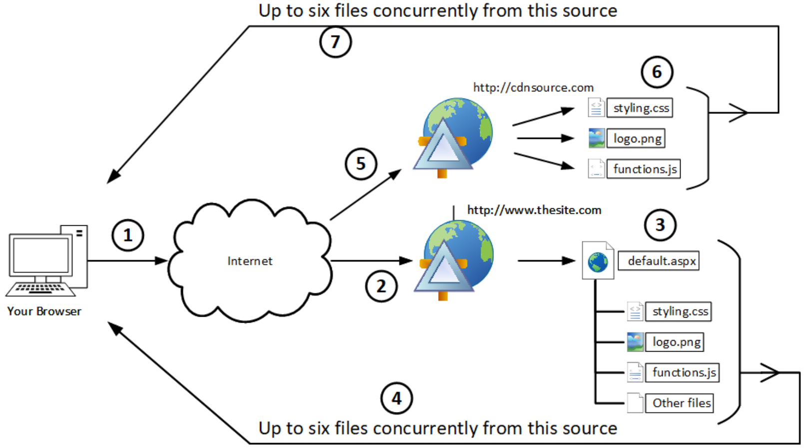
Figure 7-37. The built-in HTTP request methods.

HTTP Responses

| Code | Meaning | Examples |
|---|---|---|
| 1xx | Information | 100 = server agrees to handle client's request |
| 2xx | Success | 200 = request succeeded; 204 = no content present |
| 3xx | Redirection | 301 = page moved; 304 = cached page still valid |
| 4xx | Client error | 403 = forbidden page; 404 = page not found |
| 5xx | Server error | 500 = internal server error; 503 = try again later |

Figure 7-38. The status code response groups.

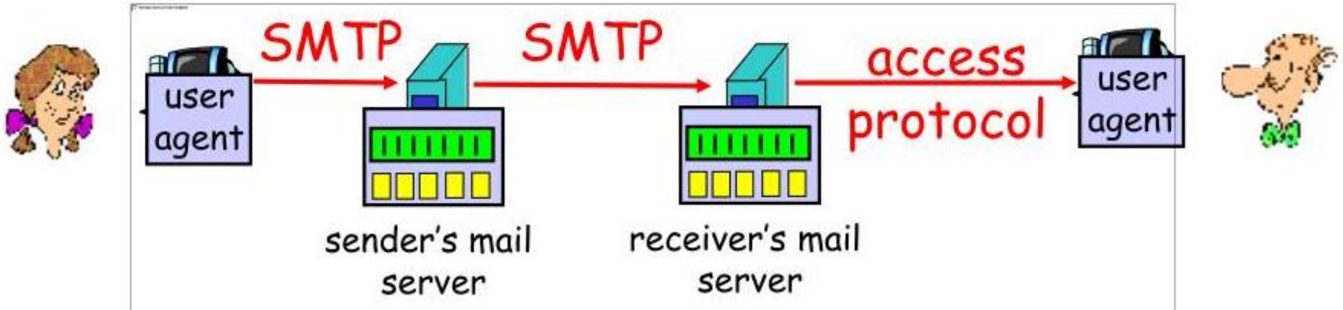
Usually, a web-site contains multiple files that you need to download to visualize a web-page. Also, the files might be located in different servers:




Note: HTTP is not only used for web navigation, even if it was started for that

# E-mail access protocols

- **SMTP** is to push an email to the receiver of an email
- **POP3** is to retrieve an email received on your email service
- **IMAP** is similar to POP3 but with more services



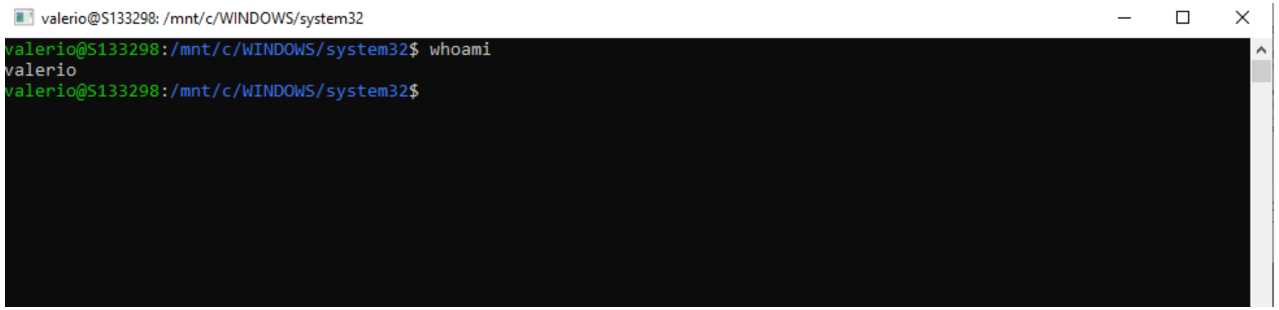

- SMTP is a push protocol. How will a user access emails?
- Mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]
    - Users can't create folders on mail server
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - more features (more complex)
    - manipulation of stored msgs on server
  - HTTP: Hotmail , Yahoo! Mail, etc.

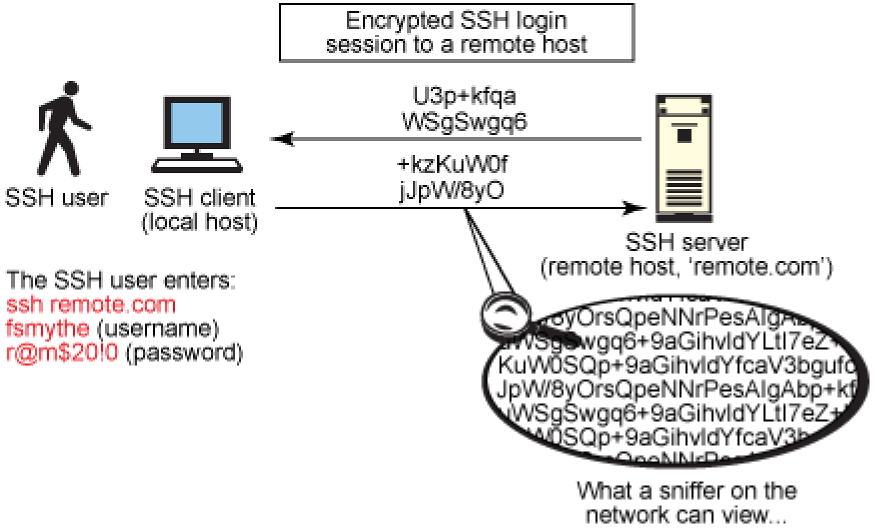
Connection protocol and port for server: SMTP is on TCP 25, POP3 is on TCP 110, IMAP is on TCP 143

# Secure Shell (SSH)

- It’s a command shell but secure to execute commands from remote

A bash shell




Connection protocol and port for server: TCP 22

# Uniform Resource Identifier (URI) and Locator (URL)

- How can we identify a web-site or any other resource on a network in a "human" understandable way?
- URI: A **Uniform Resource Identifier (URI)** is a unique sequence of characters that identifies a logical or physical resource used by web technologies.
- Some URIs provide a means of locating and retrieving information resources on a network (either on the Internet or on another private network, such as a computer filesystem or an Intranet); these are **Uniform Resource Locators (URLs)**.

### URI -- Uniform Resource Identifier

URIs are a standard for identifying documents using a short string of numbers, letters, and symbols. They are defined by RFC 3986 - Uniform Resource Identifier (URI): Generic Syntax. URLs, URNs, and URCs are all *types* of URI.

### URL -- Uniform Resource Locator

Contains information about how to fetch a resource from its location. For example:

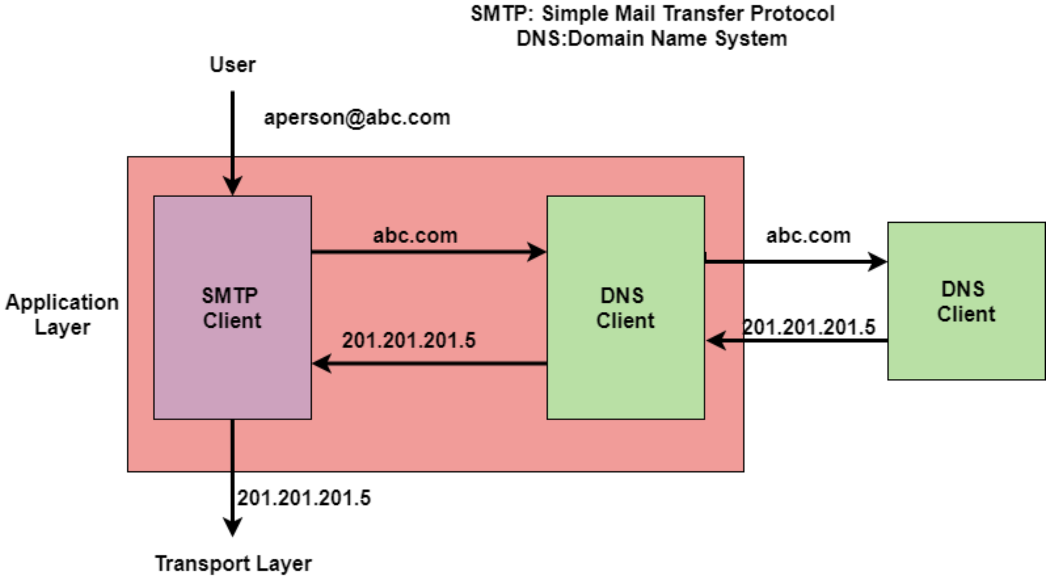
- `http://example.com/mypage.html`
- `ftp://example.com/download.zip`
- `mailto:user@example.com`
- `file:///home/user/file.txt`
- `tel:1-888-555-5555`
- `http://example.com/resource?foo=bar#fragment`
- `/other/link.html` (A relative URL, only useful in the context of another URL)

URLs always start with a protocol (`http`) and usually contain information such as the network host name (`example.com`) and often a document path (`/foo/mypage.html`). URLs may have query parameters and fragment identifiers.

# Domain Name Service (DNS): Association URL-IP address

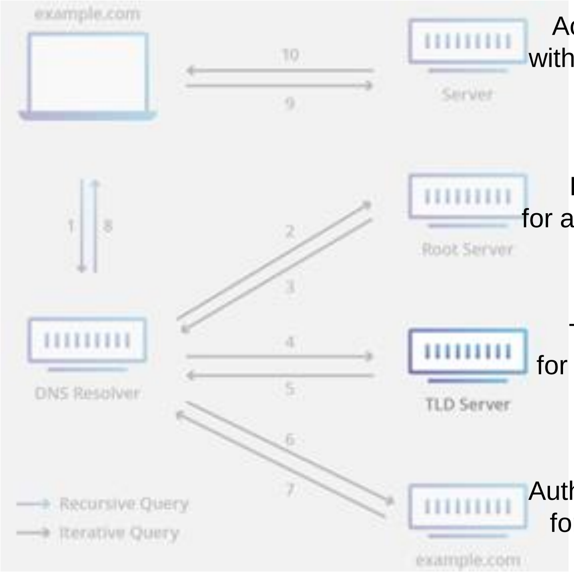
- Applications use a DNS client located on the same host to translate the URL to IP address.
- The applications (e.g., a web browser, an email client, etc.) send a request to a UDP port 80 of a DNS  server to translate the URL of interest (e.g., a website name) into the server IP address
- Finding the IP address of an URL is also called **IP resolution**

For example:
Google hosts a DSN server at 8.8.8.8 to solve the requests from users (not only to Google.com)



# Domain Name Service (DNS): Distributed Architecture

- DNS is a **distributed protocol.**
  All DNS servers fall into one of four categories: *Recursive resolvers, root nameservers, TLD nameservers (top level domain .com, .gov., .net, .us, etc.), and authoritative nameservers (final resolution of www.whatever.com)* . (Sometime also SLD, Second-level domains like .co.us, .de.us, etc.)
  In a typical DNS lookup (when there is no caching in play), these four DNS servers work together in harmony to complete the task of delivering the IP address for a specified domain to the client (the client is usually a *stub resolver* - a simple resolver built into an *operating system*).



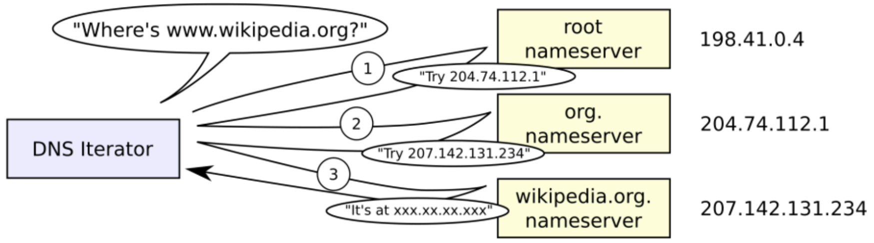

Actual server
with needed data

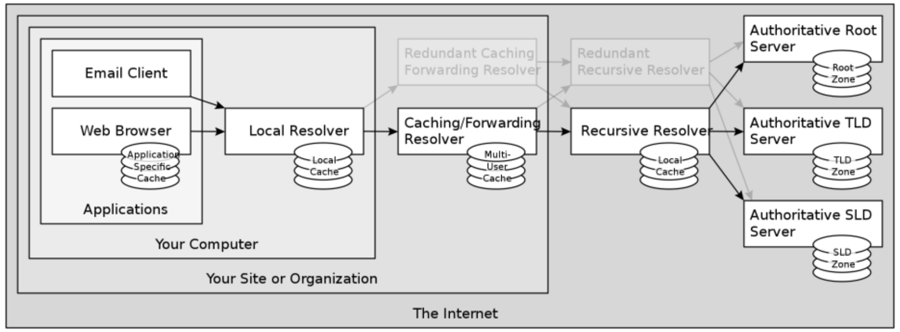
Root server
for all name servers

TLD server
for all .com URLs

Authoritative Server
for example.com

1: I need the resolution for example.com
2: Q: Where is IP of example.com?
3: R: Ask the TLD server for all .com URLs
4: Q: Where is IP of example.com?
5: R: Ask the auth. server of example.com
6: Q: Where is IP of example.com?
7. R: Is at Server
8: R: You have to connect to Server
9: Dear Server, I need your app. data
10. Here is your app. data



# Repeaters and Hubs

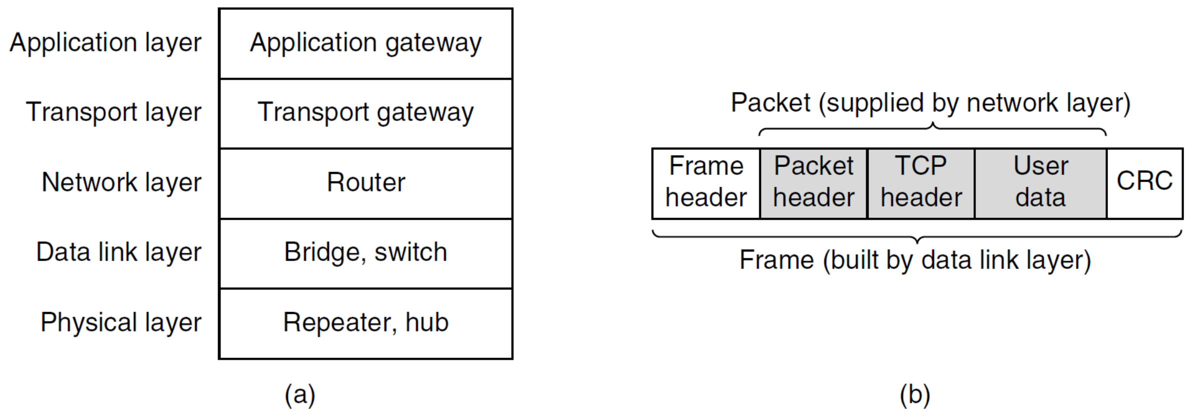



**Figure 4-45.** (a) Which device is in which layer. (b) Frames, packets, and headers.

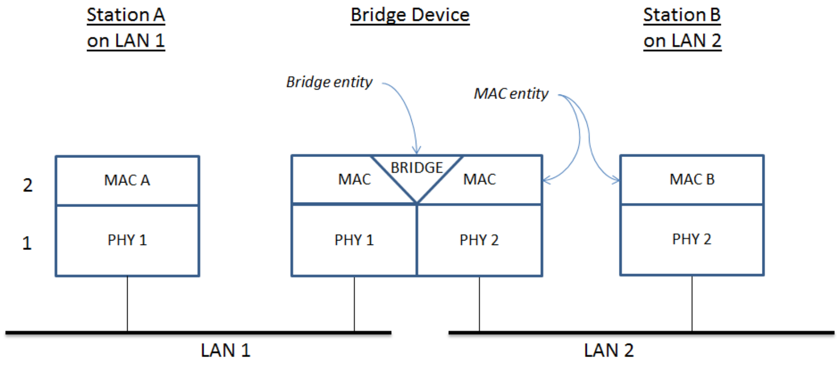
These devices provide connections at different layers:

- Repeaters and hubs simply try to replicate a physical signal from
  one trunk of network to another.
  If they are "smart", they are also able to clean the
  bits and regenerate the bit sequence

# Bridges and Switches

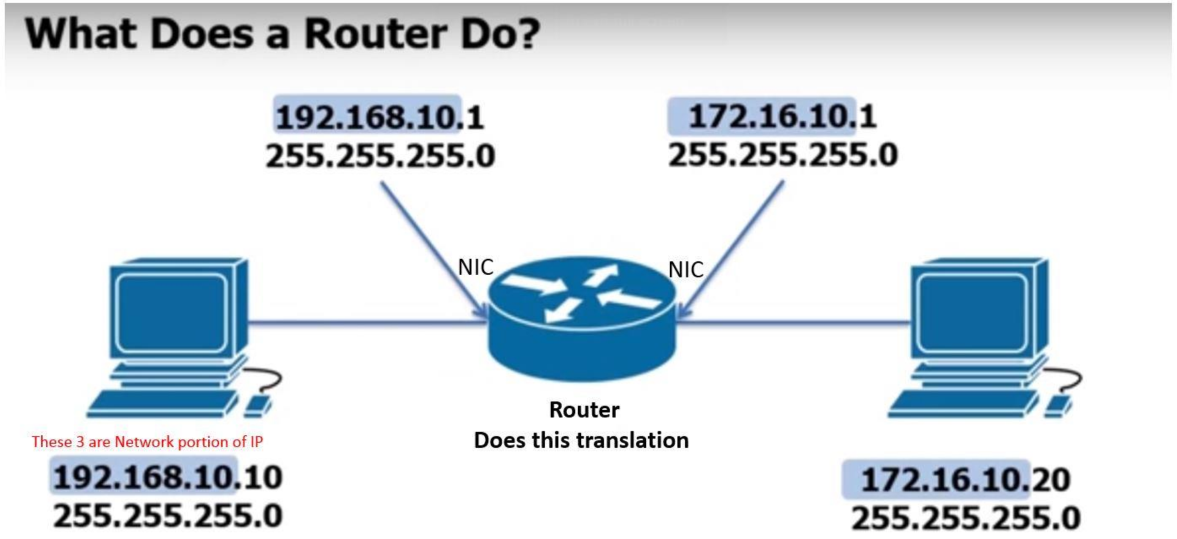
- Bridges and Switches give the service to connect physically two separate physical mediums:
- Switches: if the two media are similar (e.g., two LANs on a UTP twisted pair cable) then the switch will simply connect those two cables and all the MAC addresses in the two LANs are reachable
- Bridges: If two or more physical media are different at physical level (e.g., a Wifi on air and a LAN on twisted pair), then the bridge will show all the devices physically connected on the same level

A bridge connecting two LAN segments




# Routers

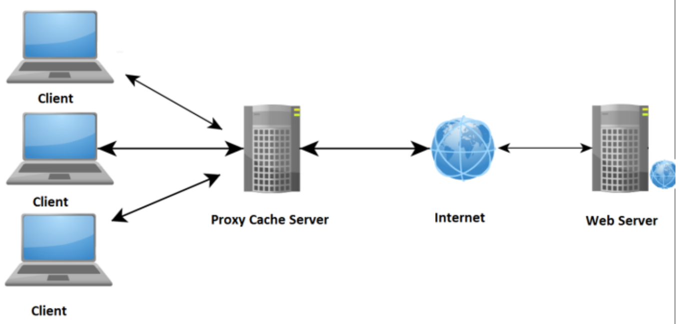
- A router is a network node, hence it has a dedicated IP address for each separate network connected to it
- The purpose of a router is to connect two separate LAN networks with different network addresses

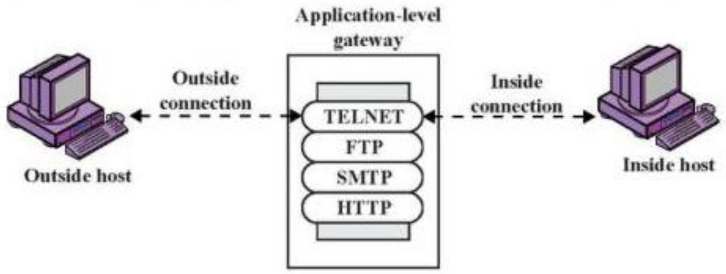



Note:
In IP communication, two machines from a different network can not directly communicate.
Its needs router logic(routing tables) to communicate or transfer the IP packets to machine in different network.

# Application level Gateways

- The term **gateway** is the general name for a machine that makes a connection between two or more networks and provides the necessary translation, both in terms of hardware and software.
- Gateways are distinguished by the layer at which they operate in the protocol hierarchy.
- A **proxy server** is a kind of gateway that intercedes between clients and servers to perform multiple requests from a client to servers
- Sometime the proxy server is combined with a **firewall** and it's called **application-level firewall**. In this case does not only replicate requests but it also performs some filtering, i.e., it changes the content to retrieve local information, or to block content considered not appropriate or malicious.




- Splices and relays two application-specific connections
  - Example: Web browser proxy
  - Daemon spawns proxy process when communication is detected
  - Big processing overhead, but can log and audit all activity
- Can support high-level user-to-gateway authentication
  - Log into the proxy server with your name and password
- Simpler filtering rules than for arbitrary TCP/IP traffic
- Each application requires implementing its own proxy