<https://byjusexamprep.com/application-layer-protocols-dns-smtp-pop-ftp-http-i>

Process - program running within a host.

* Within same host two processes communicate using inter-process communication (defined by OS)
* Processes in different hosts communicate by exchanging messages.
* **Client process:** process that initiates communication
* **Server process:** process that waits to be contacted
* Note: note applications with P2P architectures have client processes & server processes.

Sockets - process sends/receives messages to/from its socket.

* Socket analogous to door
  + Sending process shoves message out of the door
  + Sending process relies on transport infrastructure on other side of the door to deliver message to socket at receiving process.

Addressing Processes

* To receive messages process us thave identifier
* Host device has unique 32bit IP address.
* Many processes can be running on the same host therefore having the same IP address.
* Identifier:
  + IP address and port numbers associated with process on host.
  + Ex: port numbers:
    - HTTP server: 80
    - Mail server: 25

App-layer protocol defines

* Types of message exchanged: Request or Response
* Message syntax: what fields in message and how fields are delineated.
* Message semantics: meaning of information in fields
* Rules: for when and how processes send & respond to messages
* Open protocols
  + Defined in RFCs, everyone has access to protocol definition
  + Allows for interoperability
  + Ex: skype

**4 broad categories of internet transport protocol services**

**Data integrity** some apps (e.g., file transfer, web transactions) require 100% reliable data transfer other apps (e.g., audio) can tolerate some loss

**Timing**  some apps (e.g., Internet telephony, interactive games) require low delay to be “effective”

**Throughput**  some apps (e.g., multimedia) require minimum amount of throughput to be “effective” other apps (“elastic apps”) make use of whatever throughput they get

**Security** encryption, data integrity

HTTP: hypertext transfer protocol

* Web’s application layer protocol
* Client/server model
  + Client: browser that requests,receives, (suing http rptocol) and displays web objects
  + Server: web server sends (using http protocol) objects in response to requests.
* Uses TCP:
  + Client initiates TCP connection(creat socket) to server, port 80
  + Server accepts TCP connection from client
  + HTTP messages exchanged between browser and web server
  + TCP Closes
  + HTTP is stateless: server maintains no information about past client requests.
* Non-persistent HTTP - at most one object is sent over TCP connection and then the connection will close. If you want to download multiple objects, multiple connections are required.
* Persistent HTTP - multiple objects can be sent over a single TCP connection between the client and server before the TCP is closed.

RTT - round trip time - time for a small packet to travel from client to server and back.

HTTP response time:

* One RTT to initiate TCP conenction
* One RRT for HTTP request and first few bytes of HTTP response to return.
* File transmission time.
* Non-persistent HTTP response time = 2RTT + file transmission time.

**non-persistent HTTP issues**: requires 2 RTTs per object OS overhead for each TCP connection browsers often open parallel TCP connections to fetch referenced objects

**persistent HTTP:** server leaves connection open after sending response subsequent HTTP messages between same client/server sent over open connection client sends requests as soon as it encounters a referenced object as little as one RTT for all

HTTP request message:

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

Order of Messages:

1. Request line: (Get,POST, HEAD commands)
2. Header Lines
3. Carriage return: line feed at start of line indicates end of header lines.

POST method:

* Web page often includes form input
* Input is uploaded to server in entity body

URL method:

* Uses GET method
* Input is uploaded in URL field of request line.

HTTP Response message:

Order:  
 1. Status line(protocol status code phrase)

2. Header liens

3. Data : e.g. requested HTML file

Status Codes:

* 200 OK • request succeeded, requested object later in this msg
* 301 Moved Permanently • requested object moved, new location specified later in this msg (Location:)
* 400 Bad Request • request msg not understood by server
* 404 Not Found • requested document not found on this server
* 505 HTTP Version Not Supported

User-servers State: COOKIES

Four Components

1. Cookie header line of HTTP response message
2. Cookie header line in next http request message
3. Cookie file kept on user’s host, managedby user’s browser
4. Back-end database at web site

**Web cache (proxy server)** : goal is to satisfy client request without involving origin server

1. User sets browser: web access via cache
2. Browser sends all HTTP requests to cache
   1. Object in cache: cache returns object from origin
   2. Else cache requests object from origin server, then return object to client.

* Cache acts as both a client and server
  + Server for original requesting client
  + Typically cache is installed by ISP (university,company,residential ISP)
* Why?
  + Reduce response time for client request
  + Reduce traffic on an institution’s access link
  + Internet dense with caches: enables “poor” content providers to deliver content (so too does P2P file sharing)