Networks (Tutorial). Tutorial 02

Shinnazar Seytnazarov, PhD

Innopolis University

s.seytnazarov@innopolis.ru

January 21, 2022

Topic of the lecture

- Computer Networks
- Types of Networks
- Open Systems Interconnection (OSI) Model
- The Application Layer
- Application Architecture
- Principles of Network Applications
- Web and HTTP
- FTP

Topic of the tutorial

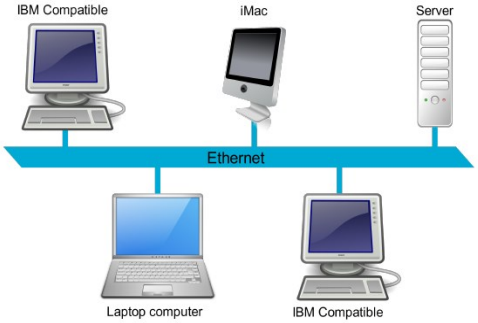
- A brief overview of physical and logical topology
- The application layer protocol (HTTP)

Network Topology

- Geographical representation of the links.
- Topology is physical layout of computers, cables and other connected devices on a network.
- Types
  - Physical Topology
  - Logical Topology

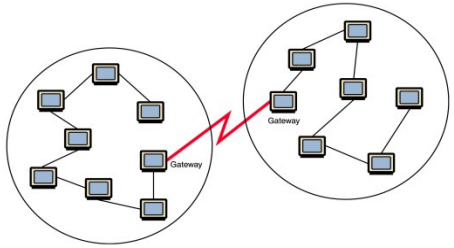
Network Characteristics

Property	Details
Network Type (based on geographical scale)	Local Area Network (LAN) Wide Area Network (WAN) Metropolitan Area Network (MAN)

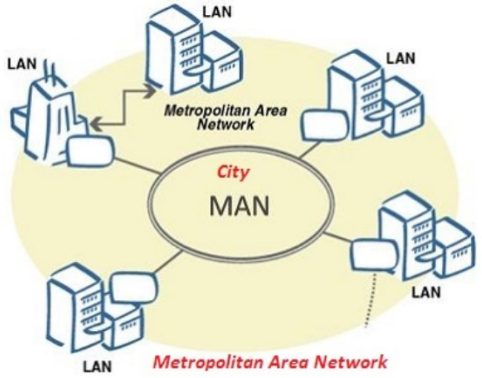


LAN network (e.g. at home)

(image taken from Wikipedia.com)



WAN network connecting several LANs



MAN network covering a large city

Network Characteristics

Property	Details
Network Type (based on geographical scale)	Local Area Network (LAN) Wide Area Network (WAN) Metropolitan Area Network (MAN)
Communication Technology (depends on network type)	**LAN: Ethernet, Wi-Fi (aka wireless LAN - WLAN)** **WAN: ATM, DSL, ISDN, fiber-optic communications.** **MAN: Gigabit Ethernet, MPLS**

Network Characteristics

Property	Details
Network Type (based on geographical scale)	Local Area Network (LAN) Wide Area Network (WAN) Metropolitan Area Network (MAN)
Communication Technology (depends on network type)	LAN: Ethernet, Wi-Fi (aka wireless LAN - WLAN) WAN: ATM, DSL, ISDN, fiber-optic communications. MAN: Gigabit Ethernet, MPLS
Performance Metrics	**Bandwidth (bits/sec), max. transfer rate** **Throughput (bits/sec), actual (or average) transfer rate** **Error rate (occurrence of bit errors), Latency**

Network Characteristics

Property	Details
Network Type (based on geographical scale)	Local Area Network (LAN) Wide Area Network (WAN) Metropolitan Area Network (MAN)
Communication Technology (depends on network type)	LAN: Ethernet, Wi-Fi (aka wireless LAN - WLAN) WAN: ATM, DSL, ISDN, fiber-optic communications. MAN: Gigabit Ethernet, MPLS
Performance Metrics	Bandwidth (bits/sec), max. transfer rate Throughput (bits/sec), actual (or average) transfer rate Error rate (occurrence of bit errors), Latency
Network Topology (physical/logical)	**Bus, ring, star, extended star, hierarchical, mesh, hybrid**

Physical Topology: the layout of the computer cables and other network devices (e.g. hosts, routers, switchers, etc)

Logical Topology: the layout of how data (signals) are transferred in a network

Network Characteristics

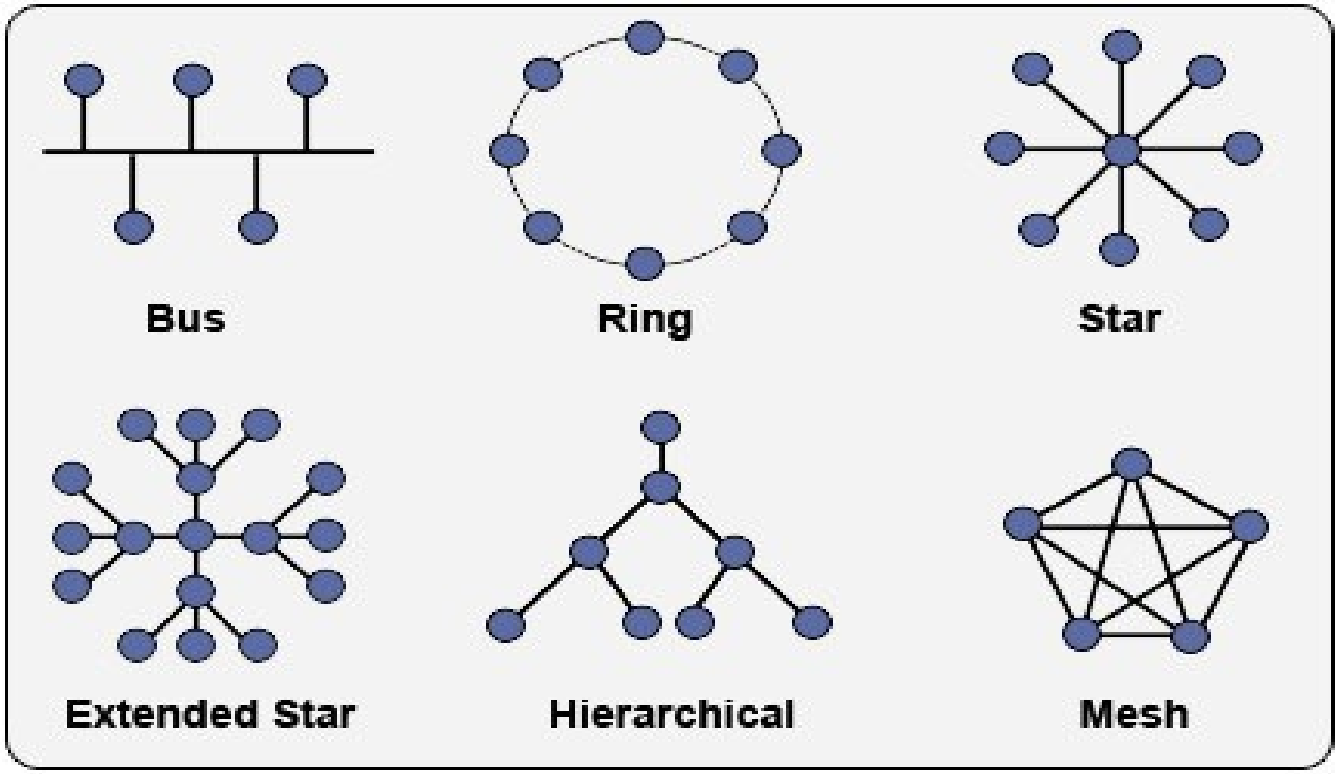
Property	Details
Network Type (based on geographical scale)	Local Area Network (LAN) Wide Area Network (WAN) Metropolitan Area Network (MAN)
Communication Technology (depends on network type)	LAN: Ethernet, Wi-Fi (aka wireless LAN - WLAN) WAN: ATM, DSL, ISDN, fiber-optic communications. MAN: Gigabit Ethernet, MPLS
Performance Metrics	Bandwidth (bits/sec), max. transfer rate Throughput (bits/sec), actual (or average) transfer rate Error rate (occurrence of bit errors), Latency
Network Topology (physical/logical)	Bus, ring, star, extended star, hierarchical, mesh, hybrid
Proprietary Type	**Open systems (relies on OSI model)** **Proprietary systems (relies on private technologies)**

Types of Topology

- **Physical Topology**
  - Actual layout of the computer cables and other network devices.

- **Logical Topology**
  - The way in which the network appears to the devices that use it.
  - Refers to how data is actually transferred in a network.

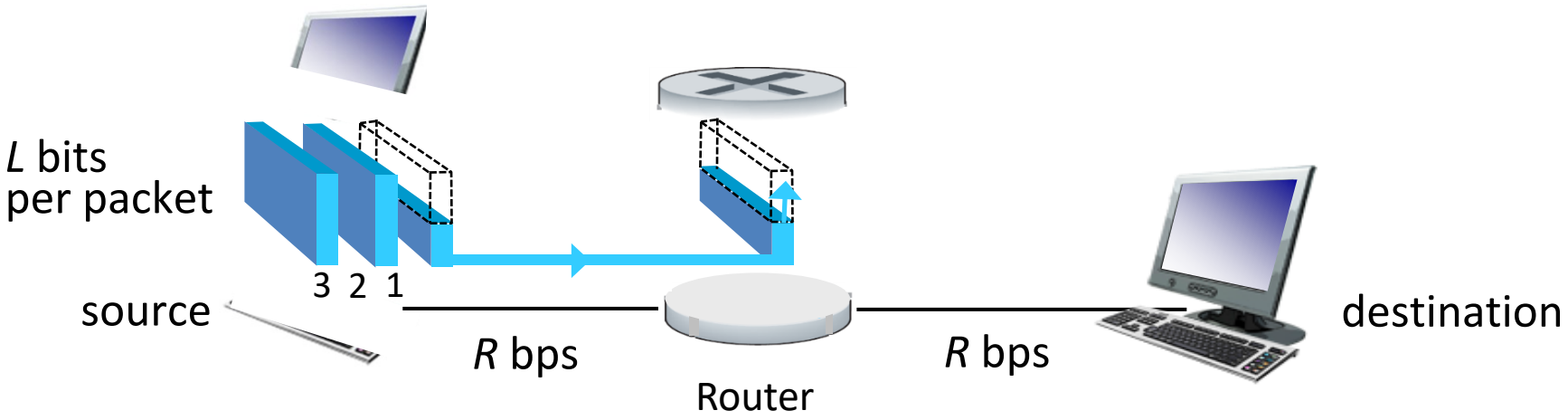
Network Topology



Selection Criteria

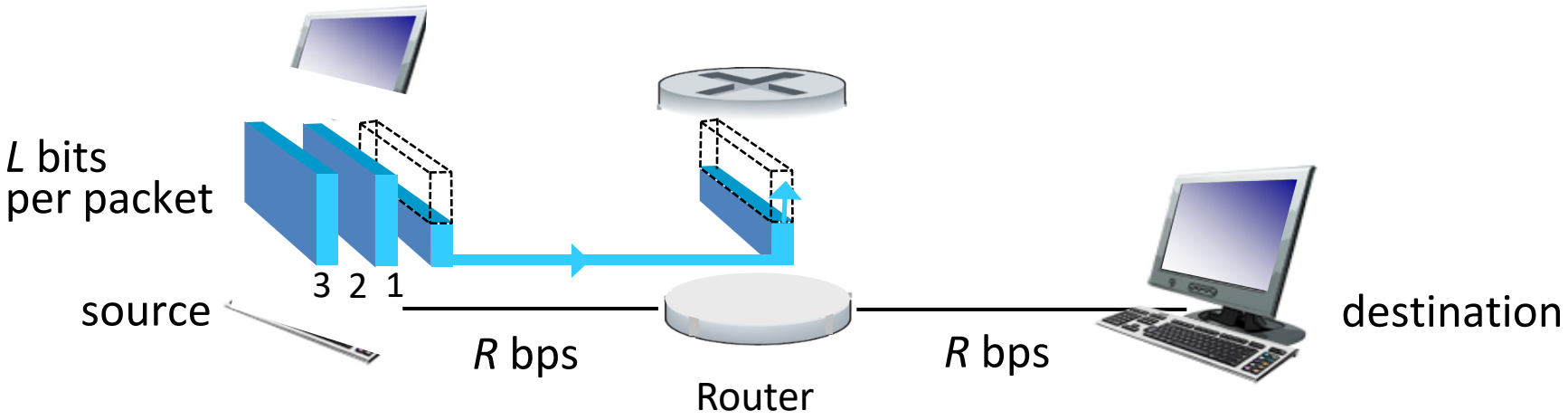
- Size (no of node) of the system
- Cost of the components and service required
- Management of network
- Architecture of network
- Cable type
- Expandability of the network
- The desired performance
- Reliability

Class exercise



- Source takes L/R seconds to transmit (push out) the packet into link which has R bps bandwidth
- store and forward : entire packet must arrive at router before it can be transmitted on next link to destination
- End-to-end delay for a packet = 2L/R (assuming zero propagation delay)

Class exercise



Source starts transmitting the 1st packet at time 0 and R is link bandwidth

- Q1) assuming zero propagation delay, when the destination receives all three packets? Answer: 4L/R
- Q2) How about if each link has propagation delay of t seconds?
- Q3) What if there were N routers?

What is a Protocol?

- In diplomatic circles, a protocol is the set of rules governing a conversation between people
- We have seen that the client and server carry on a machine-to-machine conversation
- A network protocol is the set of rules governing a conversation between a client and a server

Network Protocols

- The rules are defined by the inventor of the protocol – may be a group or a single person.
- The rules must be precise and complete so programmers can write programs that work with other programs.
- The rules are often published as an RFC* along with running client and server programs.

*RFC = request for comments

Application Layer Protocols

Protocol	Application
HTTP: Hypertext Transfer	Retrieve and view Web pages
FTP: File Transfer	Copy files from client to server or from server to client
SMTP: Simple Mail Transport	Send email
POP: Post Office	Read email

Hypertext Transfer Protocol (HTTP)

- Tim Berners-Lee added to the Internet to create the World Wide Web had two fundamental dimensions
  - Connectivity
  - Interface.

- He invented a new protocol for the computers to speak as they exchanged hypermedia documents

- A computer that asked for a file from another computer would know, when it received the file, if it was a picture, a movie, or a spoken word.

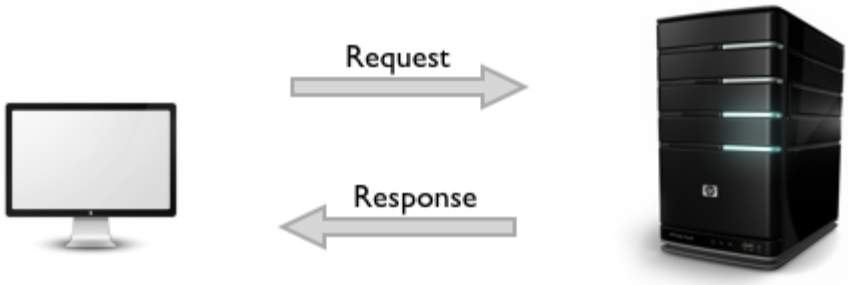
Hypertext Transfer Protocol (HTTP)

- It is an application-layer protocol for communicating between distributed systems.

- It allows for communication between a variety of hosts and clients, and supports a mixture of network configurations.

HTTP – Client-Server Architecture

- The HTTP protocol is based on a request/response paradigm



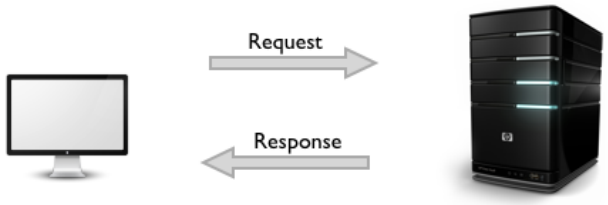
- The communication usually takes place over TCP/IP, but any reliable transport can be used.

- The default port for TCP/IP is **80**, but other ports can also be used.

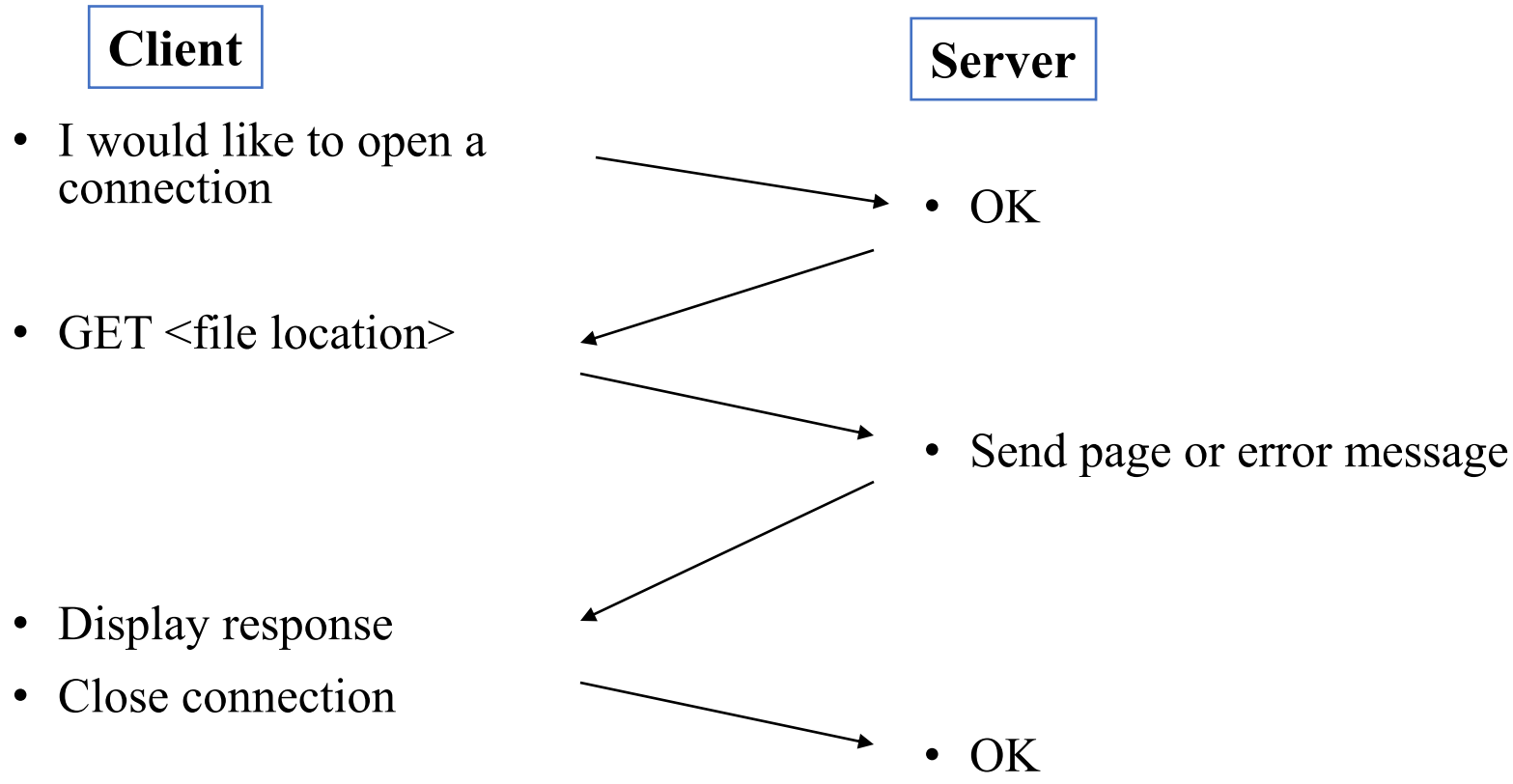
HTTP – Client-Server Architecture

- **Client**
  - A client establishes a connection with a server and sends a request in the form of a request method, URI, and protocol version, followed by a message containing request modifiers, client information, and possible body content.

- **Server**
  - The server responds with a status line, including its protocol version and a success or error code, followed by a message containing server information, entity metainformation, and possible body content.



HTTP Conversation



HTTP is the set of rules governing the format and content of the conversation between a Web client and server

HTTP Communication – 1/2

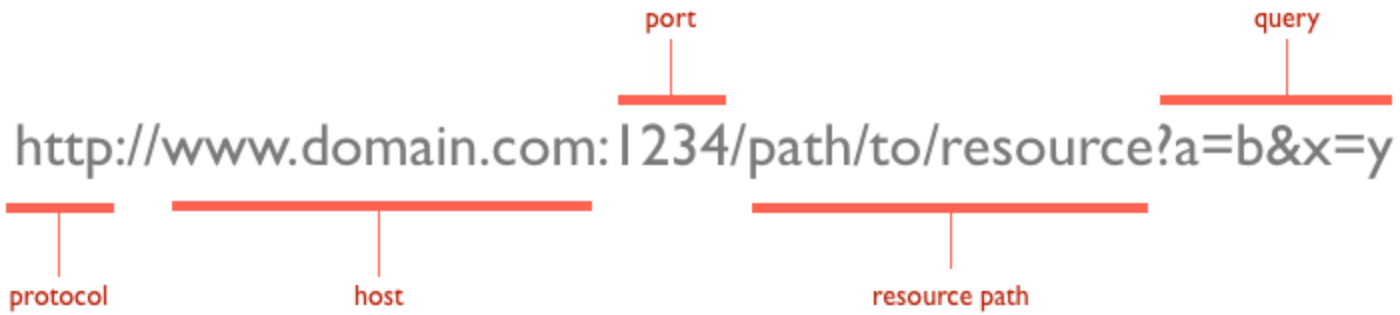
- It is a connectionless protocol
- The protocol is called connectionless because once the single request has been satisfied, the connection is dropped.
- It greatly simplifies the server construction and relieves it of the performance penalties of session housekeeping

HTTP Communication – 2/2

- It is a stateless protocol
- After the server has responded to the client's request, the connection between client and server is dropped and forgotten.
- There is no "memory" between client connections.
- The pure HTTP server implementation treats every request as if it is brand-new

Uniform Resource Locators (URLs)

- At the heart of web communications is the request message, which are sent via URLs.



- The protocol is typically http, but it can also be https for secure communications.

- URLs reveal the identity of the particular host with which we want to communicate, but the action that should be performed on the host is specified via HTTP verbs

Verbs

- **GET**: *fetch* an existing resource. The URL contains all the necessary information that server needs to locate and return the resource.
- **POST**: *create* a new resource. It requests usually carry a payload that specifies the data for the new resource.
- **PUT**: *update* an existing resource. The payload may contain the updated data for the resource.
- **DELETE**: *delete* an existing resource

Verbs

- **HEAD**
  - Similar to GET, but without the message body.
  - **Usage:** generally to check if the resource has changed, via timestamps.
- **TRACE**
  - It is used to retrieve the hops that a request takes to round trip from the server.
  - This can be used for diagnostic purposes.
- **OPTIONS**
  - It is used to retrieve the server capabilities.
  - On the client-side, it can be used to modify the request based on what the server can support.

Status Codes

- **1xx: Informational Messages**
  - All HTTP/1.1 clients are required to accept the Transfer-Encoding header.

- **2xx: Successful**
  - **202** Accepted: the request was accepted but may not include the resource in the response.
  - **204** No Content: there is no message body in the response.
  - **205** Reset Content: indicates to the client to reset its document view.
  - **206** Partial Content: indicates that the response only contains partial content. Additional headers indicate the exact range and content expiration information.

Status Codes

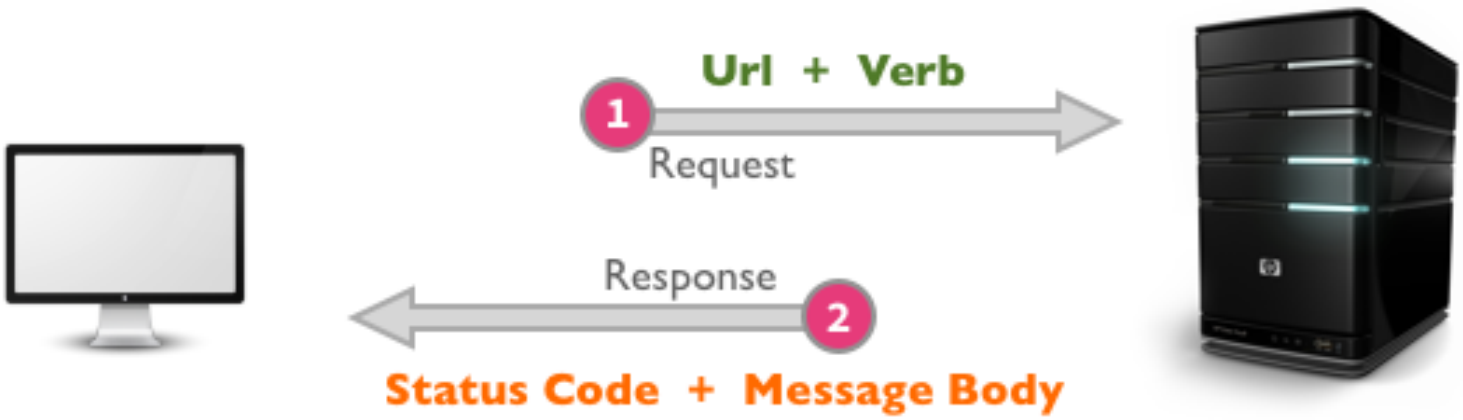
- **3xx: Redirection**
  - **301** Moved Permanently: the resource is now located at a new URL.
  - **303** See Other: the resource is temporarily located at a new URL. The Location response header contains the temporary URL.
  - **304** Not Modified: the server has determined that the resource has not changed and the client should use its cached copy. This relies on the fact that the client is sending ETag (Enttity Tag) information that is a hash of the content. The server compares this with its own computed ETag to check for modifications.

Status Codes

- **4xx: Client Error**
  - **400** Bad Request: the request was malformed.
  - **401** Unauthorized: request requires authentication. The client can repeat the request with the Authorization header. If the client already included the Authorization header, then the credentials were wrong.
  - **403** Forbidden: server has denied access to the resource.
  - **405** Method Not Allowed: invalid HTTP verb used in the request line, or the server does not support that verb.
  - **409** Conflict: the server could not complete the request because the client is trying to modify a resource that is newer than the client's timestamp. Conflicts arise mostly for PUT requests during collaborative edits on a resource.

- **5xx: Server Error**
  - **501 Not Implemented**: the server does not yet support the requested functionality.
  - **503 Service Unavailable**: this could happen if an internal system on the server has failed or the server is overloaded. Typically, the server won't even respond and the request will timeout.

Request and Response Message Formats



Message Structure

```
message = <start-line>
          *(<message-header>)
          CRLF
          [<message-body>]

<start-line> = Request-Line | Status-Line
<message-header> = Field-Name ':' Field-Value
```

- The message can contain one or more headers, of which are broadly classified into:
  - General headers: that are applicable for both request and response messages.
  - Request specific headers.
  - Response specific headers.
  - Entity headers.

Message Header

```
general-header = Cache-Control
               | Connection
               | Date
               | Pragma
               | Trailer
               | Transfer-Encoding
               | Upgrade
               | Via
               | Warning
```

Entity Headers

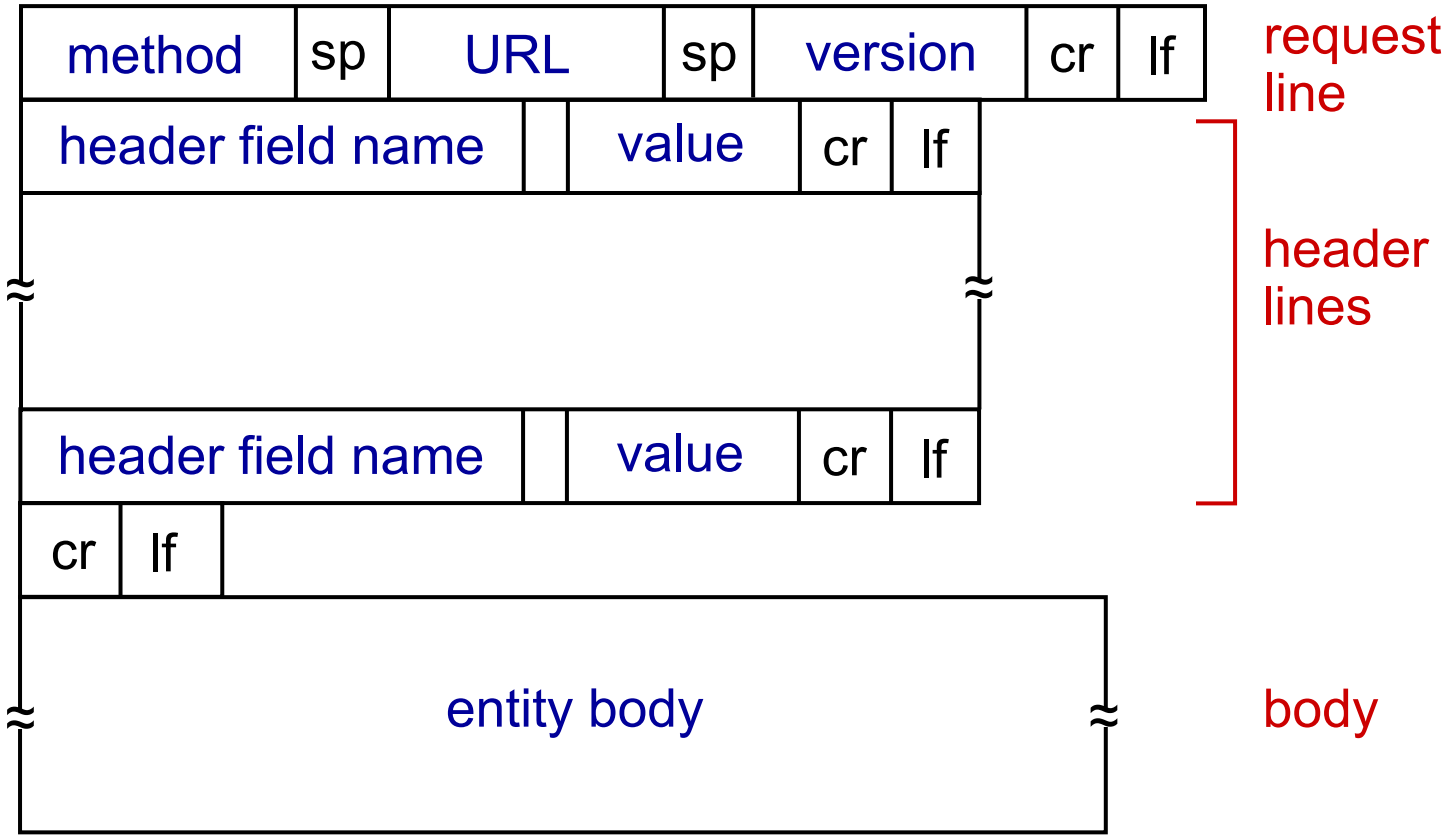
- Request and Response messages may also include entity headers to provide meta-information about the content

```
entity-header  = Allow
               | Content-Encoding
               | Content-Language
               | Content-Length
               | Content-Location
               | Content-MD5
               | Content-Range
               | Content-Type
               | Expires
               | Last-Modified
```

Request Format

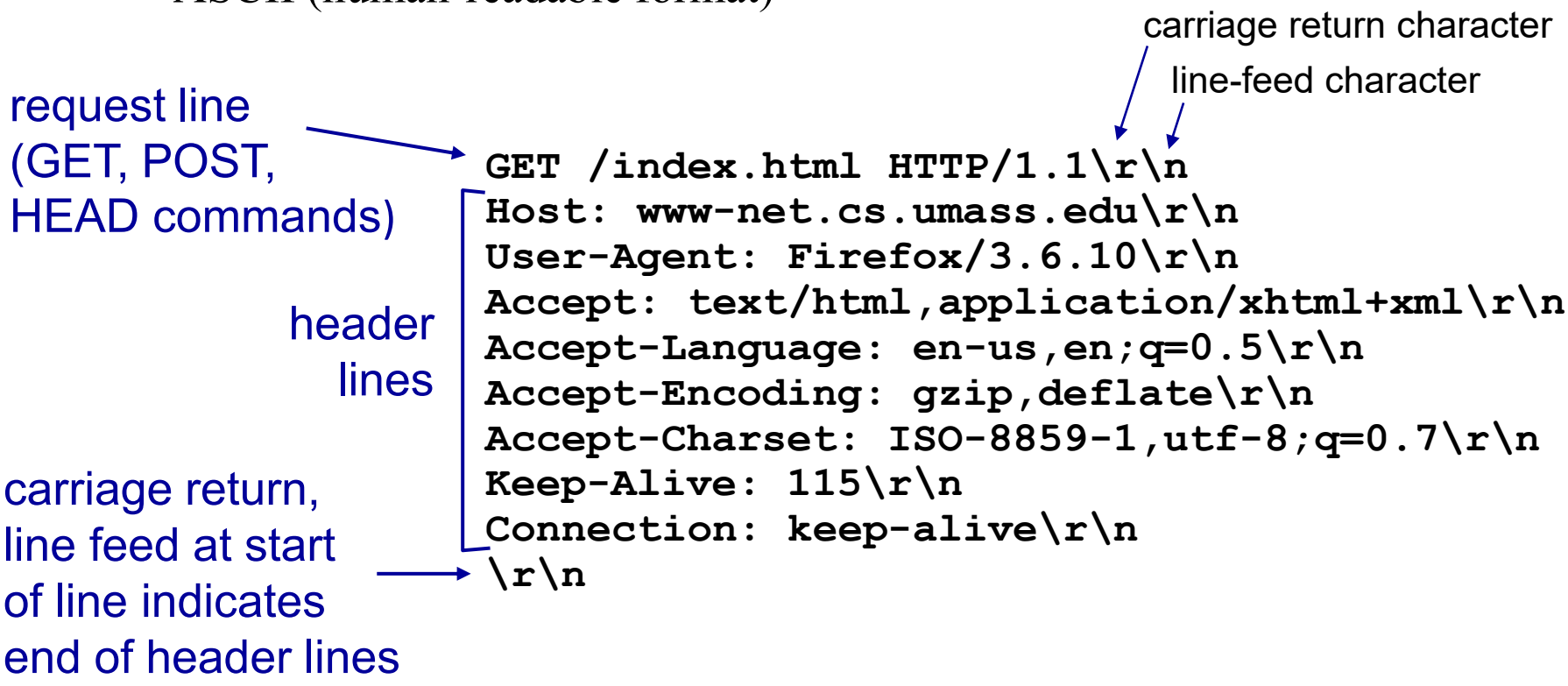
```
Request-Line = Method SP URI SP HTTP-Version CRLF
Method = "OPTIONS"
       | "HEAD"
       | "GET"
       | "POST"
       | "PUT"
       | "DELETE"
       | "TRACE"
```

HTTP Request Message: General Format



HTTP Request Message

- Two types of HTTP messages: *request*, *response*
- HTTP request message:
  - ASCII (human-readable format)



Response Format

```
Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF
```

- HTTP-Version is sent as HTTP/1.1
- The Status-Code is one of the many statuses discussed earlier.
- The Reason-Phrase is a human-readable version of the status code.

Example: It looks like:

```
HTTP/1.1 200 OK
```

HTTP Response Message

status line (protocol status code status phrase)

header lines

data, e.g., requested HTML file

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02
   GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```

Server-side Connection Handling

- The operations involve:
  - Establishing a socket to start listening on port 80 (or some other port)
  - Receiving the request and parsing the message
  - Processing the response
  - Setting response headers
  - Sending the response to the client
  - Close the connection if a Connection: close request header was found

HTTP/1.1 - The Next Generation

- The HTTP/1.1 (RFC 2616), is a replacement for HTTP/1.0
- It has higher performance and adding some extra features needed for use in commercial applications.
- It's designed to make it easy to implement the basic functionality needed by all browsers
- Additionally, more powerful features such as
  - Security and authentication much simpler.

HTTP/1.1 - The Next Generation

1. Persistent Connections

- A client and a server can signal the close of a TCP connection.
- This signaling takes place using the Connection header field

- Prior to persistent connections:
  - A separate TCP connection was established to fetch each URL
  - It increase the load on HTTP servers and causing congestion on the Internet.

HTTP/1.1 - The Next Generation

- **Persistent Connections (Advantages)**
  - By opening and closing fewer TCP connections, CPU time is saved in routers and hosts
  - Network congestion is reduced by reducing the number of packets caused by TCP opens (handshakes' cost)
  - Latency on subsequent requests is reduced since there is no time spent in TCP's connection opening handshake.
  - It can evolve more gracefully, since errors can be reported without the penalty of closing the TCP connection.

HTTP/1.1 - The Next Generation

2. Request Pipelining

- Allows a client to send several requests without waiting for a response
- Server responds in the same order

3. Chunked Encoding

- Allows sender to break a message into arbitrary sized chunks
- Useful for dynamically created response messages

SSL: Secure Web Communications

- *SSL protocol is application independent
- Operates between application layer and transport layer
- Application protocols such as HTTP sit on top of it and TCP/IP beneath it

*Secure Sockets Layer are cryptographic protocols designed to provide communications security over a computer network.

Ensuring SSL version compatibility

- There are different versions of SSL depending on the encryption algorithm used.
- The browser sends the versions it supports
- The server sends the certificate. The certificate includes:
  - The identity of the organization to which the web server belongs
  - The certificate's expiration date
  - The public key
  - The identity of the organization that issued the certificate, known as a certification authority (CA)
- Browsers store and recognize certificates issued by a number of well-known CAs.

Acknowledgements

- Most part of this tutorial was prepared by M.Fahim, G.Succi, and A.Tormasov
- Some slides are prepared by A. Burmyakov

Reference

- This tutorial is based on the following documents as well as lecture materials over the internet.
  - http://condor.depaul.edu/dmumaugh/readings/handouts/SE435/HTTP/http.pdf
  - https://code.tutsplus.com/tutorials/http-the-protocol-every-web-developer-must-know-part-1--net-31177