Web Protocols

HTTP 1.0/1.1

HTTP 0.9: The One-Line Protocol

- The original HTTP proposal by Tim Berners-Lee (1989).

- High-level design goals
  - file transfer functionality
  - ability to request an index search of a hypertext archive
  - format negotiation.
  - an ability to refer the client to another server

HTTP 0.9: The One-Line Protocol

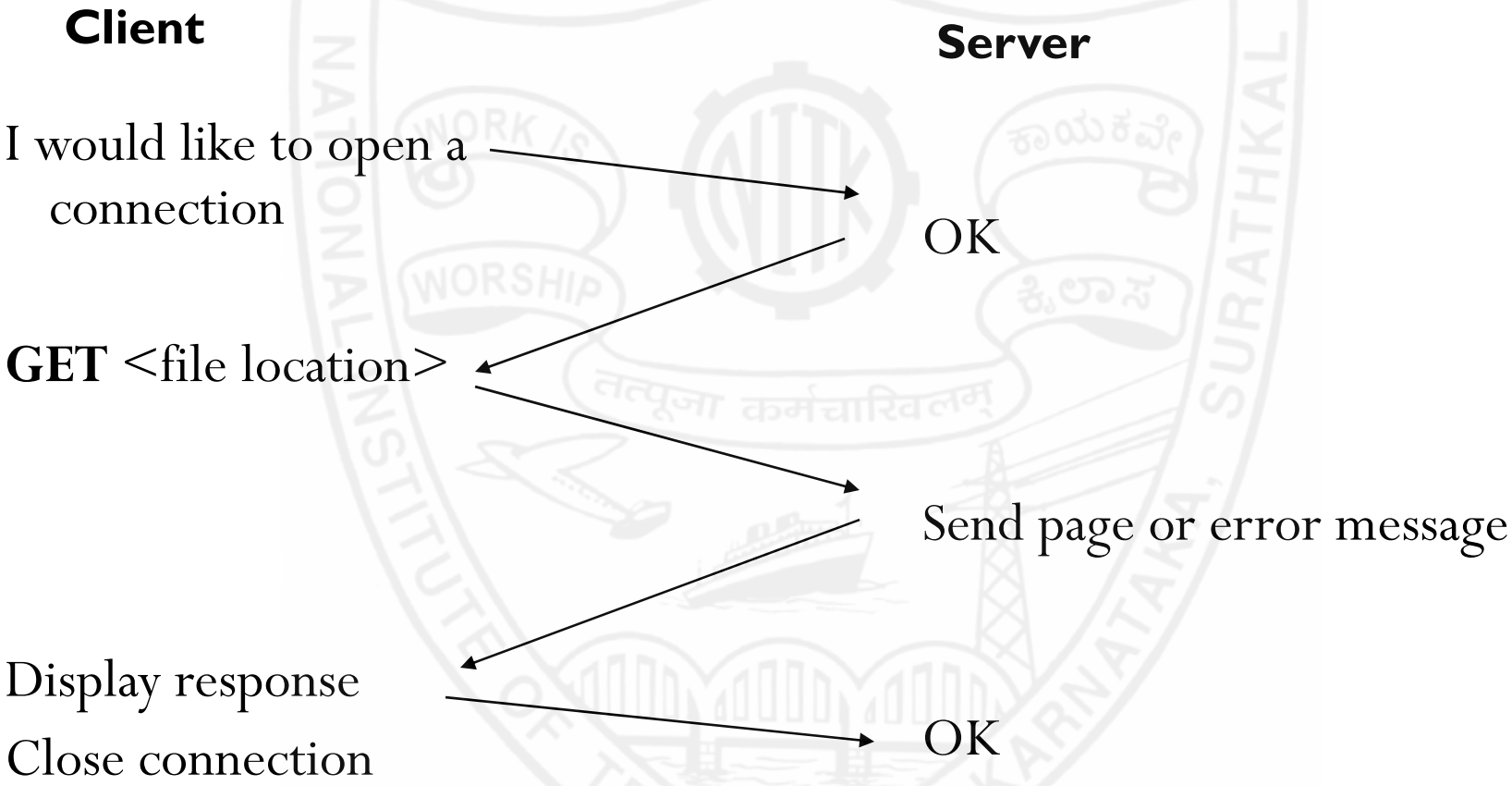
- Implementation:
  - Client request is a **single** ASCII character string
    - terminated by a carriage return (CRLF).
  - Server response is **an ASCII character stream.**
  - Server response is in **hypertext only**
  - Connection **is terminated after the document transfer** is complete.

Hyper Text Transfer Protocol (HTTP)

- Works in a client/server computing environment.
  - Runs on top of TCP on the standardized port 80;
  - Stateless
  - Asynchronous

- Based on *message based interoperability* (request-response model).
  - Request is specified in ***text (ASCII)*** format
  - response is in ***MIME/IMT format***

A HTTP conversation



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

Hyper Text Transfer Protocol (contd.)

How HTTP works

- Address entered - www.example.org
- Browser functions –
  - Creates a message conforming to HTTP protocol (***HTTP Request***),
  - Uses DNS to obtain the IP address for www.example.org,
  - Creates a TCP connection with the machine at this IP address,
  - Sends the HTTP request message over this TCP connection.
  - Server sends back result (***HTTP Response***).
  - Browser deciphers the response and creates a display of the information in the client area of the browser. (***Rendering***)

Hyper Text Transfer Protocol (contd.)
HTTP Request

- **General format of client request.**

Syntax

Request_method Resource_address HTTP/version-no

General_header(s)

Request_header(s)

Entity_header(s)

------ blank-line--------

Message body (Additional Data) (optional)

Hyper Text Transfer Protocol (contd.)
HTTP Request (Line 1)

- Basic Request methods (introduced in HTTP 1.0)
  - GET - *retrieve a webpage reached by the given Request-URL*
  - POST - *posts additional data to the web server appending it to the HTTP request msg.*
    - Data is attached after the headers.
  - HEAD - *requests the header info of the webpage for a given Request-URL*

Property	GET	POST
BACK button/ Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

Hyper Text Transfer Protocol (contd.)

- Additional request methods introduced in HTTP 1.1 and beyond…
  - **PUT** – request that the enclosed entity be stored under the Request-URL
  - **DELETE** – request that the server delete resource identified by Request-URL
  - **OPTIONS** – request for information about communication options.
  - **TRACE** – invoke a remote, application-layer loopback of request message
  - **CONNECT** – used by proxies in SSL connections.
  - **PATCH** – applies partial modifications to a resource

Comparing PUT and POST

PUT	POST
Replacing existing resource or Creating if resource is not exist *http://www.example.com/customer/{id}* *http://www.example.com/customer/123/orders/456* Identifier is chosen by the client	Creating new resources(and subordinate resources) *http://www.example.com/ customer/* *http://www.example.com/customer/123/orders* Identifier is returned by server
Idempotent i.e. if you PUT a resource twice, it has no effect. Ex: Do it as many times as you want, the result will be same. x=1;	POST is neither safe nor idempotent. It is therefore recommended for non-idempotent resource requests. Ex: x++;
Works as specific	Works as abstractive
If you create or update a resource using PUT and then make that same call again, the resource is still there and still has the same state as it did with the first call.	Making two identical POST requests will most-likely result in two resources containing the same information.

Safe and Idempotent Properties

Safe HTTP method – method that does not modify resources.

Idempotent HTTP method – can be called many times without any change in associated outcomes.

HTTP Method	Idempotent	Safe
OPTIONS	yes	yes
GET	yes	yes
HEAD	yes	yes
PUT	yes	no
POST	no	no
DELETE	yes	no
PATCH	no	no

Hyper Text Transfer Protocol (contd.)
HTTP Request (Line 1)

- ***Resource_address:*** is the URL that specifies location of the requested resource on web server.

- ***HTTP/version-number:*** Tells the web server what HTTP protocol the web client is using (1.0/1.1/2)

Hyper Text Transfer Protocol (contd.)
HTTP Request - *HTTP Headers*

- Used for passing additional info to the web server.

- Three different types:
  - General Header
  - Request Header
  - Entity Header

Hyper Text Transfer Protocol (contd.)
HTTP Request - *HTTP Headers*

1. *General Headers*

- *Date* - specifies when message is generated *(date and time).*
- *Pragma* - used for specifying freshness of resource to the server and to the intermediaries like proxy servers.

e.g. pragma : no-cache *(in HTTP1.0)*

Cache-Control: no-cache *(in HTTP 1.1 and later)*

Hyper Text Transfer Protocol (contd.)
HTTP Request - *HTTP Headers*

2. *Request Headers*

- *Authorization* - provides authorization info to web server.
- *From or host* - provides information about the source of the HTTP message.
- *If-modified-since* - asks the web server to provide requested resource only if it has been modified since the specified time in the header.
- *Referer* - the URL from which the client arrived at the resource.
- *User Agent* - provides information on user agent (browser) used by web client.

Hyper Text Transfer Protocol (contd.)
HTTP Request - *HTTP Headers*

2. Request Header (contd.)

- some optional additional fields:
  - *Accept* - MIME types of resources accepted by browser
  - *Accept-Charset* - charset accepted by browser
  - *Accept-Encoding* - encoding accepted by browser
  - *Accept-Language* - language accepted by browser

HTTP 1.1 additions

Hyper Text Transfer Protocol (contd.)
HTTP Request - *HTTP Headers*

- ***Entity headers:*** define meta-information about the entity-body or, if no body is present, about the resource identified by the request.
  - *Allow* - indicates request methods allowed.
  - *Content_encoding* - specifies compression method applied to content.
  - *Content_length* - indicates length of content in no. of octets.
  - *Content-range* - *for range requests (HTTP 1.1 only)*
  - *Content_type* - indicates MIME type of content.
  - *Expires* - indicates date and time of content expiry.
  - *Last_modified* - indicates when webpage was last modified.

Hyper Text Transfer Protocol (contd.)
HTTP Request (Last line)

- **Additional Data:**
  - web client can post additional data to the server after blank line.
    - In case of POST/PUT/DELETE requests

Hyper Text Transfer Protocol (contd.)
HTTP Request

Simple Example for a HTTP Request –

```
GET /index.html HTTP/1.1
host: www.nitk.ac.in
user-agent: Mozilla/5.0 (windows; …)
accept: text/html, image/gif, image/jpeg, */*
```

Hyper Text Transfer Protocol (contd.)

Real-world Example for a HTTP Request –

```
GET /depts HTTP/1.1
Host: www.nitk.ac.in
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.3) Firefox/3.5.3
Accept: text/html, application/xhtml+xml, application/xml, text/* , */*
Accept-Language: en-us, en-us, en; q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7
Connection: keep-alive                    //only in HTTP 1.1 and HTTP/2
Keep-Alive: 300
Allow: GET, HEAD, POST
Cookie:PREF=ID=141ca2d1746b4:U=f22e9e944a56f:FF=4:LD=en:NR=10:CR=2:TM=124956
    7334:LM=1251146058:GM=1:S=qWowBrte7hrXniGp;
    NID=27=n9Khexo85YHnovw93wK4qC2lZtGa1DnzVQEB6iul9tn62fsJ7gFuMVK252ceLC
    D3iS54r-nHD6kWDdD1JP77akDhMl0EWzoPt3cM5g8mapG9SskdRSyEyLWcJK1LrX
Cache-Control: max-age=0          //only in HTTP 1.1 and HTTP/2
```

Hyper Text Transfer Protocol (contd.)
HTTP Response

- General format of the Web server's response to a HTTP request.

Syntax

HTTP/version-number status-code result-message (status line)

General_header(s)

Response_header(s)

Entity_header(s)

------ blank-line--------

Entity body (optional)

Hyper Text Transfer Protocol (contd.)
HTTP Response (line 1)

- Status Code:
  - indicates the result of the request. (e.g. not found, unauthorized, O

Code	Class	Standard Use
1xx	Informational	Provides information to client before request processing is completed
2xx	Success	Request has been successfully completed
3xx	Redirection	Client needs to use a different resource to fulfil request
4xx	Client Error	Client's request is not valid
5xx	Server Error	An error occurred during server processing of a valid client request.

Code	Meaning
200	OK
201	Created
204	No Content
206	Partial Content
301	Moved Permanently
302	Moved Temporarily
400	Bad Request
401	Unauthorised
403	Forbidden
404	Not Found
500	Internal Server Error
502	Bad Gateway
504	Gateway Timeout

Hyper Text Transfer Protocol (contd.)
HTTP Response – *Response Headers*

- ***Accept, accept-charset, accept-language, accept-encoding*** *(same as in HTTP request)*
- ***Accept-Ranges*** - server indicates its acceptance of range requests/byte serving for resource. (in HTTP 1.1)
- ***Age*** - time elapsed since response was generated by server.
- ***Location*** - redirect the client to a location other than Request-URL for completion of the request.
- ***Retry-After*** - indicate to client how long the service is expected to be unavailable.
- ***Server*** - information about software used by the server to handle the request.

Hyper Text Transfer Protocol (contd.)
HTTP Response *(line 4)*

- ***Entity headers:***
  - *Location* - provides new URL if content has been moved to new location.
  - *Server* - provides info on server.
  - *www-authenticate* - used to provide authorization info.

- ***Entity Body:***
  - The response data is enclosed as the entity body (usually a hypertext file.)

Hyper Text Transfer Protocol (contd.)

A sample HTTP Response

```
HTTP/1.1 200 OK
Date: Wed, 15 Jul 2016 14:37:12 GMT
Server: Microsoft-IIS/2.0
Content-Type: text/html, charset=UTF-8
Content-Encoding: gzip
Last-Modified: Fri, 10 Jul 2019 08:25:13 GMT
Server:gws
Content-Length: 3667
Allow: GET,POST,OPTIONS,HEAD
Cache-Control: max-age=60
Connection: keep-alive //only in HTTP 1.1 and HTTP/2
```

HTTP 1.1 Enhancements

a) ***Persistent Connections and pipelining:***

- connection is kept open such that web client can send multiple requests over the same connection.
  - Helps handle *TCP Slow Start*, in a better way.
  - Support for *Pipelining*

Connection: keep-alive (15s, 60 s, 300s …. Depending on browsers)

Connection: Upgrade

HTTP 1.1 (contd.)

b) Range Request

- allows a web client to retrieve part of the file by using the Range header.
  - *Accept-ranges* and *Content-range* headers for facilitating byte serving.

HTTP 1.1 (contd.)

c) Cache Control

- Purpose of caching is to shorten retrieval time of web pages.
  - HTTP1.0
    - supports only basic caching control (*If-Modified-Since* and *Pragma*)
  - HTTP1.1
    - provides separate *Cache-control* response headers.
    - Also provides conditional headers like *If-Unmodified-Since, If-Match, If-None-Match*.

HTTP 1.1 (contd.)

d) Connection Header

- data is sent in a series of "chunks".
- Sender does not need to know the length of the content before it is sent (servers can begin transmitting dynamically-generated content)
  - HTTP 1.0: *Content-length* header
  - HTTP 1.1: *Transfer-encoding* header

HTTP 1.1 (contd.)

e) New Status Codes

- **100 Continue:** The client SHOULD continue with its request.
- **101 Switching Protocols:** indicates change in protocol due to *Upgrade* message header sent by client.
- **407 Proxy Authentication Required**
- **416 Requested Range Not Satisfiable**

HTTP 1.1 (contd.)

f) Provision for additional request methods:

- PUT, DELETE, OPTIONS, TRACE, CONNECT, PATCH

g) Better support for data compression

- *accept-encoding* header

h) Better support for languages

- *accept-language* header

i) Support for Proxy Authentication

More Reading

- Berners-Lee, Tim, Roy Fielding, and Henrik Frystyk. "Hypertext transfer protocol--HTTP/1.0." (1996).

- Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (1999). Hypertext transfer protocol–HTTP/1.1.

- RFC 2616 – Hypertext Transfer Protocol - HTTP 1.1
  https://tools.ietf.org/html/rfc2616