Internet

* global network of networks
* from internetwork
* inter– outside the boundary
* network – interconnection of devices
* categorized by:
* nodes – actual devices
* interconnection technology –
* wired - ex. coaxial ,fiber, UTP
* wireless- ex. infrared
* protocols
* device drivers
* linked together based on IP
* supports communications using TCP/IP
* IPv6 -64 bit
* IPv4-32 bit
* uses or makes accessible high level services
* Initially intended for scientific and military use.
* 1969 – beginning of the internet
* DARPA (Defense Advanced Research Project Agency)
* IoT- Internet of things
* 1972 – email was developed (first killer application)

Circuit Switch Connection

* Establish circuit and maintain it to communicate
* Not scalable

Store and Forward

* Send messages to the next hub until destination is reached

Wide Area Information Services (WAIS)

* Connected to servers from different locations to get data from each server periodically and index these data

Gopher Protocol

* Distributing, searching and retrieving documents over the internet
* Hierarchy is involved with the index having sub-indexes

Usenet

* Similar to an online discussion group

World Wide Web (WWW)

* Established in 1989
* Created by Tim Berners-Lee
* Worked with CERN
* HTTP, HTML, URL
* Web server and web client
* Information system that allows documents to be connected to other documents
* An arrangement of web servers that boost particularly designed records
* Two most popular browsers people use is the Google Chrome and the Mozilla Firefox
* There are a few applications called Web programs that make it simple to get to the World Wide Web
* All significant Web locales have balanced their substance outline and improvement way to deal with oblige the quickly expanding division of the populace getting to the Web from little screen telephones rather than extensive screen desktop and smart phones

Web Server

* Hosts web resources
* Listens for requests

Web Client

* Example is web browser
* Gives requests to web servers
* Follows HTTP to communicate with the server

Semantic Web

* Next step in the evolution of the web
* Makes use of artificial intelligence to understand the question given by user

Hypertext Transfer Protocol (HTTP)

* Application layer communications protocol used to access resources
* Developed by W3C (World Wide Web Consortium) and IETF (internet Engineering Task Force)
* Standard way of communicating through applications
* Version 0.9 came out in 1991
* HTTP 1.0 (RFC 1945, May 1996) – first standardized
* HTTP 1.1 (RFC 2068 Jan 1997, RFC 2616 Jun 1999), RFC 7230 – 7235 (Jun 2014)
* HTTP 2 (RFC 7540 May 2015) – patterned after SPDY of Google

HTTP Fundamentals

* Runs on top of TCP/IP, using port 80 as default or 443 for HTTPS
* HTTPS is encrypted and needs digital or self-signed certification
* IANA – allocates certain ports to certain applications
* Based on client-server architecture
* Clients a.k.a. user agent
* Web browsers, web crawlers/spiders, other end user tools and applications
* Any application that communicates with HTTP protocol
* Server
* Origin server – resources are actually there
* Proxy server, gateway, tunnel (blind relay between two points) – can be used for authentication
* Uses request-response protocol
* Client sends an HTTP request message to server (pull protocol)
* Server processes the request and replies with HTTP response message
* In HTTP 2, server can push resources to client without client requesting
* Stateless communication
* Do not keep information about clients in between requests
* Other functionalities
* Cache control – storage for easy and fast access
* Content media type (MIME) specification
* Language and character set specification
* Content/ transfer coding
* Content negotiation – talk to tell what the recipient can handle
* Client-server protocol negotiation – asking server if it can handle higher version and if yes, the server will upgrade
* Persistent connections – telling the server to not close the connection for further requests
* In HTTP 1.0, connection is terminated after a resource has been retrieved from responding to a request
* Request pipelining – sending requests one after another
* In HTTP 1.0, only one resource can be transferred per connection
* Authentication/ authorization

HTTP Resource Accessing

* HTTP resources are identified using URIs, which tells what the resource is, or more, specifically HTTP URL, which tells where the resource is
* Scheme (http or https)
* Authority
* User information/ information credentials
* Host-domain name (resolved to an IP address using DNS) of the server where the resource resides, or will be created
* Port number – default is 80
* Path to resource (resolved relative to the document root) – may refer to a static or dynamic resource
* Query – typically provided as key = value pairs, with ampersand separators between key/ value pairs, and may be URL-encoded
* Fragment identifier – “bookmark”

Absolute URL

* Scheme and domain name are always required

Relative URL

* Scheme, user info, and domain name can be omitted

HTTP Request Message

* Request Line (CRLF – terminated line consisting of three spaced-separated items)
* Request Method
* GET
* HEAD
* POST
* PUT
* DELETE
* CONNECT
* OPTIONS
* TRACE
* Request URI – location of the requested resource
* HTTP protocol version
* Request Header Fields – information about the request and the client
* Accept-Charset
* Accept-Encoding
* Accept-Language
* Authorization
* Expect
* From
* Host
* If-Match
* If-none-match
* Range
* If-range
* If-modified-since
* If-unmodified-since
* Max-forwards
* Proxy-authorization
* Referer
* TE
* User-agent
* Message headers
* HTTP 1.1 requires at least the host request header to be provided
* Empty line (CRLF)
* Message body a.k.a. payload – optional

HTTP Response Message

* Response Header Fields
* Accept-Ranges
* Age
* ETage
* Location
* Proxy Certificate
* Retry-After – when maintenance of a web server; timeouts
* Server – displays what server
* Vary – changing header values
* www-Authentication
* Status line (CRLF)
* HTTP protocol version
* Status code – 3-digit code that designates the status
* Reason phrase – descriptive meaning og the status code
* 1xx (info), 2xx (success), 3xx (redirection), 4xx (client), 5xx (server)
* Message headers
* Empty line
* Message body – optional

HTTP Request Methods

* Standard
* GET
* Transfer a current selected representation of the resource identified by the request URI
* Requests data from a specified resource
* Most commonly used method
* Must be supported by all compliant general-purpose servers
* Can be stored
* Can be bookmarked
* Must never be utilized when managing delicate information
* Are ought to be utilized just to recover information
* Asks for have length limitations
* Remains in the browser history
* HEAD
* Same as GET but the entity is not sent
* Used to retrieve metadata about the entity
* Must also be supported by all
* POST
* Performing resource-specific processing of entities enclosed in the message body
* Submits data to be processed to a specified resource
* Has a payload
* Are never cached
* Cannot be bookmarked
* Do not remain in the browser history
* No restrictions on the data length
* PUT
* Store the enclosed entity in the message body under a specified URI
* By default, the method is not allowed
* When allowed, authentication must be provided
* Uploads a representation of the specified URI
* DELETE
* Remove the resource associated with the specified URI
* Like PUT, the method is not allowed by default
* OPTIONS
* Request information on what can be done with the resource specified
* Returns the HTTP methods that the server supports
* TRACE
* Perform or request a loopback of the requested message (echo back)
* Typically used for testing/diagnosis of the request/response chain
* Allows the customer to perceive what is being gotten at the flip side of the demand chain and use the data for testing
* CONNECT
* Establishment of an encryption tunnel to communicate with https
* Converts the request connection to a transparent TCP/IP tunnel
* Extension Methods
* WebDAV
* PROPFIND - method recovers properties characterized on the asset recognized by the Request-URI
* PROPPATCH – method forms guidelines indicated in the demand body to set and additionally or expel properties characterized on the asset recognized by the Demand URI
* MKCOL – make collection (new folder)
* COPY
* MOVE
* LOCK – lock resources
* UNLOCK – unlock resources
* Safe Methods
* Doesn’t change the resources
* Method that can be cached
* GET, HEAD, OPTIONS, TRACE
* Idempotent Methods
* Repeated request result to the same response
* GET, HEAD, OPTIONS, TRACE, PUT, DELETE
* Cacheable Methods
* Requests that generate cache
* GET, HEAD are cacheable

HTTP Message Headers

* General Header Fields(client & server)
* Cache-control – controlling cache and telling whether a resource need to be cached or not
* Connection – control whether connection is persistent or not
* Date – when the request was generated
* Pragma – from older version, generic directive
* Trailer – header at the end of the data
* Transfer-encoding
* Upgrade – for protocol negotiation
* Via – path, where the request passed through
* Warning – error in the message
* Request Header Fields (client)
* Accept – specifying acceptable file type; server responds with negotiation, accept if other type can be delivered
* Accept-Charset
* Accept-Encoding
* Accept-Language
* Authorization
* Ensure protection from unauthorized access
* When proper credentials are entered, another request will be sent with authorization
* Without authorization, www-authenticate challenge will be sent and a pop-up authorization will appear
* Proxy Authorization – for proxy server
* Expect – for two-face connection
* From – contact credentials of sender
* Host – required in HTTP 1.1
* If-Match – comparing the file in the cache and server to check if file is unmodified to avoid state entity.
* If
* If-Range – “is what I have a portion of what you have”
* If-Unmodified-Since
* Ranges – Specifying portions of the resource the client wants to receive
* Max-Forwards – for tracing and limiting a trace.
* Referer – where the request originated; to check where traffic is coming from
* TE – trailer encoding
* User-Agent – info about the clients, identity
* Response Header Fields (server)
* Accept-Ranges – allows partial resources
* Age – how long ago the response has been generated
* ETag (entity tag)
* Location – for redirection. If present, the server will fetch the latest resource from location
* Proxy-Authenticate
* Retry-After – amount of time the client has to wait before trying again
* Server – information about the server
* Vary – for content negotiation
* Entity Header Fields
* Allow – methods that can be called
* Content-Encoding – when server changed form of entity
* Content-Language – language characteristics; for accessibility purposes
* Content-Length – size of payload
* Content-Location – when content is from somewhere else; actual location of the resource
* Content-MD5 (deprecated header)
* Content-Range – when doing range request
* Content-Type – MIME type
* Expires – for controlling cache access; cache-control; (e.g. max-age = ??)
* Last-Modified – possible reference value to check if resource is fresh

HTTP Status Codes

* Server Responses
* Informational (1xx)
* 100 – continue
* 101 – switching protocol
* Success (2xx)
* 200 – OK
* 201 – Created
* 202 – Accepted; request is accepted but not yet generated
* 203 – Non-Authoritative Information; some are not from the origin server.
* 204 – No Content; not returning anything but success
* 205 – Reset Content; client can reset forms and data entries
* 206 – Partial Content; in other words, this is not the whole resource.
* Redirection (3xx)
* 300 – Multiple Choices; when resource is available in different forms
* 301 – moved Permanently; resource is no longer here and location must be changed
* 302 – Found; when a new resource is generated and can be found somewhere
* 303 – See Other; same as 302
* 304 – Not Modified; cache and server resource are the same
* 305 – Use Proxy; access through proxy
* 306 – n/a (unused status code)
* 307 – Temporary Redirect; like 301 but location must not be changed
* Client Error (4xx)
* 400 – Bad Request; malformed request
* 401 – Unauthorized; accessing protected resource
* 402 – Payment Required; for online payments
* 403 – Forbidden; resource must not be accessed
* 404 – Not Found; resource might be there but the client made a mistake in typing
* 405 – Method Not Allowed
* 406 – not Acceptable; server won’t do it
* 407 – Proxy Authentication Required
* 408 – Request Time-out; request taking too long to process the request
* 409 – Conflict; request can’t be done because state of resources can be compromised
* 410 – Gone; no forwarding address, no longer there, purposely removed
* 411 – Length Required; specify content length
* 412 – Precondition Failed; condition has failed
* 413 – Request Entity Too Large; payload is too big
* 414 – Request-URI Too Large; URI is too long
* 415 – Unsupported Media Type
* 416 – Request Range not Satisfiable
* 417 – Expectation Failed; server cannot honor request
* 426 – Upgraded Required (recently introduced status code)
* Server Error (5xx)
* 500 – Internal Server error; server’s fault, misconfiguration
* 501 – Not Implemented; method can’t be handled
* 502 – Bad Gateway; gateway machine is having problem
* 503 – Service Unavailable; when server is under maintenance
* 504 – Gateway Timeout
* 505 – HTTP Version not Found

If ever need ng definition sa mga header field terms, pwede dito :D :)

HTTP 1.1 RFC 2616 : Header Field Definitions

https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html