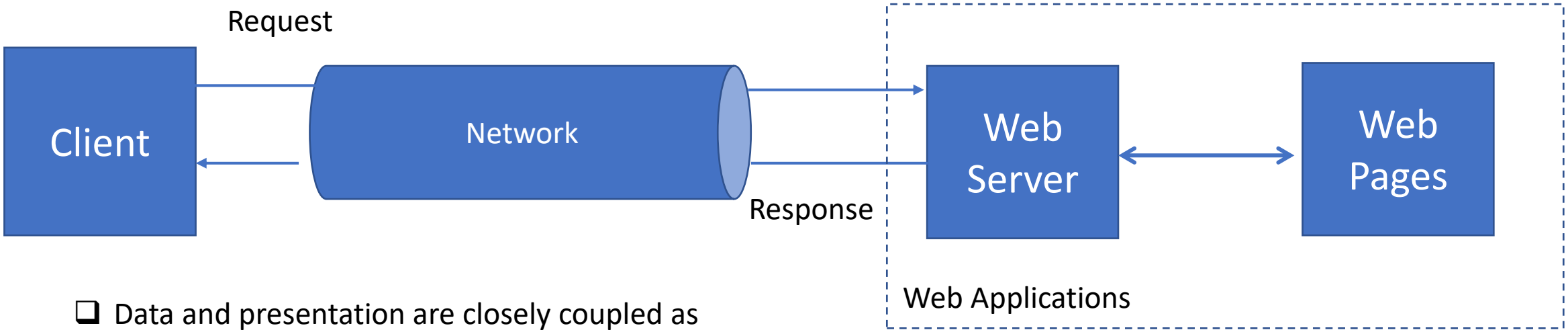
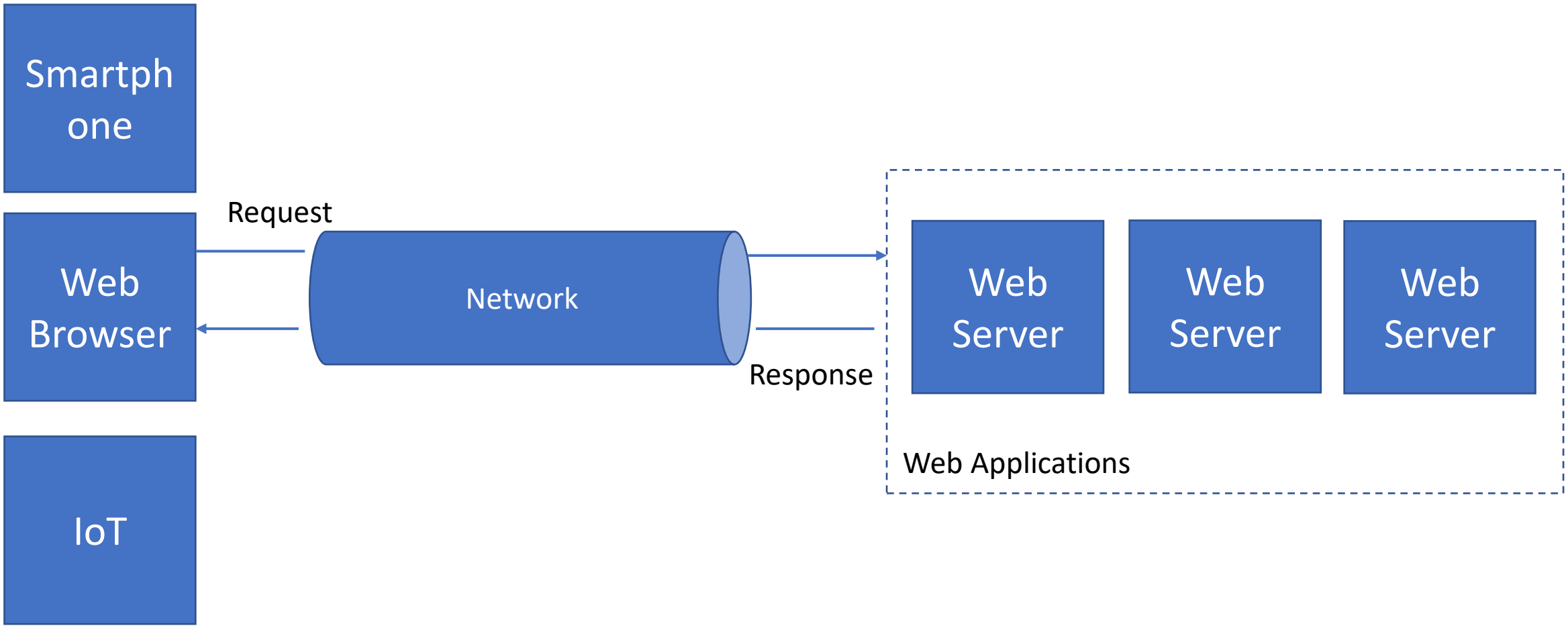
HTTP Revisited

Chandreyee Chowdhury

Web App Architecture-web 1.0



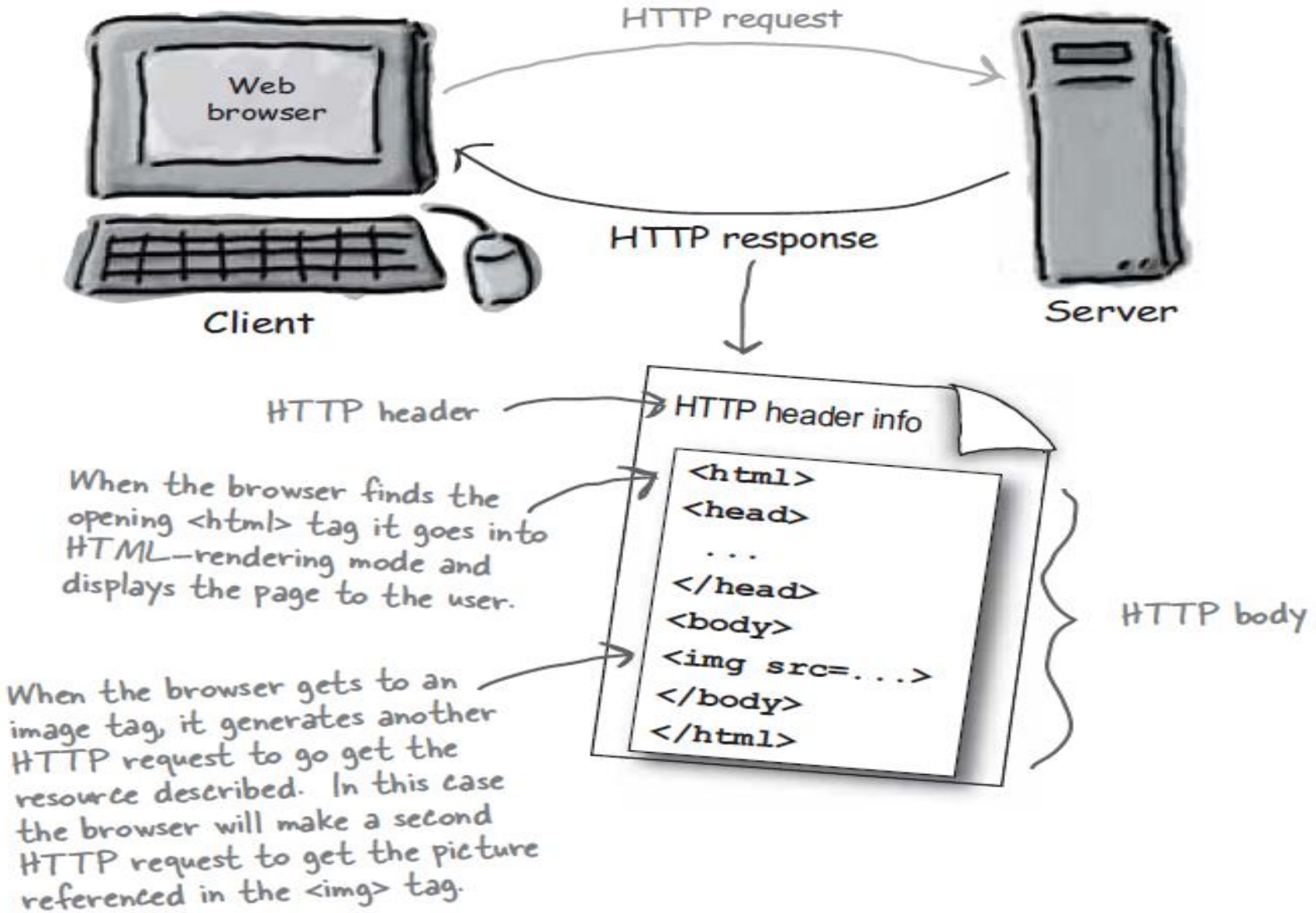
- Data and presentation are closely coupled as web pages are static

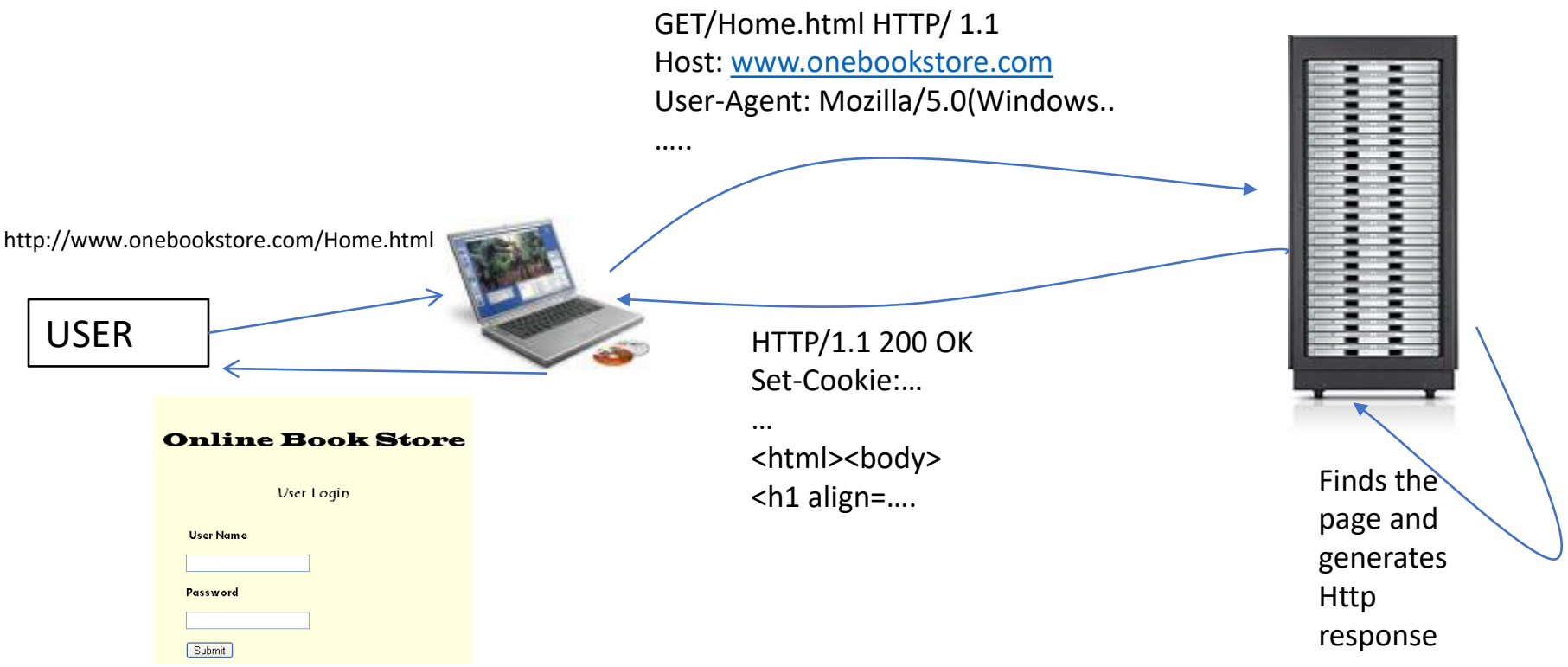

Smartph
one
Web
Browser
IoT
Request
Network
Response
Web
Server
Web
Server
Web
Server
Web Applications

Why HTTP

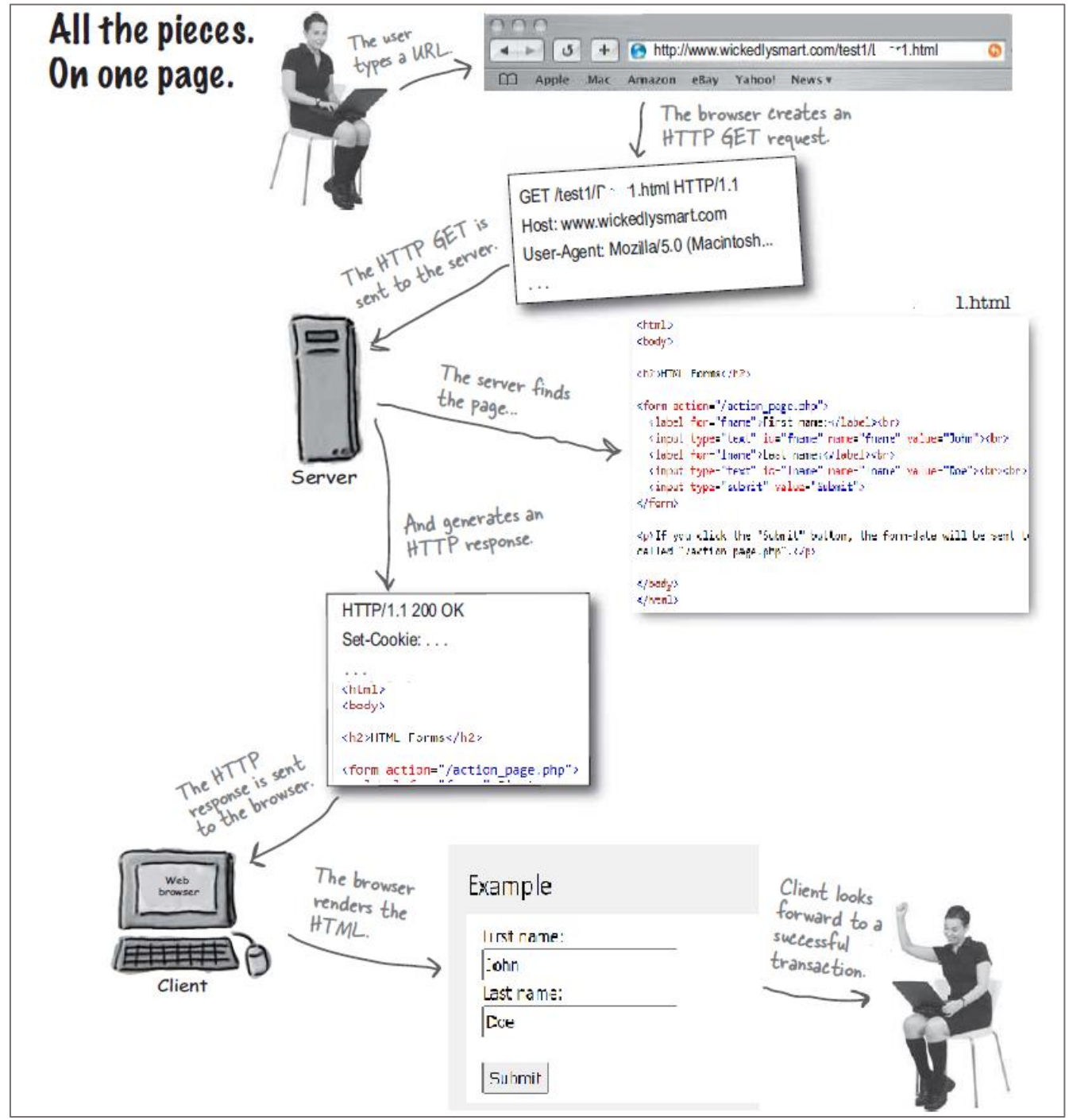
- It provides a uniform interface to access the resources and services from a web server or cloud
- Reuse infrastructure for ubiquity
  - Web is ubiquitous
- Reusing
  - Application frameworks and libraries
  - Load balancing infrastructure for different applications including interacting with cloud
  - Session handling
- Distribute requests throughout the servers

HTTP and HTML




GET/Home.html HTTP/ 1.1
Host: www.onebookstore.com
User-Agent: Mozilla/5.0(Windows..
…..
http://www.onebookstore.com/Home.html
USER
HTTP/1.1 200 OK
Set-Cookie:…
…
<html><body>
<h1 align=….
Finds the page and generates Http response
Online Book Store
User Login
User Name
Password
Submit

HTTP Request Response

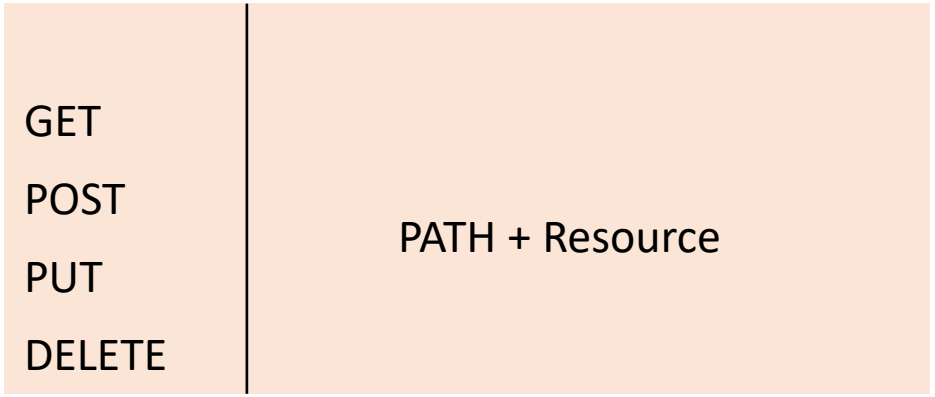


Http Request

- ***Every request has a method and a resource path***
- ***HTTP GET***
  - The total amount of characters in a GET is really limited (depending on the server)
  - The data you send with the GET is appended to the URL up in the browser bar, so whatever you send is exposed
  - Because of this, the user can bookmark a form submission if you use GET
- HTTP POST
  - The data is included in the request body
  - More data can be sent
  - General purpose sending of data

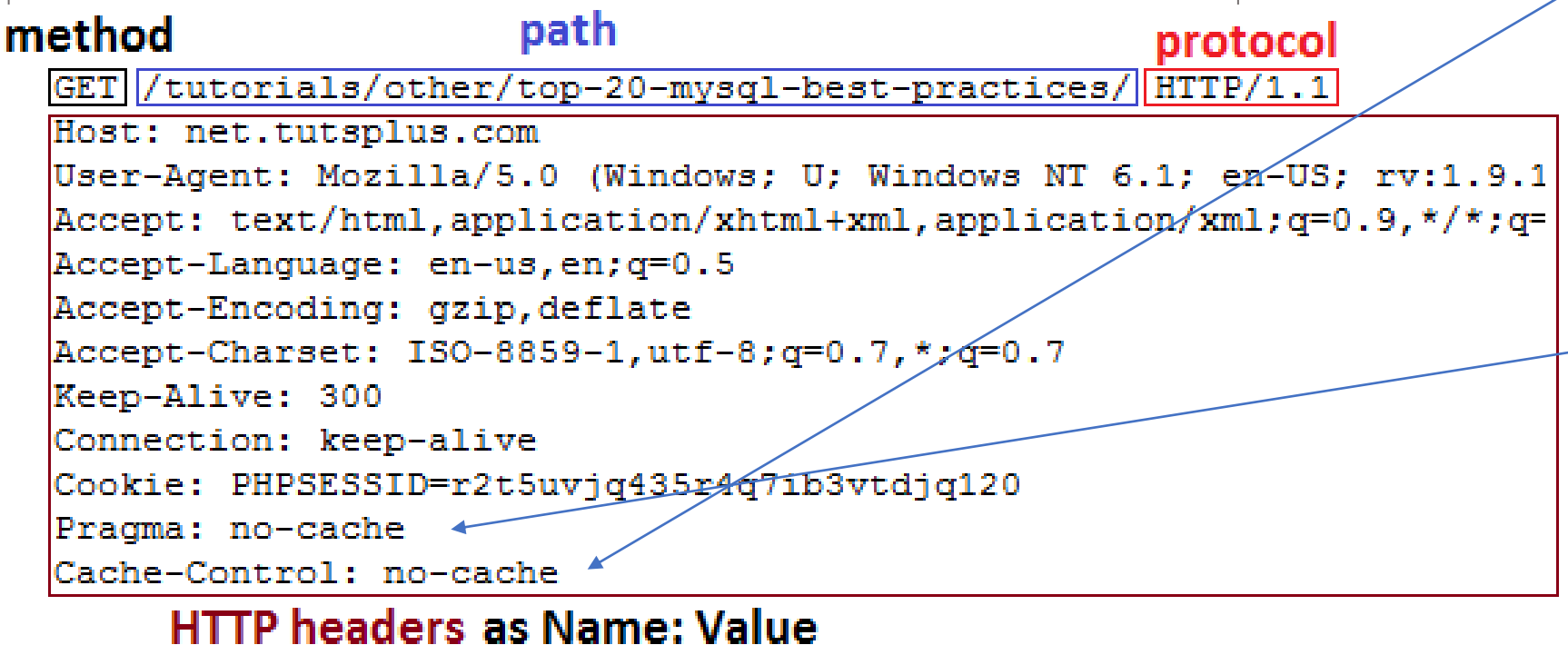
Http Methods

- Put
  - Asking the server to store some Data
- Delete
  - Remove some information from the server



Request line

- GET/com/Kolkata/Home.html HTTP/ 1.1
- Method
- <path+resource>

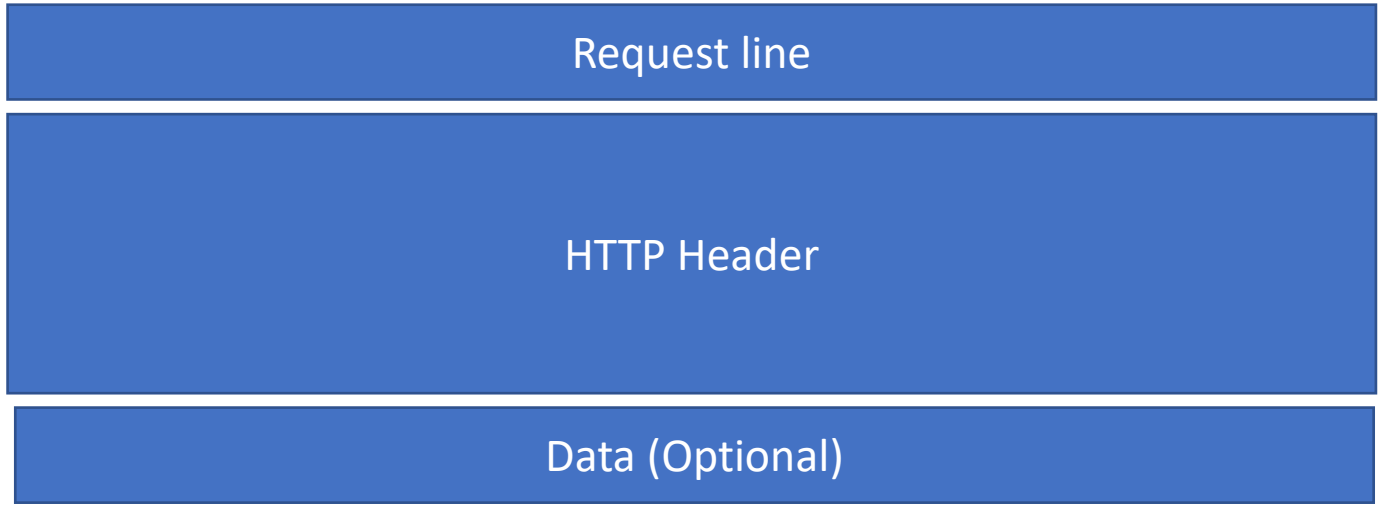


The response may be stored by *any* cache, even if the response is normally non-cacheable. However, the stored response MUST *always* go through validation with the origin server first before using it

The server MUST NOT use a cached copy when responding to such a request.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

HTTP Headers



- Headers are meta information but body of a HTTP message contains pure data
  - These are the extra information that the client is giving the server to help it complete that Request.
- If message body is sent without the header then the server may process the request
  - May not send the response in the expected format
- When body of a message is missing when it was required, the relevant information would not be processed

Uniform Resource Locator

https://wishnet.in/home/login.html

- The way resources are identified is called a URL
- `http://<host name>:<port number>/path/resource?key1=value1&key2=value2`
- Using the query parameters we can pass extra information about a specific aspect of a resource to the server
- URL encoding encodes any character that is not allowed in the query param spec
- For dynamically constructed URLs with data, it is better to encode all URLs as data may not follow the spec
- It is good to provide the correct file extension in the encoded URLs but not a requirement

Image/jpg

Image/png

Text/plain

Text/html

Data Types

- The way data is stored in the server and the way it is sent in the body may differ
  - MIME type allows this adaptation
- A media type (also known as a Multipurpose Internet Mail Extensions or MIME type) indicates the nature and format of a document, file, or assortment of bytes. MIME types are defined and standardized in IETF's RFC 6838
- There should be some way of interpreting the type of data sent in body
  - Image data
- The type represents the general category into which the data type falls, such as video or text.
  - The subtype identifies the exact kind of data of the specified type
- All of these different MIME types are identifiers for a well known format for the data in the body of either a request or a response.
  - Based on the MIME type the data will be processed
- MIME types are changed between client and server

Content-Type: multipart/form-data

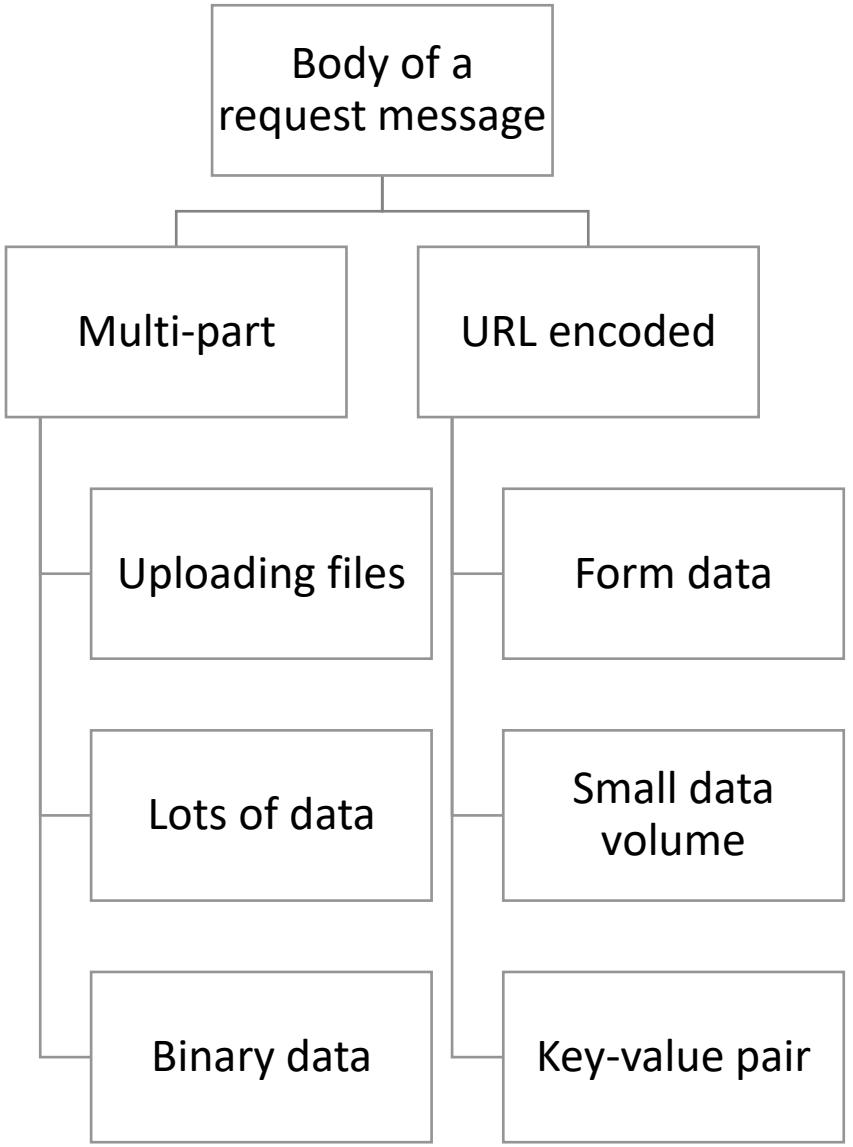
- These are written as content types
- Multipart types indicate a category of document broken into pieces, often with different MIME types

```
<form action="/SelectCoffeeType.html" method="post" enctype="multipart/form-data">
   <input type="text" name="description" value="some text">
   <input type="file" name="myFile">
   <button type="submit">Submit</button>
</form>
```

```
POST /foo HTTP/1.1
Content-Length: 68137
Content-Type: multipart/form-data; boundary=--------------------------974767299852498929531610575

----------------------------974767299852498929531610575
Content-Disposition: form-data; name="description"

some text
----------------------------974767299852498929531610575
Content-Disposition: form-data; name="myFile"; filename="foo.txt"
Content-Type: text/plain

(content of the uploaded file foo.txt)
----------------------------974767299852498929531610575--
```

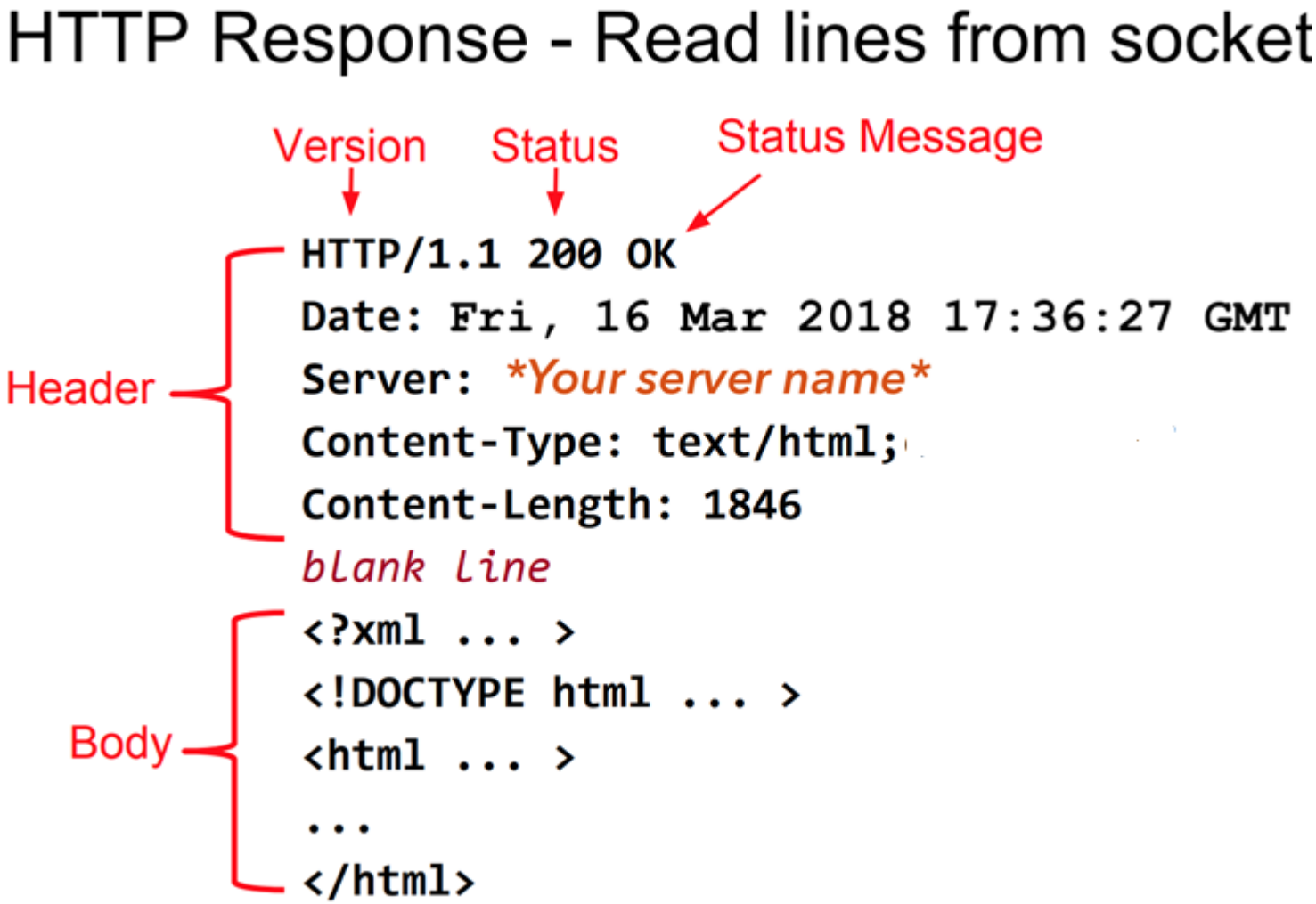
Request Body Encoding

- These are written as content types



HTTP Response

- We cant be present at the sever to check what has happened
- 1XX
- 2XX
- 3XX
- 4XX
- 5XX



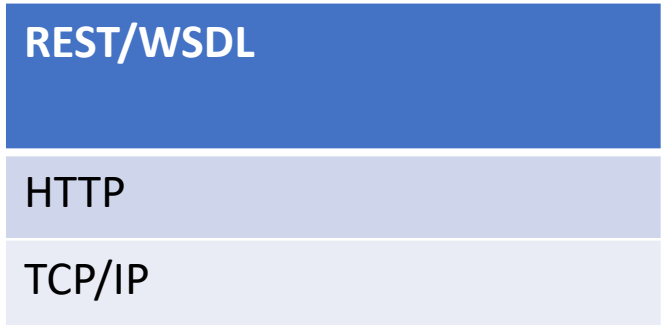
Response Codes

- 1XX- informal continuing process
- 2XX- successful
  - 200 means the client can assume that the server has successfully handled the request
- 3XX-redirection
  - Resend the request as the requested resource may have been moved
- 4XX- client error
  - Requested resource not found
  - Problem in request formatting
- 5XX-server error
  - The response body may contain the detailing of the error
- Depending on the response code and the MIME type, the body of the response is processed

Cookies

- They are very small limited pieces of data, that the server sends back to the client
- Goes through response header
- Size of the cookie has to be small
- Date/lifetime
- Encryption while sending the cookie
  - client only sends this cookie back to the server if a secure link, or an HTTPS communication protocol is being used

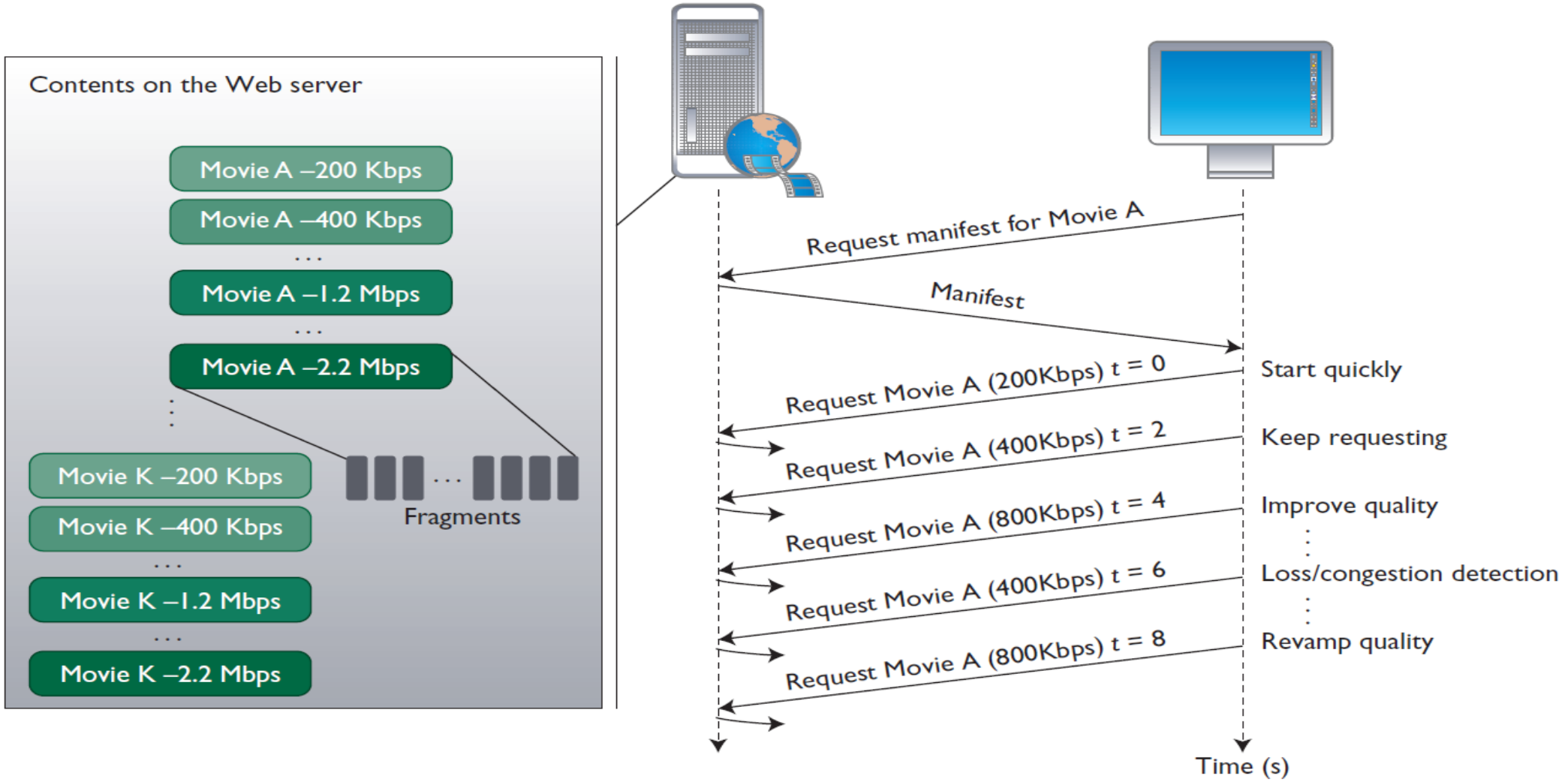
Protocol Layering



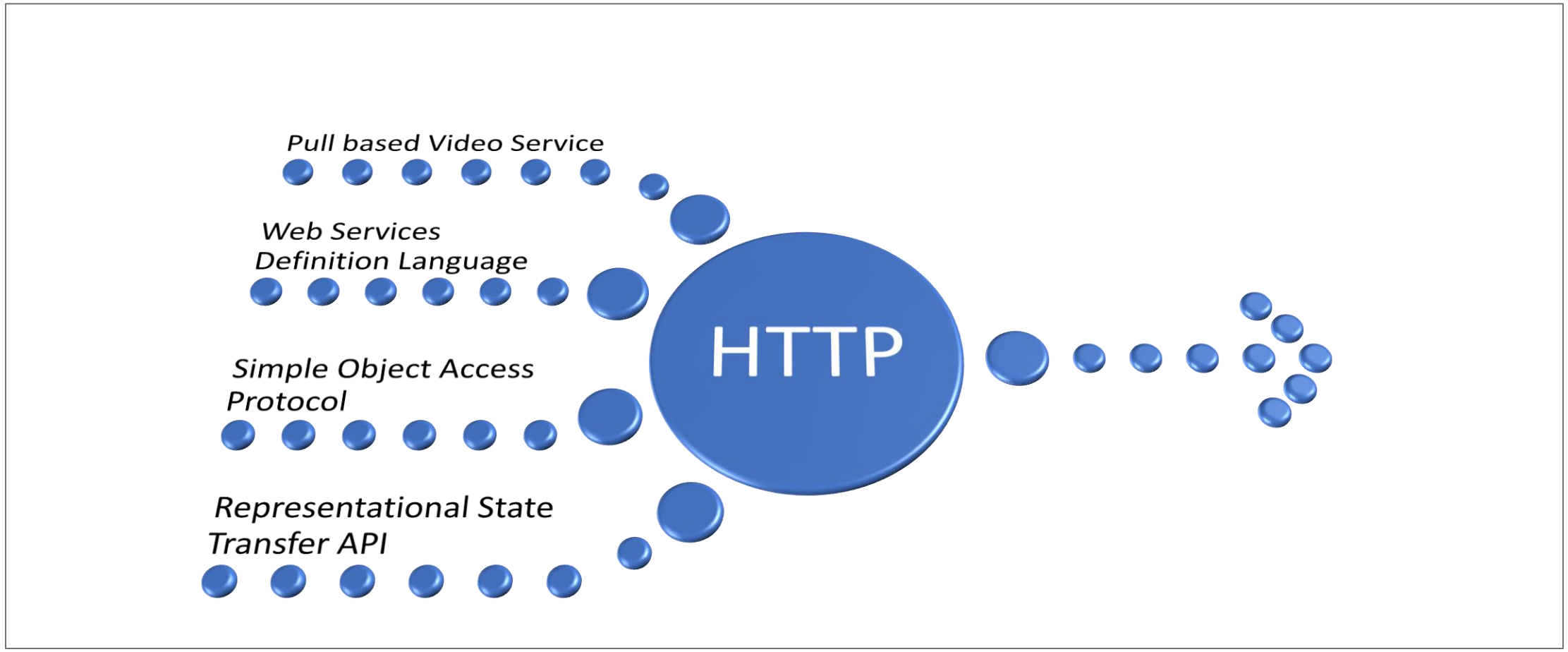
- Specifying rules for interaction and exchange of messages
- formatting of those messages
- If a nice interface is designed for the protocol then other protocols could even be layered on top of this protocol
- So HTTP is just a protocol layer among many in the application
- To design an application, the underlying protocol should be understood
- To design services over HTTP, we want to understand what the different options are for building on top of HTTP
  - How to encode parameters in the body or URL-query parameter or request body
  - Do we send request bodies that are multi-part encoded or form URL encoded?

Existing protocols on HTTP describe how you take messages or services and describe them using HTTP-based interactions

- **Microsoft Smooth Streaming**



Services on top of HTTP



Other protocols

- Web services and REST are methodologies for building on top of HTTP that describes very specific formats and principles for building things on HTTP
- Most applications support REST like services or webservices like properties
- Building in some object which can live in the cloud or some service which can live in the cloud and can receive information and be interacted with over HTTP
- Depending on the design of that service, those interactions may follow a REST model, or they may follow a web service model.

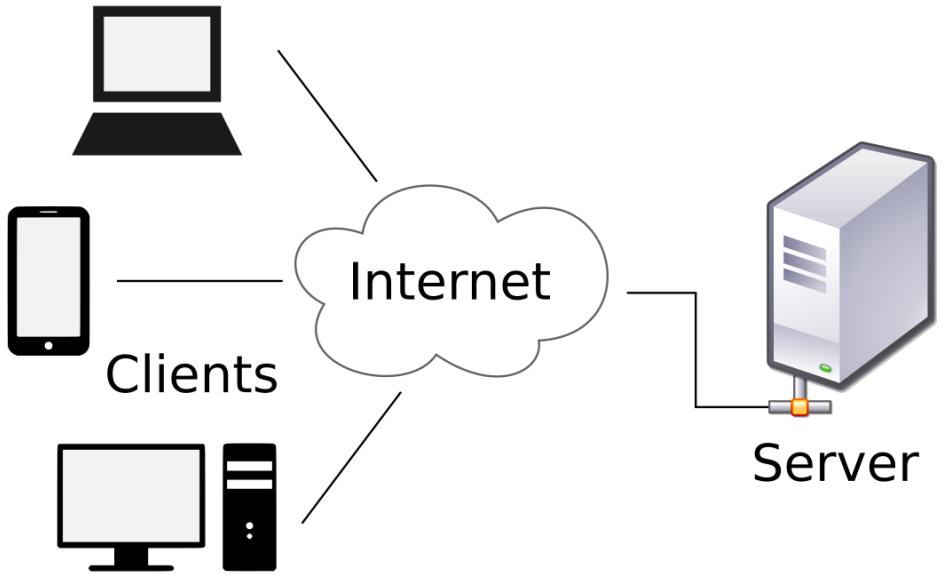
GET	https://www.coursera.org/learn/cloud-services-java-spring-framework/lecture/HBmkG/protocol-layering-http-design-methodologies
POST	
PUT	
DELETE	

REST or REST like Services

- URL Addressing scheme
- /video — All videos list
- /video/1 — Specific video using an identifier
- /video/1/duration — Duration of a video or specific part of the video

- GET-retrieve data
- PUT-create or replace
- POST-To add a new video and get back its URL identifier
- DELETE-remove resources at the server side

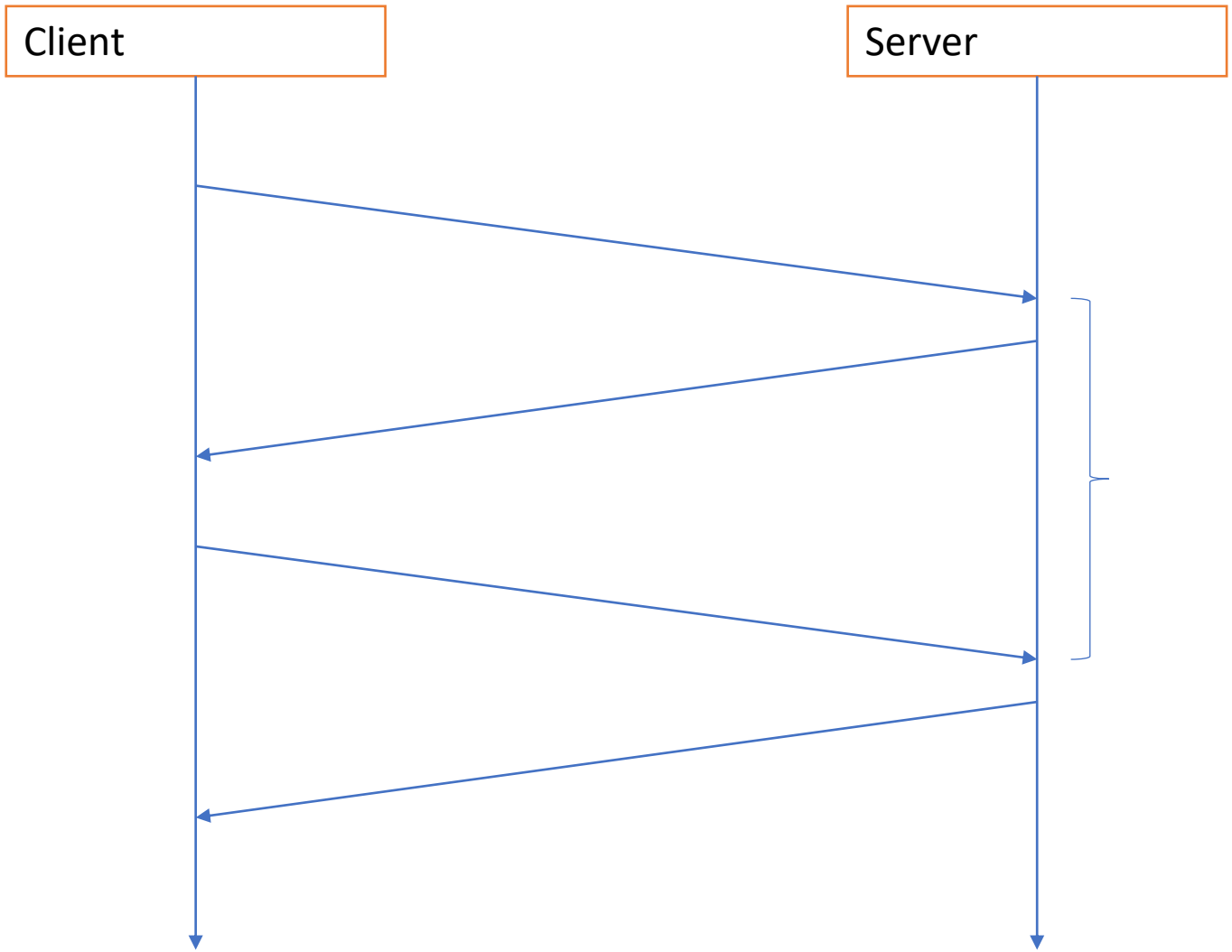
HTTP is client driven



- If client 2 updates important data that client 1 just pulled in, unless client 1 makes a second request for an update the server cannot notify client 1
- Server is not able to push data as and when needed

Pushing Data

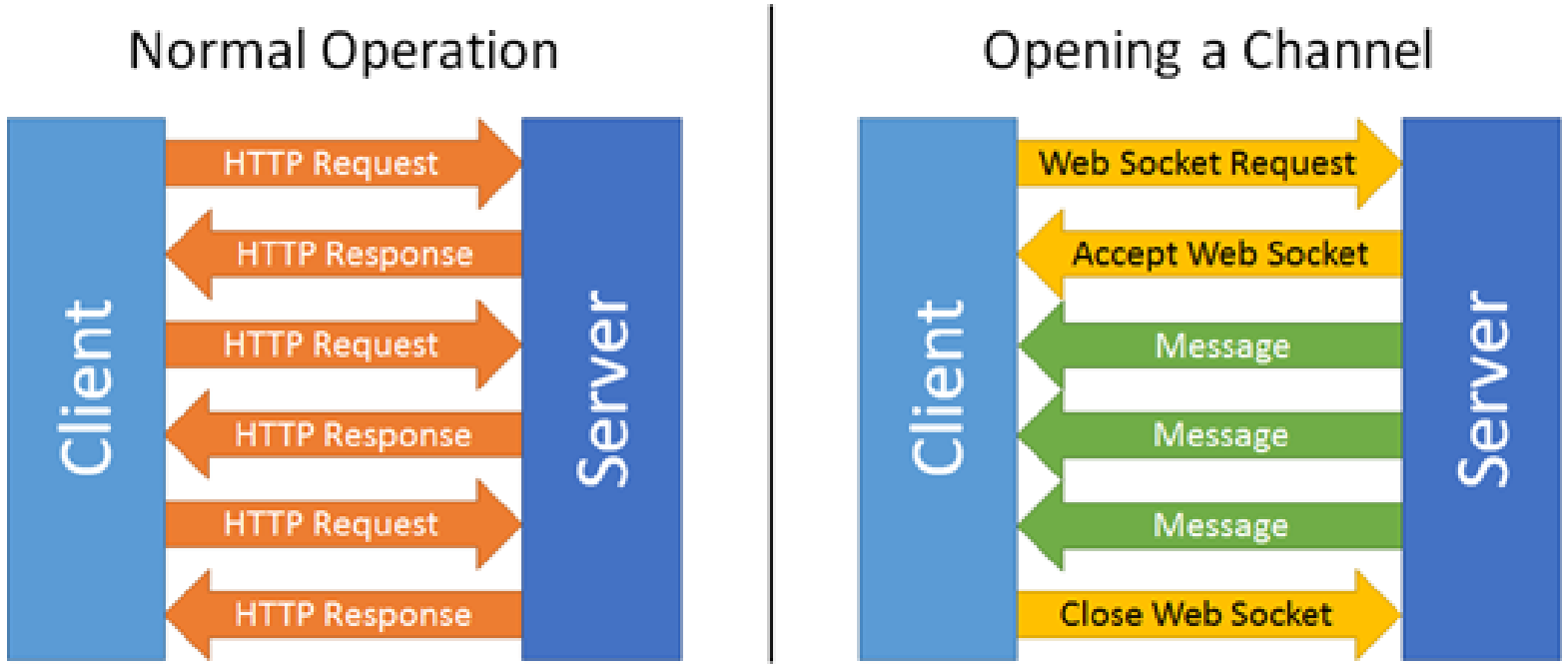
- Manual- User may click on “refresh” when (s)he has decided to pull the data
- Every time the window is opened data is pulled from the server
- Periodic update
  - Overhead on the server processes as it needs to process the request even when no update is to be notified
  - Polling
  - Exponential backoff



Periodic Update

https://dzone.com/articles/thoughts-on-server-sent-events-http2-and-envoy-1

WebSockets



- communication protocol which features bi-directional, full-duplex communication over a persistent TCP connection
- Any party can push data anytime
- Single TCP connection for full duplex traffic
- Message transfer on websockets does not require all parts of HTTP to be sent (header, URL, content type, body etc.)
- Simply send binary messages or some other format back and forth in a server
- By default, port 80 is used
- Port 443 is used for connection tunneled over the TLS

Web Socket HTTP compatibility

Web Socket Advantages

- Stateful connection
- Message overhead of polling is more than web socket
- STOMP- Simple Text Oriented Messaging Protocol

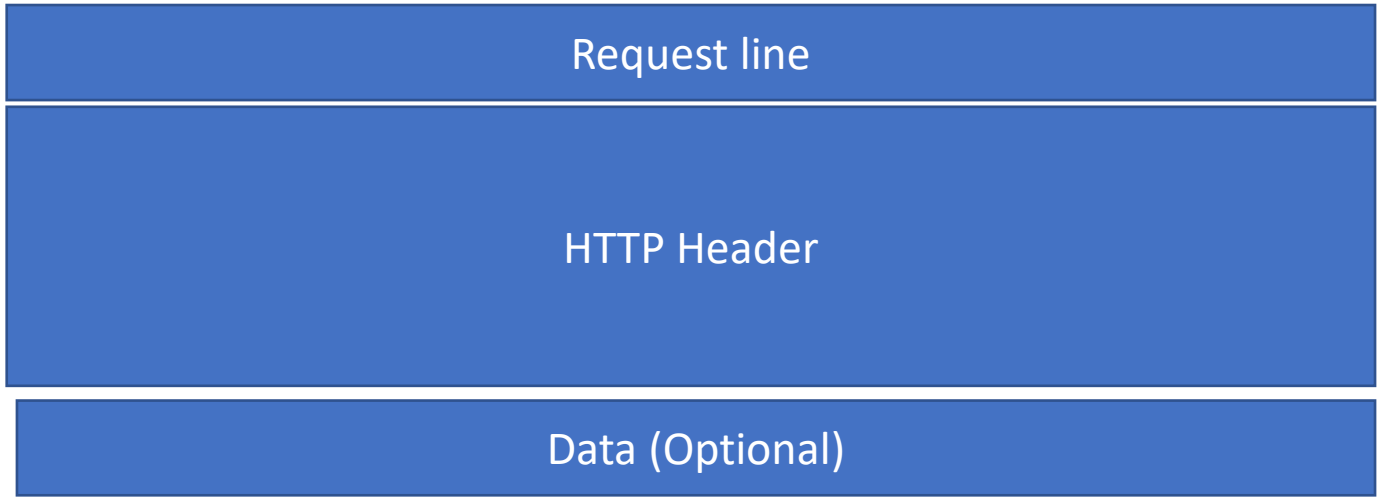
https://spring.io/guides/gs/messaging-stomp-websocket/

S. El Mimouni and M. Bouhdadi, "Formal modeling of the Simple Text Oriented Messaging Protocol using Event-B method," 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA), 2015, pp. 1-4, doi: 10.1109/AICCSA.2015.7507170.

Web Socket Problems

- Web sockets enable a server to push data only if the client is connected
- Web sockets are difficult to synchronize with more clients
- Keeping an open connection can have substantial resource impact
- For shared hosting servers, web socket is not a scalable option
- http responses can be cached by browser or by proxies
    - There is no such built-in mechanism for requests sent via webSockets

Server Data Push

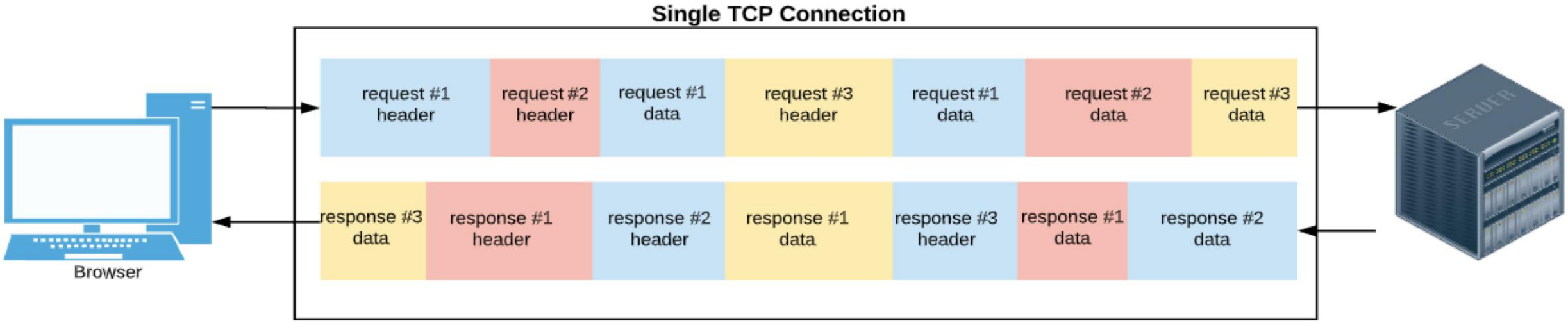


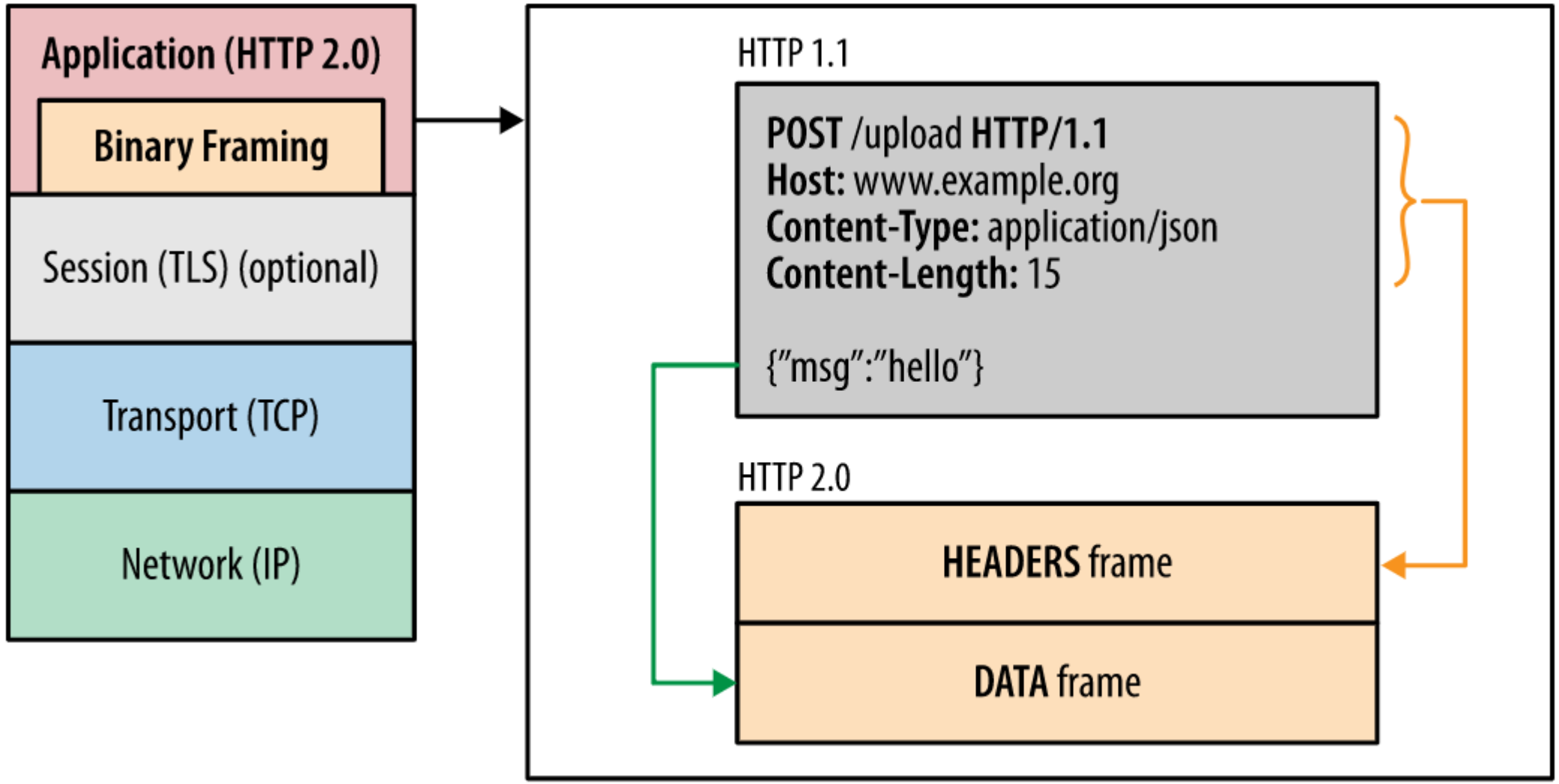
- Server Sent Event (SSE)-an API for managing and parsing events from long-running HTTP connections
- Head of line blocking problem
  - if the client sends requests A, B, and C, but request A requires a lot of server resources, B and C are blocked until A finishes
- HTTP 2.0
  - In addition to the response to the original request, the server can push additional resources to the client

Request line

HTTP Header

Data (Optional)

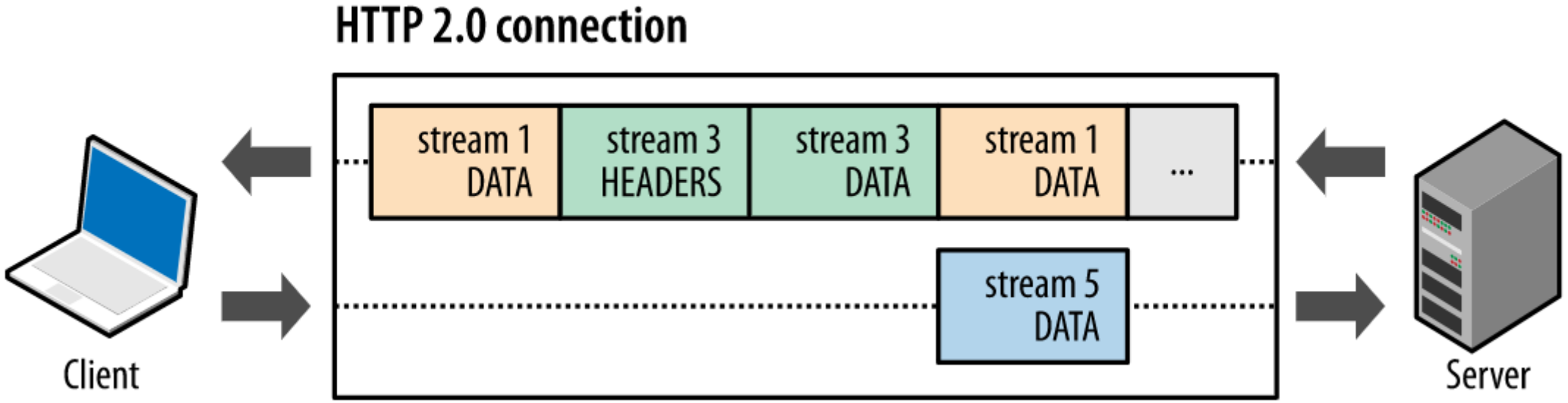




https://developers.google.com/web/fundamentals/performance/http2

HTTP/2.0

- *Stream*: A bidirectional flow of bytes within an established connection, which may carry one or more messages.
- *Message*: A complete sequence of frames that map to a logical request or response message.
- *Frame*: The smallest unit of communication in HTTP/2, each containing a frame header, which at a minimum identifies the stream to which the frame belongs.
- Each stream may be assigned an integer weight between 1 and 256.
- Each stream may be given an explicit dependency on another stream.



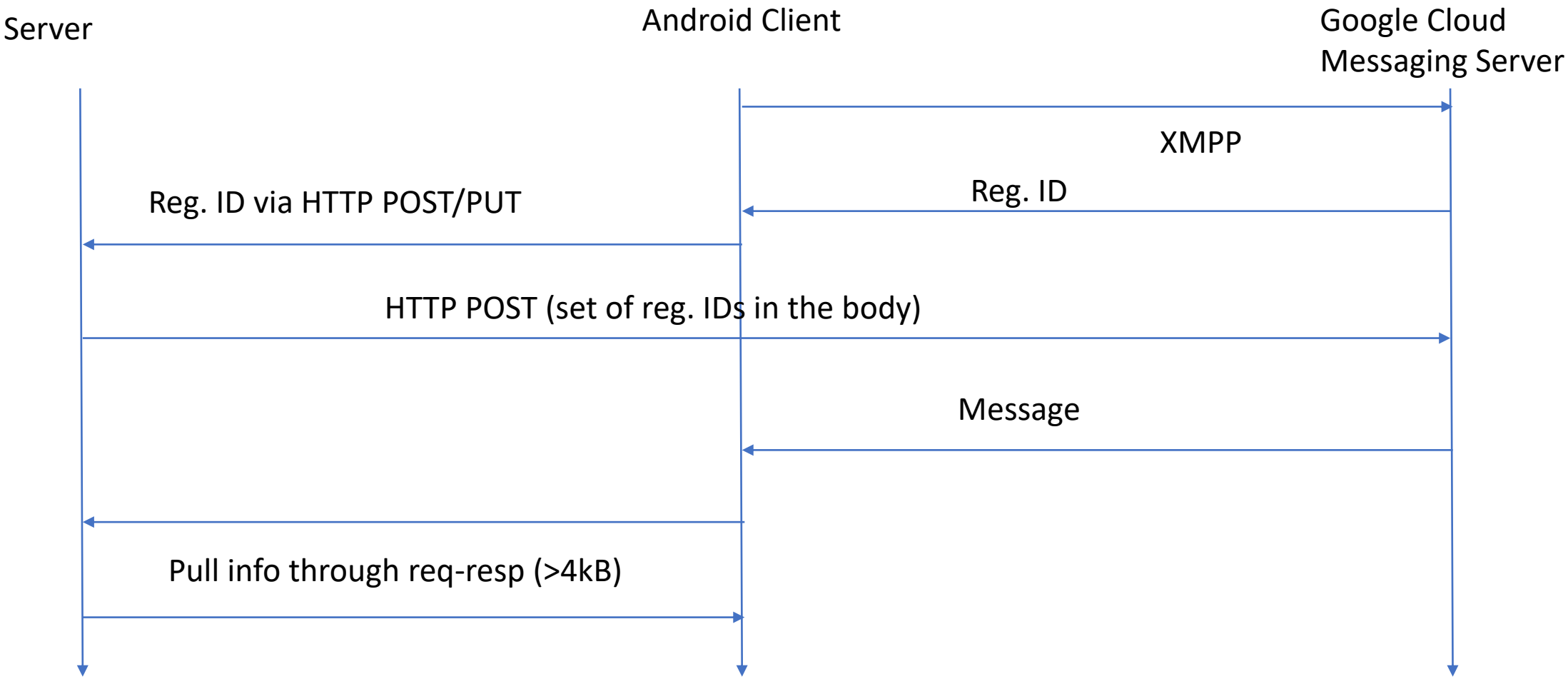
Http/2 benefits

- With HTTP/1.x, if the client wants to make multiple parallel requests to improve performance, then multiple TCP connections must be used
- This also results in head-of-line blocking and inefficient use of the underlying TCP connection
- The ability to break down an HTTP message into independent frames, interleave them, and then reassemble them on the other end is the single most important enhancement of HTTP/2.
- In fact, it introduces a ripple effect of numerous performance benefits across the entire stack of all web technologies
- Interleave multiple requests in parallel without blocking on any one.
- Interleave multiple responses in parallel without blocking on any one.
- Use a single connection to deliver multiple requests and responses in parallel.
- Deliver lower page load times by eliminating unnecessary latency and improving utilization of available network capacity.

Intermittent Internet Connection

- How to handle intermittent internet connections
  - Event mechanisms and notifications should be designed to detect disconnection and reconnection
  - Periodically reconnect/ or getting an event from Android

Push Notification



Push to Sync Model

- GCM handles the disconnection and reconnection issues through Android Cloud libraries
- Server sends an event pushed to the client that results in a pull by the client from the server
- Better than polling as the client now knows when to pull information
- Both message size and privacy issues dictate the exact design of the push notification
- Advantages
  - No need to push out large amount of information
  - The user can control the data dissemination and security over that dissemination of data