HTTP - Part 1

Communication

5. Application Layer e.g. HTTP,

4. Transport Layer e.g. TCP, UDP

3. Network/Internet Layer e.g. IP4

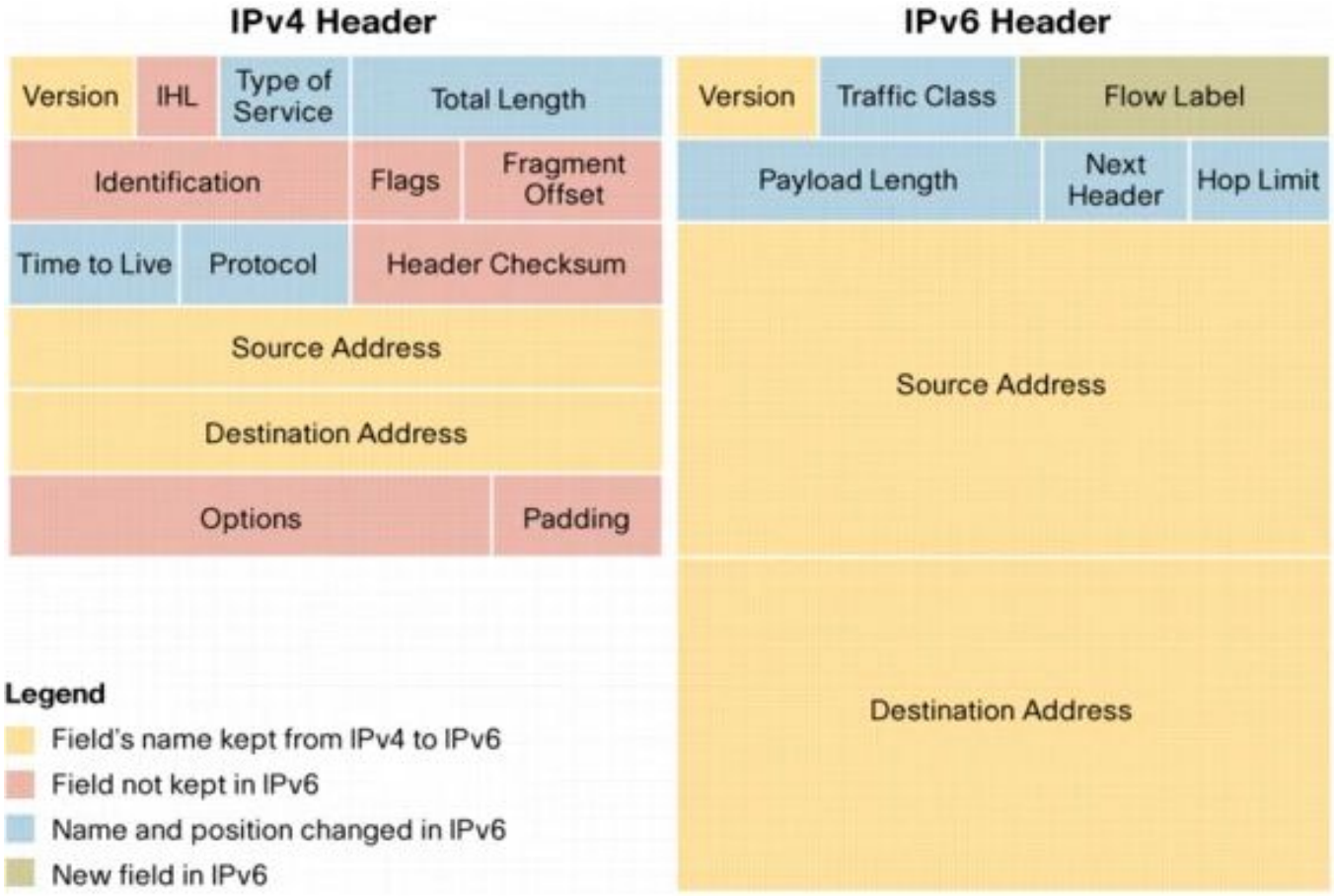
2. Data Link Layer e.g. 802.11, Ethernet

1. Physical Layer e.g. Modems, Twisted Pair

IP 4

Offset	4-bit	8-bit	16-bit	32-bit
0	Ver.	Header Length	Type of Service	Total Length
32	Identification		Flags	Offset
64	Time To Live	Protocol	Checksum	
96	Source Address			
128	Destination Address			
160	Options and Padding			
160 or 192	Data			

http://www.informit.com/content/images/chap3_0672323516/elementLinks/03tab


IPv4 Header
Version
IHL
Type of Service
Total Length
Identification
Flags
Fragment Offset
Time to Live
Protocol
Header Checksum
Source Address
Destination Address
Options
Padding
IPv6 Header
Version
Traffic Class
Flow Label
Payload Length
Next Header
Hop Limit
Source Address
Destination Address
Legend
Field's name kept from IPv4 to IPv6
Field not kept in IPv6
Name and position changed in IPv6
New field in IPv6

TCP & UDP

- Transmission Control Protocol
  - TCP
  - Core Protocol
  - Provide reliable, ordered communication
  - Used as the basic protocol for many applications e.g. WWW

- User Datagram Protocol
  - UDP
  - Fast, compact but no guarantees

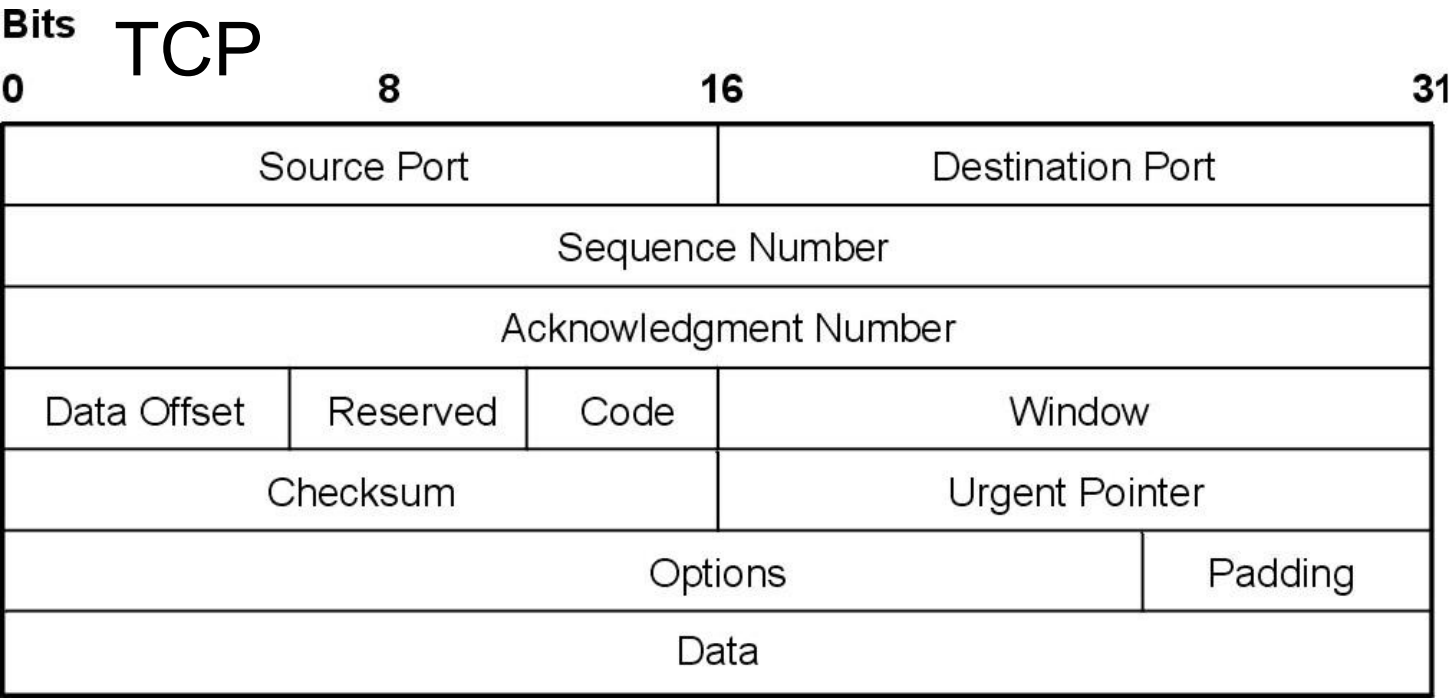
UDP

- One way communication
- No guaranteed delivery
- Often used for VoIP, DNS, etc

TCP Connection

- TCP is the most common protocol
- Web runs on TCP
- Guaranteed delivery
- Two-way communication

TCP



http://www.cisco.com/en/US/i/Other/Cisco_Press/ITG/10-19-01/TR890706.jpg

TCP/IP COMMUNICATION

- Every machine has to have a unique IP address
  - WWW.USASK.CA
  -
- Every machine has to use a port
- Some Ports have special meanings
  - E.g. 80
  -

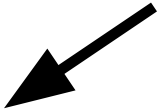
Domain Name System

- IP address are not sufficient
  - Hard to remember
  - IP addresses of servers can change
- Naming system for computers
  - Hosts file (old way)
  - Using DNS servers (new way)
- Each domain name can be translated into IP
  - nslookup – translation tool
  - www.cs.usask.ca -> …….

How does DNS work ?

- Each computer knows IP of at least one DNS server
- If your DNS server does not know the requested name it polls another DNS server

HTTP – HYPERTEXT TRANSFER PROTOCOL



5. Application Layer e.g. HTTP,

4. Transport Layer e.g. TCP, UDP

3. Network/Internet Layer e.g. IP4

2. Data Link Layer e.g. 802.11, Ethernet

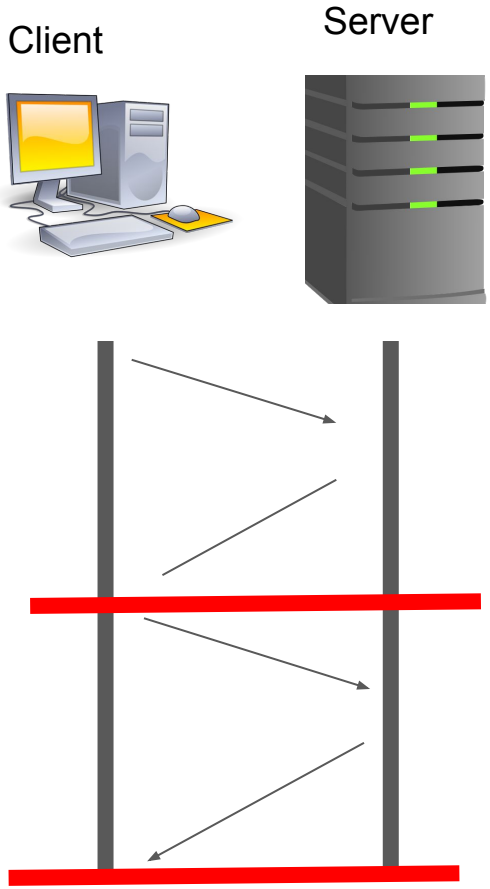
1. Physical Layer e.g. Modems, Twisted Pair

HTTP VERSIONS

- Many different versions
  - HTTP 0.9 – supports only GET
  - HTTP/1.0 (1996)
  - HTTP/1.1 (1997)
  - HTTP/2 (2015)
  - HTTP/3 (2022)
    - **QUIC**
- Browser & Server negotiate version

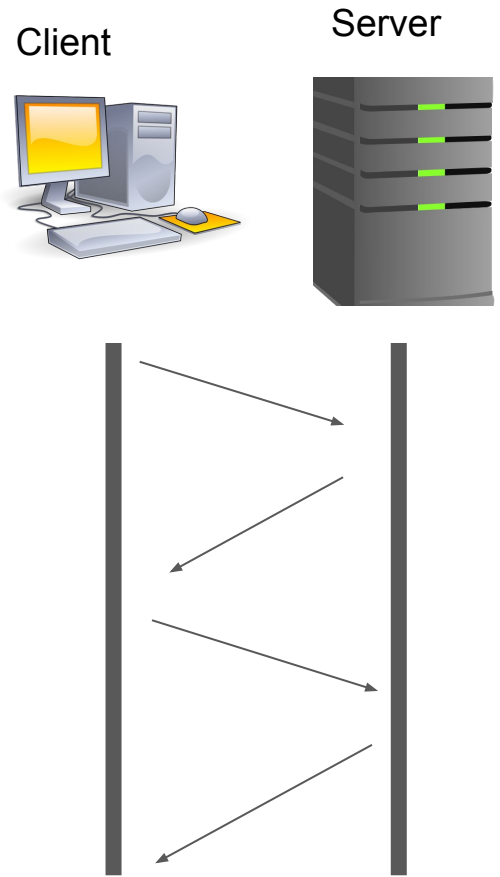
HTTP 1

- 1996
- Request == create TCP connection



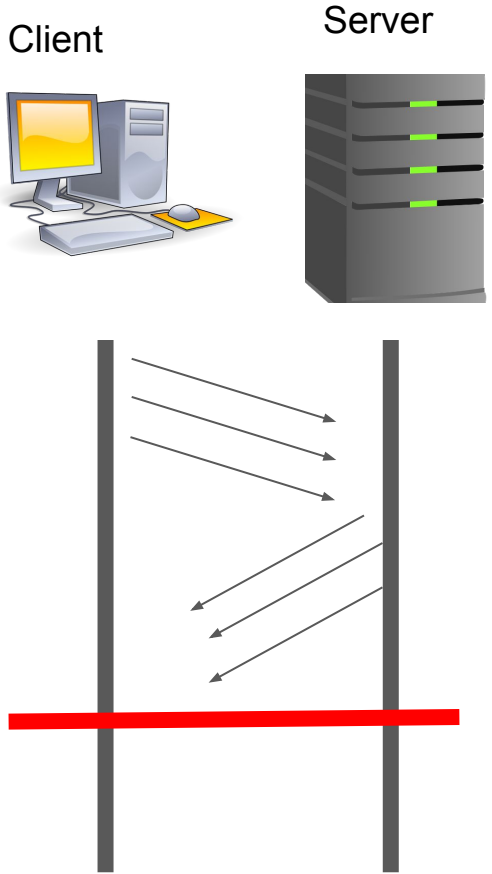
HTTP 1.1

- 1997
- “Keep-alive”
  - Ability to reuse existing TCP connection
  - Speedup due to costly creation of connection
-



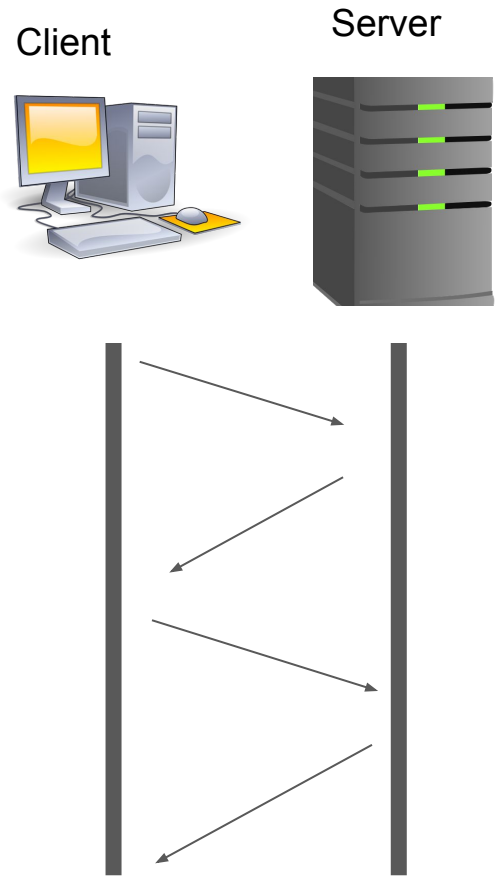
HTTP 1.1

- Pipelining
  - Send multiple requests
  - Receive results in same order
- Problem infrastructure
  - Proxies couldn’t handle
  - Not a successful feature !
  - Removed by some browsers
- Problem head-of-line blocking
  - Slow request can block subsequent ones
    - Packet loss
    - Server needs more time ..



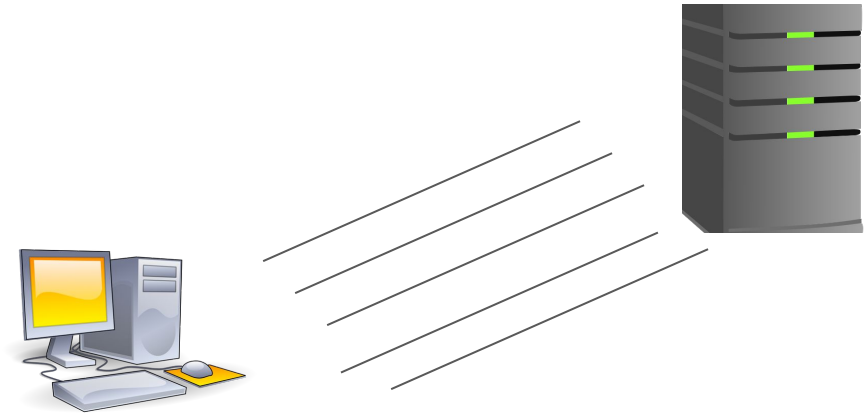
HTTP 1.1

- 1997
- “Keep-alive”
  - Ability to reuse existing TCP connection
  - Speedup due to costly creation of connection
-



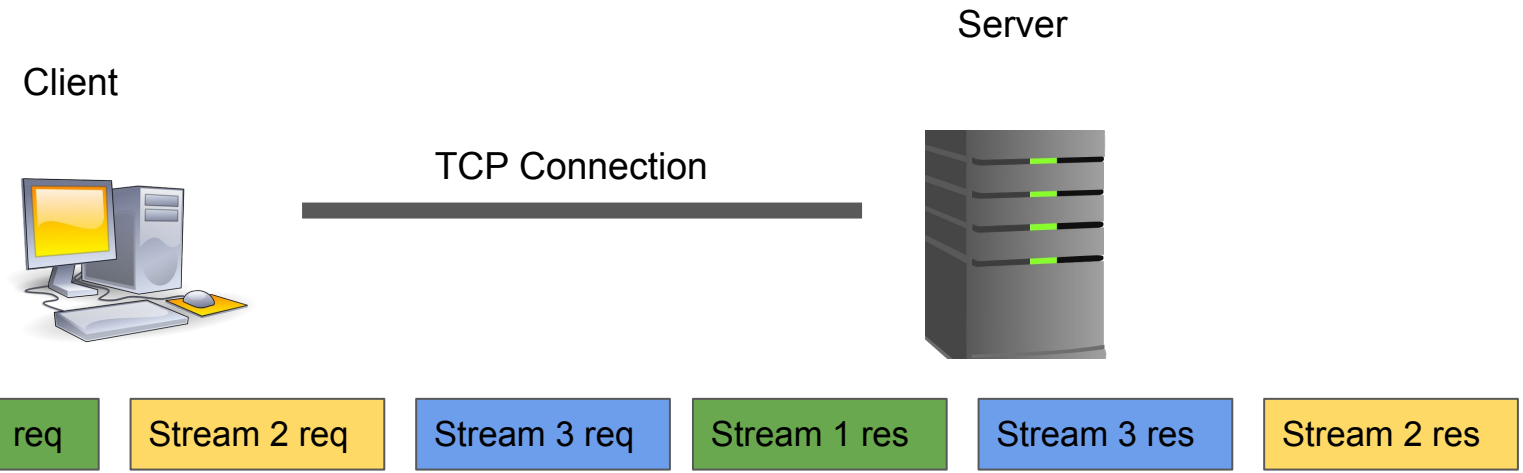
HTTP 1.1

- Multiple active connections
- Ensure faster request/response



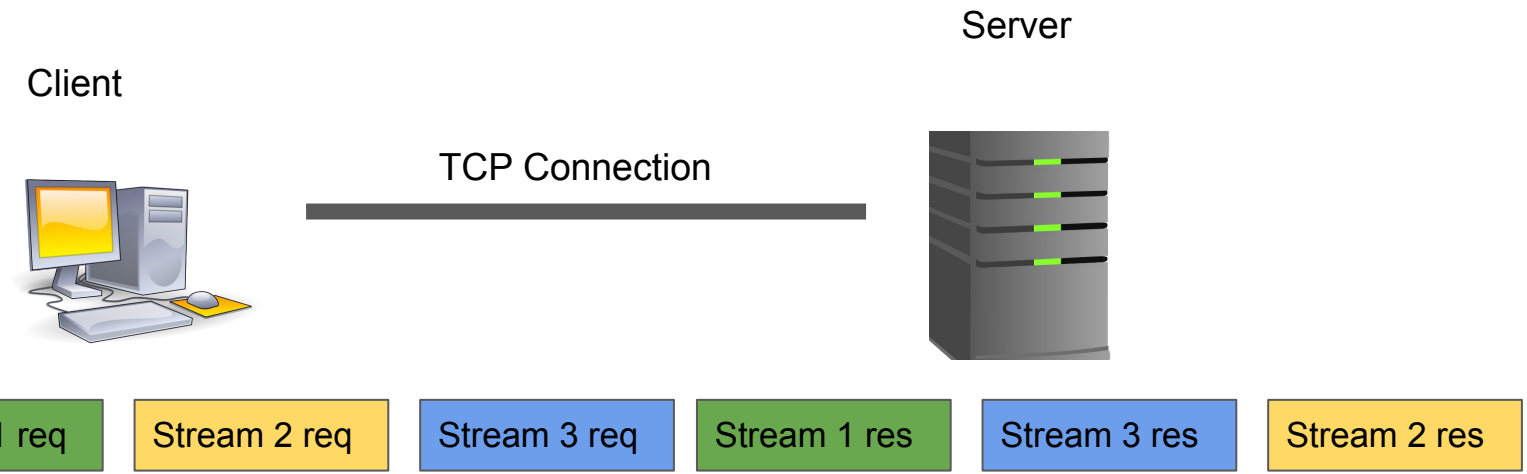
HTTP 2

- 2015
- “streams”
  - Ability to reuse existing TCP connection
  - Speedup due to costly creation of connection
- Header compression



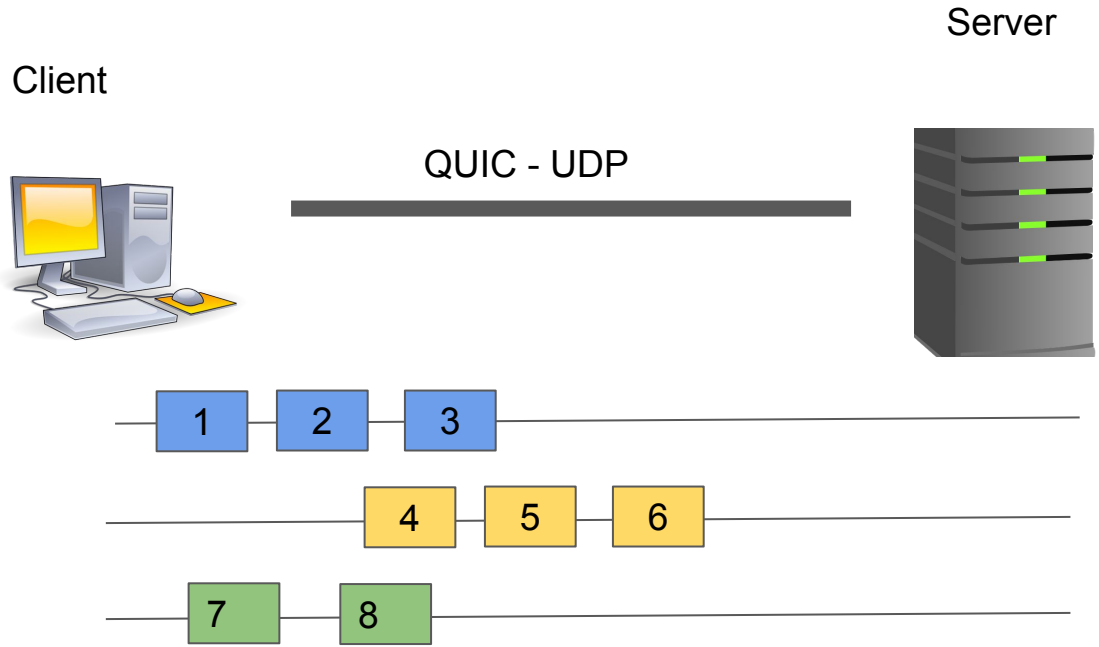
HTTP 2

- Head-of-line Blocking
  - Solved on application layer
  - Still issue on transport layer
- Push
  - Send updates to client
  - Send resources/response before request
  - ??????
  - Index.htm -> need for css & javascript -> send it after index.html
-



HTTP 3

- Idea -> 2020 -> 2022 in use
  - QUIC -> UDP
  - Packet loss in one stream doesn’t impact other streams



HTTP 3

- Target
  - Mobile, data intensive interactions
    - Allows switching networks e.g. 5G, 3G, 4G
- Connection ID
  - Enables connection across networks
  - Different network but use same ID

HTTP COMMANDS

- HEAD
  - Asks for the response identical to the one that would correspond to a GET request, but without the response body. This is useful for retrieving meta-information written in response headers, without having to transport the entire content.
- GET
  - Requests a representation of the specified resource. By far the most common method used on the Web today. Should not be used for operations that cause side-effects (using it for actions in web applications is a common misuse). See 'safe methods' below.
- POST
  - Submits data to be processed (e.g. from an HTML form) to the identified resource. The data is included in the body of the request. This may result in the creation of a new resource or the updates of existing resources or both.
- PUT
  - Uploads a representation of the specified resource.
- DELETE
  - Deletes the specified resource.

http://en.wikipedia.org/wiki/HTTP

OTHERS

http://en.wikipedia.org/wiki/HTTP

- TRACE
  - Echoes back the received request, so that a client can see what intermediate servers are adding or changing in the request.
- OPTIONS
  - Returns the HTTP methods that the server supports. This can be used to check the functionality of a web server.
- CONNECT
  - Converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy

Idempotent & Safe

- Idempotent -> don’t change response
- Safe -> can be cached

Example

```
GET /echo HTTP/1.1
Host: reqbin.com
Accept: */*
```

```
GET /echo HTTP/1.1
Host: reqbin.com
Accept: text/html
```

```
GET /echo/get/json HTTP/1.1
Host: reqbin.com
Accept: application/json
```

```
GET /tutorials/other/top-20-mysql-best-practices/ HTTP/1.1
 Host: code.tutsplus.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.5)
Gecko/20091102 Firefox/3.5.5 (.NET CLR 3.5.30729)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq120
```

```
HTTP/1.x 200 OK
Transfer-Encoding: chunked
Date: Sat, 28 Nov 2009 04:36:25 GMT
Server: LiteSpeed
Connection: close
X-Powered-By: W3 Total Cache/0.8
Expires: Sat, 28 Nov 2009 05:36:25 GMT
Etag: "pub1259380237;gz"
Cache-Control: max-age=3600, public
Content-Type: text/html; charset=UTF-8
Last-Modified: Sat, 28 Nov 2009 03:50:37 GMT
Content-Encoding: gzip
Vary: Accept-Encoding, Cookie, User-Agent


<!DOCTYPE html>

<html>

 <head>
```

```
POST /foo.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.5) Gecko/20091102
Firefox/3.5.5 (.NET CLR 3.5.30729)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 43

 DATA
```

```
GET /search?q=test HTTP/2
Host: www.bing.com
User-Agent: curl/7.54.0
Accept: */*
```

HTTP STATUS CODES

- 1xx
  - Informal
- 2xx
  - Client Request Successful
- 3xx
  - Redirection
- 4xx
  - Client Request incomplete
- 5xx
  - Server Errors

HTTP HEADERS

- General
  - Not related to client, server or HTTP
- Request
  - Preferred document formats and server parameters
- Response
  - Information
- Entity
  - Information about the data being sent between client and server

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