**Internetwork**

\*Inter - outside the boundaries

\*network – global information system; interconnecting devices using undersea cables or satellites

Interconnection technologies: wired and wireless

**IPV4** – 32-bit; 4 billion unique addresses

**IPV6** – 64-bit addresses

**Internet –** uses TCP; backbone of higher level services

* **1969**: DARPA – packet/switch network circuit: establishing a circuit and maintaining it while communicating; store and forward
* **1972**: e-mail- first to harness the internet; stand-alone programs
* **1989**: WWW – Created by Tim Berners-Lee in CERN

**World Wide Web –** HTTP, HTML, URL : addressing mechanism

* Web server and web client- applications
* Documents connected to other documents by links etc.
* **Web servers**: hosts web resources; listen for requests for specific resources
* **Web client**: browsers etc.

**\*WAIS (Wide Area Information Servers)** – query different data and index them

**\*Gopher(protocol)** – hierarchies of information; main index sub-index

**\*Usenet** – online discussion groups/bulletin boards

**HTTP** – standard way of communication over the web; information are textual

* connection, request, response, disconnection
* TCP/IP Port 80
* Client-server architecture
* **HTTP 1.0** (RFC 1945, May 1996), **HTTP 1.1** (RFC 2068 Jan 1997, RFC 2616 Jun 1999, RFC 7230-72355 Jun 2014), **HTTP 2** – SPDY (RFC 7540 May 2015)
* **Clients/user agents**: web browsers, crawlers (ex: GoogleBot)
* **Servers**: origin server; web resources are physically there; machine with an application running
* **Proxy**: checks authentication; forward requests and give back requests
* **Tunnel**: forwards the communication without knowing what it does
* HTTP uses a request-response standard protocol: **“pull protocol”** (client sends request to server), **“push protocol”** (server volunteers new information via notifications), **“polling”** (checks server at regular intervals).
* HTTP is a stateless communications protocol; servers do not keep information about clients
* **Cache**: stores in a location to be retrieved again
* **MIME**: Multipurpose Internet Mail Extension; tells what kind of resource you are sending
* **URI** (Uniform Resource Identifier): **URN** (Uniform Resource Name), **URL** (Uniform Resource Locator)
* **IANA**: allocates certain ports to certain applications
* **Socket**: IP Address and Port number
* **Link Rot**: broken links
* **SSL/TLS**: takes HTTP traffic and transmits it in an encrypted form HTTPS
* **Digital certificates** and **self-signed certificates**: to secure sites using HTTPS
* **Application layer communications protocol**: used to access web resources; Tim Berners-Lee

**REQUEST MESSAGES**

* **Request Line** – Method – extension methods
* Request URI
* HTTP Protocol Version
* **Message Headers** – contain additional information of the client’s request (optional in HTTP 1.0)

- terminated by empty line (CRLF)

**\*General Header Fields** = client and server

\***Request Header Fields** = client

\***Response Header Fields** = servers

\***Entity Header Fields** = describe payload

**HTTP RESPONSE MESSAGES**

* **Status Line** – HTTP Protocol Version
* **Status code**: Informational (1xx), Success (2xx), Redirection (3xx), Client Error (4xx), Server Error (5xx)
* **Reason Phrase**: Not Found, Forbidden etc.
* **Message Headers**: additional information about response: with response headers

**HTTP REQUEST METHODS** – standard

* **GET** – transfer current selected representation of the resource identified by the Request URI; most commonly used HTTP method.
* **HEAD** – body of response is not sent; used to retrieve metadata
* **POST** – submit; send something to the server
* **PUT** – send an entity in the message body
* **DELETE** – remove the resource
* **DAV** – Distributed authoring and versioning
* **TRACE**- request a loopback: send back a request
* **CONNECT** – exchange of credentials

**SAFE METHODS** – doesn’t affect/change resource

**IDEMPOTENT METHODS** – repeated execution should result in a same effect

**CACHEABLE METHODS** – allows it to be cached

**GENERAL HEADER FIELDS**

* **Cache-control**
* **Connection header** – control whether a connection is persistent or not
* **Pragma**- generic directive; indicate that something is cacheable
* **Upgrade**- for protocol negotiation