Full Stack Application Development- SE ZG503

Akshaya Ganesan

CSIS, WILP

BITS Pilani

Pilani Campus

innovate achieve lead

BITS Pilani

Pilani Campus

Lecture No: 6.1 – HTTP Connection

Module 3: Web Protocols

HTTP
- HTTP Request- Response and its structure
- HTTP Methods;
- HTTP Headers
- Connection management - HTTP/1.1 and HTTP/2

Synchronous and asynchronous communication

Communication with Backend
- AJAX, Fetch API
- Webhooks
- Server-Sent Events

Polling

Bidirectional communication - Web sockets

Agenda

- Connection management - HTTP/1.1 and HTTP/2
- HTTPS

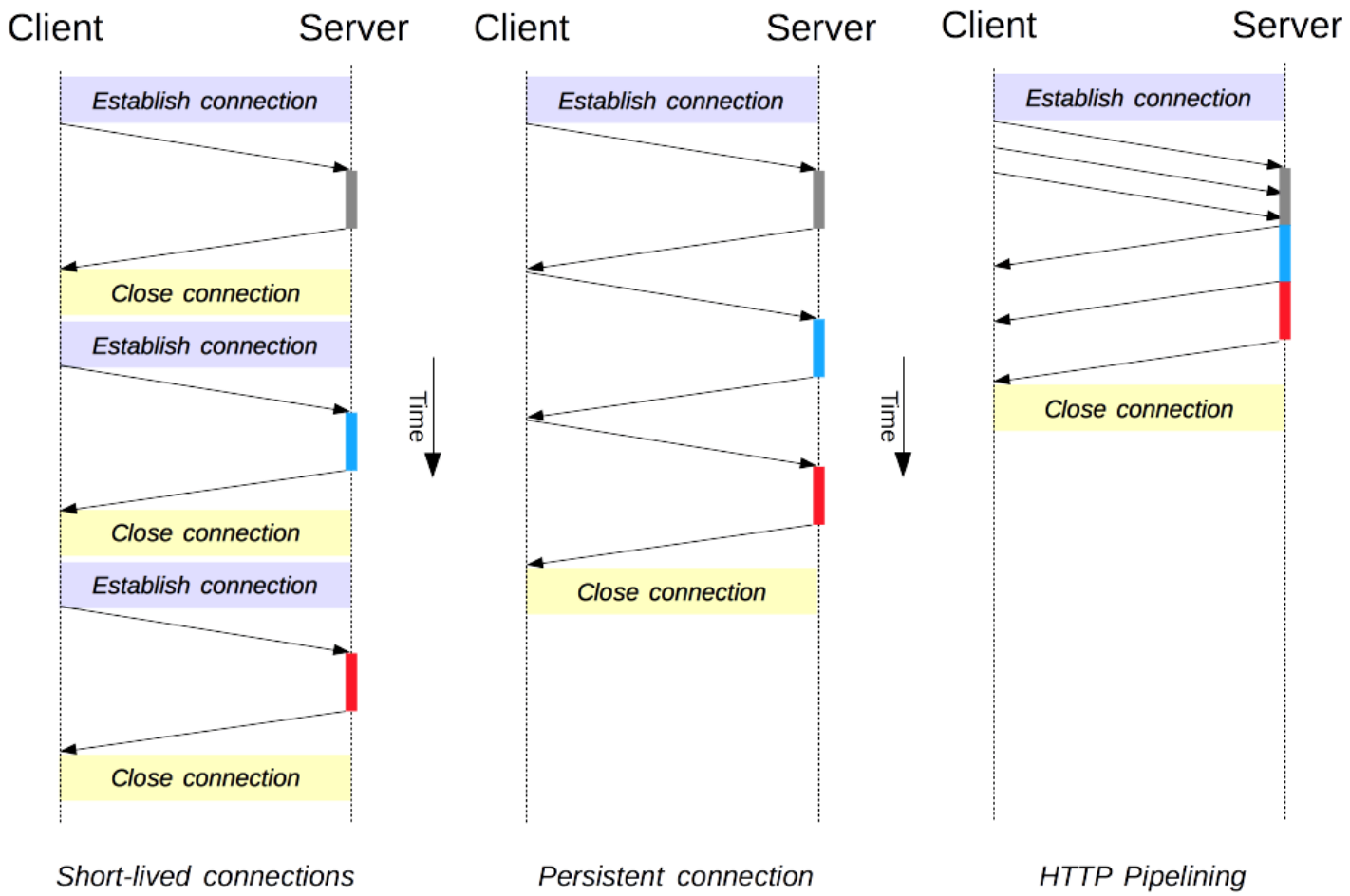
HTTP Headers- Connection Management

- The Connection general header controls whether or not the network connection stays open after the current transaction finishes.
- If the value sent is keep-alive, the connection is persistent and not closed, allowing for subsequent requests to the same server to be done.
- Connection: keep-alive
- Connection: close

Keep-alive Header

- The Keep-Alive general header allows the sender to hint about how the connection may be used to set a timeout and a maximum amount of requests.
- Keep-Alive: timeout=5, max=1000

HTTP Connection 1.x



Reference: https://developer.mozilla.org/en-US/docs/Web/HTTP/Connection_management_in_HTTP_1.x

HTTP/1

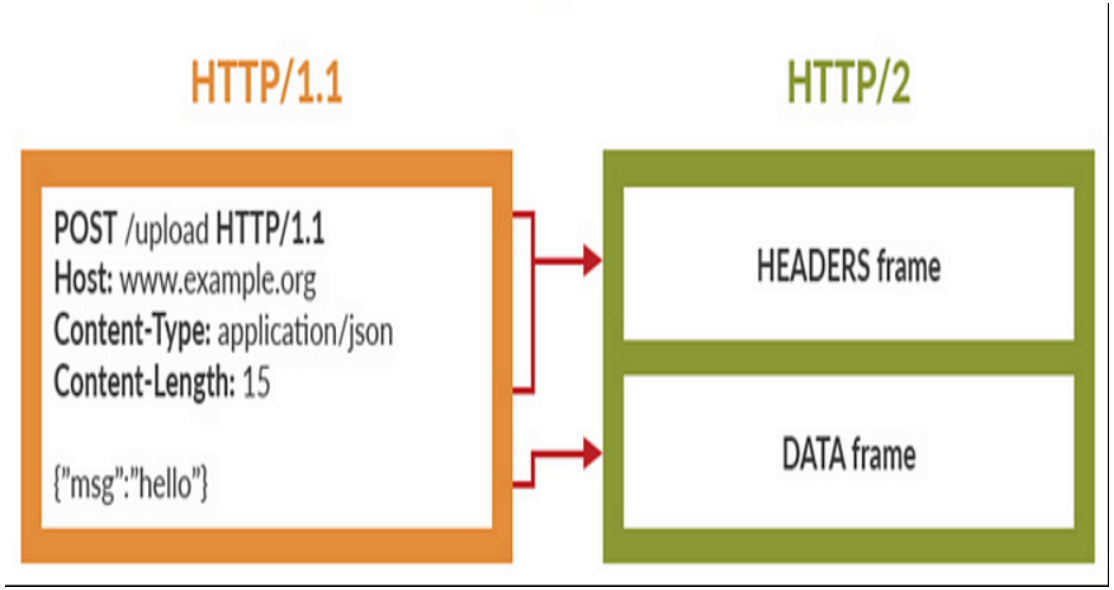
- HTTP/1.x clients need to use multiple connections to achieve concurrency and reduce latency;
- -> Head of line Problem
- HTTP/1.x does not compress request and response headers, causing unnecessary network traffic;
- HTTP/1.x does not allow effective resource prioritization, resulting in poor use of the underlying TCP connection

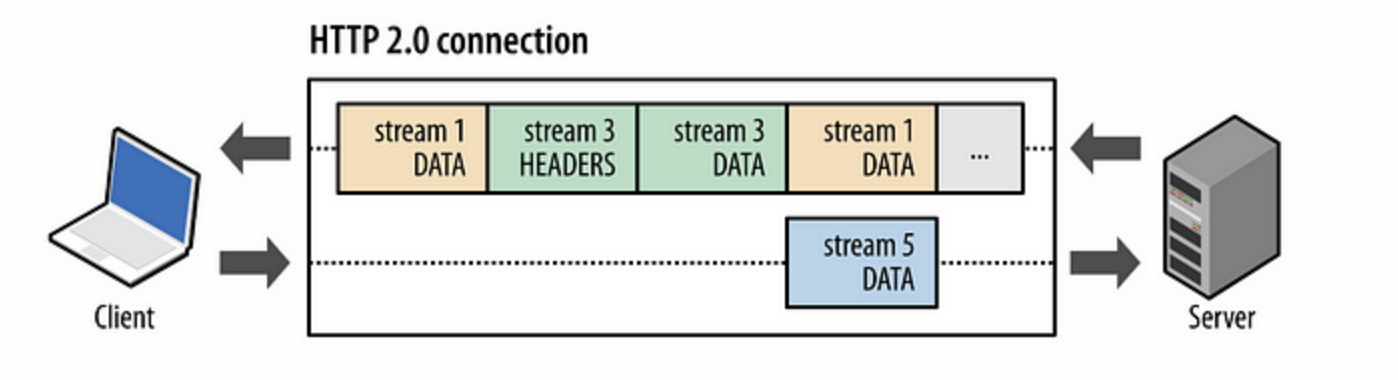
HTTP/2

- *HTTP/2 enables a more efficient use of network resources and a reduced perception of latency by introducing header field compression and allowing multiple concurrent exchanges on the same connection… Specifically, it allows interleaving of request and response messages on the same connection and uses an efficient coding for HTTP header fields. It also allows prioritization of requests, letting more important requests complete more quickly, further improving performance.*
- *The resulting protocol is more friendly to the network, because fewer TCP connections can be used in comparison to HTTP/1.x. This means less competition with other flows, and longer-lived connections, which in turn leads to better utilization of available network capacity. Finally, HTTP/2 also enables more efficient processing of messages through use of binary message framing.*

— Hypertext Transfer Protocol version 2, Draft 17

HTTP/2

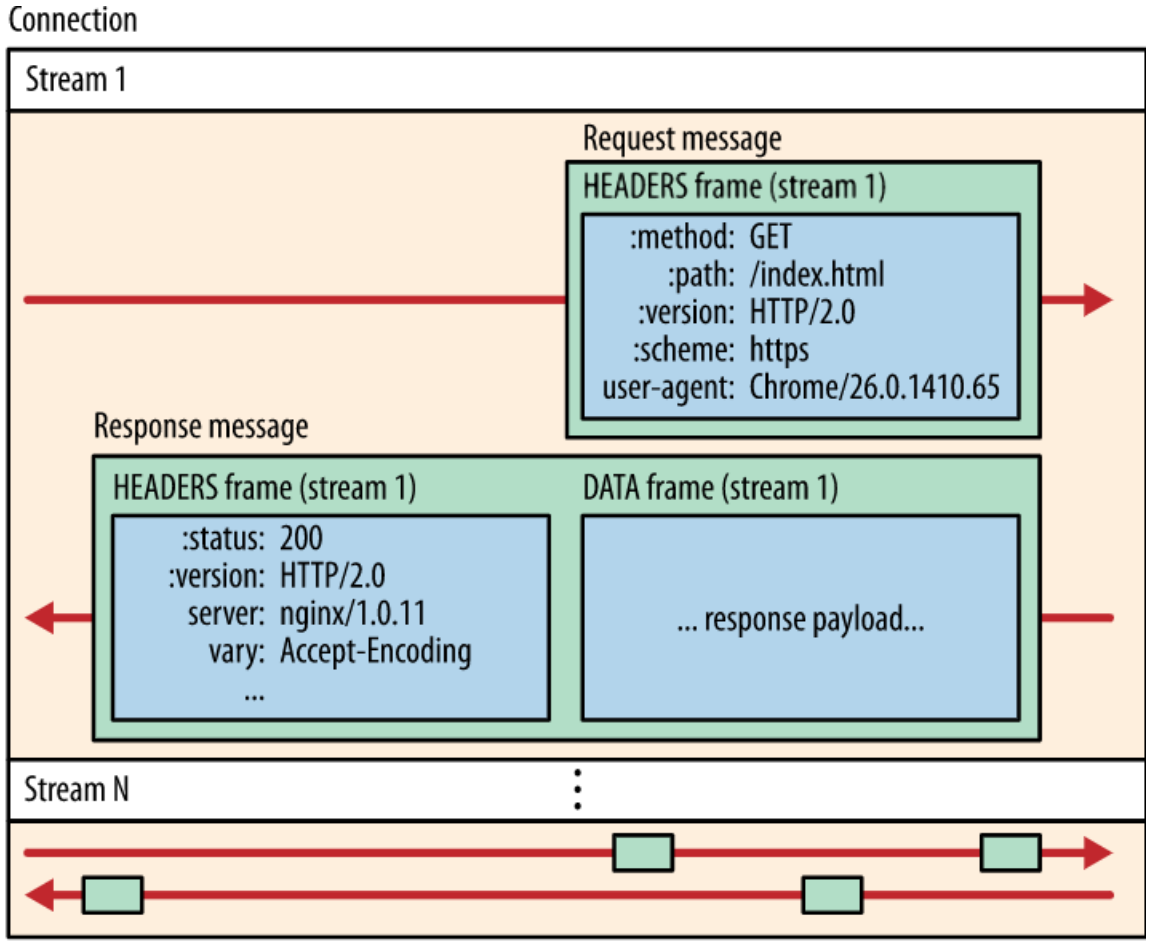




Reference: https://web.dev/performance-http2/

HTTP/2

- Frames -> Messages -> Streams -> A single TCP connection



Reference: https://web.dev/performance-http2/

HTTP/2

- The introduction of the new binary framing mechanism changes how the data is exchanged between the client and server.
- *Stream*: A bidirectional flow of bytes within an established connection, which may carry one or more messages.
- *Message*: A complete frame sequence that maps 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.

HTTP/2

- HTTP/2 does not modify the semantics of HTTP, meaning the core concepts found in HTTP/1.1, such as methods, status codes, URIs, and header fields, remain the same.
- Instead, HTTP/2 modifies how the data is formatted (framed) and transported between the client and server, both of which manage the entire process, and hides protocol complexity within a framing layer. As a result, all existing applications can be delivered over the protocol without modification.

Features and Benefits of HTTP/2

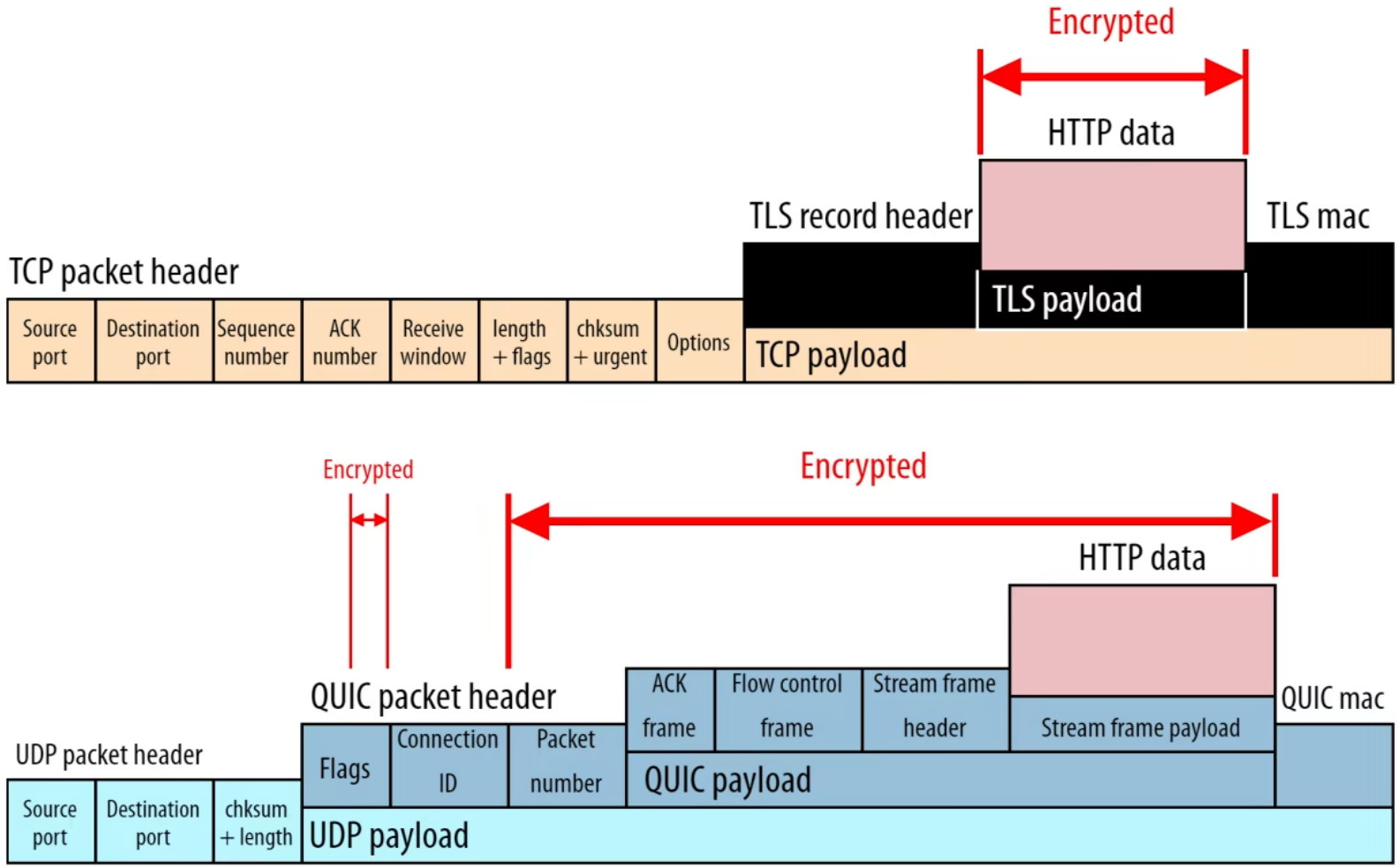
- It’s a **binary** protocol
- It used header **compression** HPACK to reduce the overhead size
- It allows servers to “**push**” responses proactively into client caches instead of waiting for a new request for each resource
- It reduces additional **round trip times (RTT),** making your website load faster without any optimization.
- HTTP/2 supports response **prioritization**.

HTTP/2 also has problems, We now have HTTP/3 as well!

QUIC

- QUIC (Quick UDP Internet Connections) is a new encrypted-by-default Internet transport protocol.
- **Built-in security (and performance)**
- The initial QUIC handshake combines the typical TCP three-way handshake and the TLS 1.3 handshake.
- QUIC handshake only takes a single round-trip between client and server to complete
- It also encrypts additional connection metadata
- Reduced head-of-line blocking
- Improved congestion control

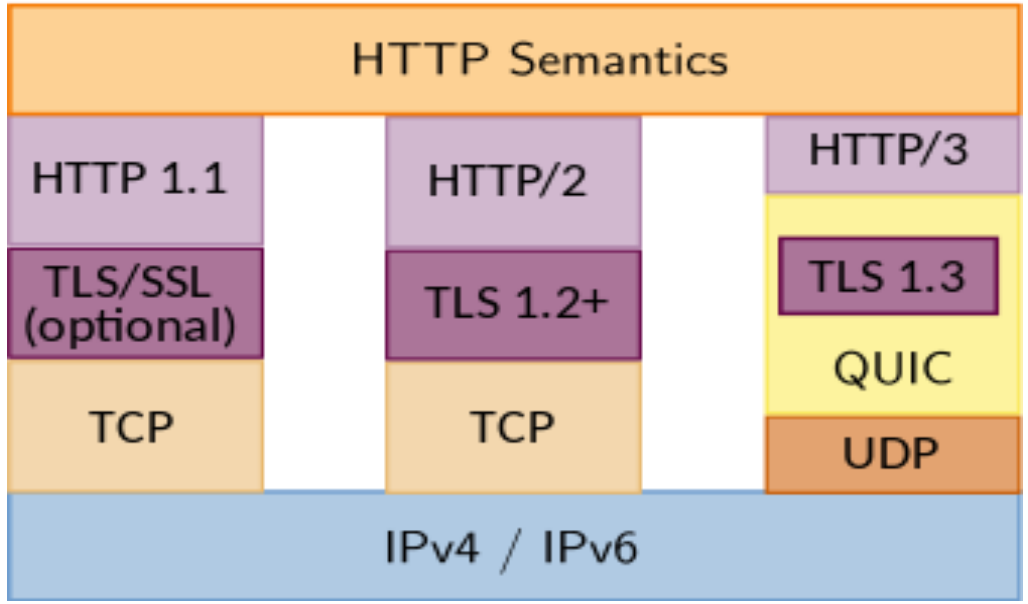
Connection Metadata - encrypted



Benefits - QUIC

- QUIC is more secure for its users.
- QUIC’s connection set-up is faster.
- QUIC can evolve more easily.

HTTP Versions



HTTPS

HTTPS

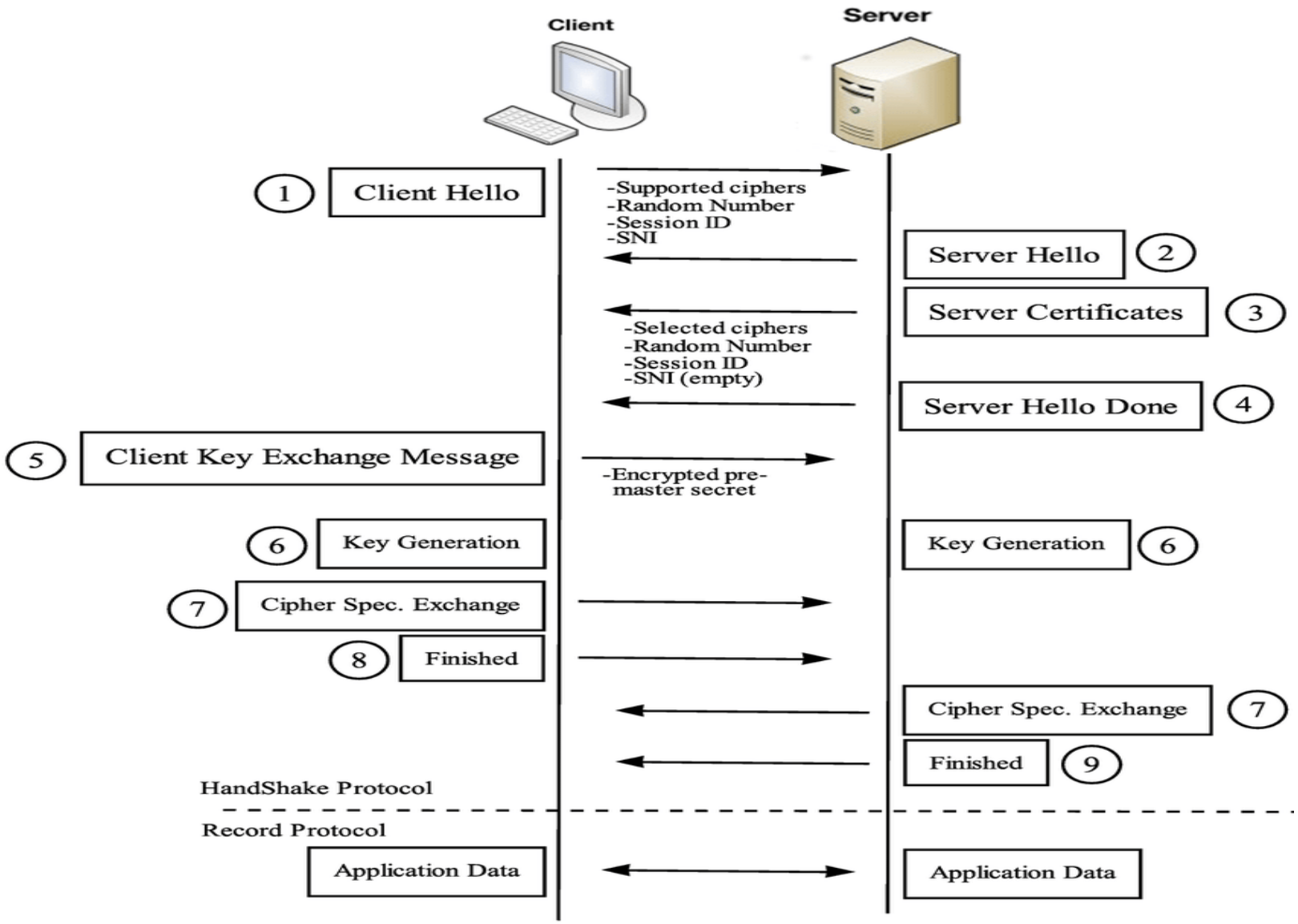
- Hypertext Transfer Protocol Secure is a secure version of HTTP.
- This protocol enables secure communication between a client (e.g. web browser) and a server (e.g. web server) by using encryption.
- HTTPS URLs begin with **https** instead of **http**.
- HTTPS uses a well-known TCP port 443.

- Make sure you always send requests over HTTPS and never ignore invalid certificates.
- HTTPS is the only thing protecting requests from being intercepted or modified!!

TLS handshake

- HTTPS uses **Transport Layer Security (TLS)** protocol or its predecessor **Secure Sockets Layer (SSL)** for encryption.
- The TLS handshake enables the TLS client and server to establish the secret keys with which they communicate.
- Agree on the version of the protocol to use.
- Select cryptographic algorithms.
- Authenticate each other by exchanging and validating digital certificates.
- Use asymmetric encryption techniques to generate a shared secret key, which avoids the key distribution problem.
- SSL or TLS then uses the shared key for the symmetric encryption of messages, which is faster than asymmetric encryption.

The TLS handshake



https://www.researchgate.net/figure/TLS-handshake-protocol_fig1_298065605

TLS Handshake

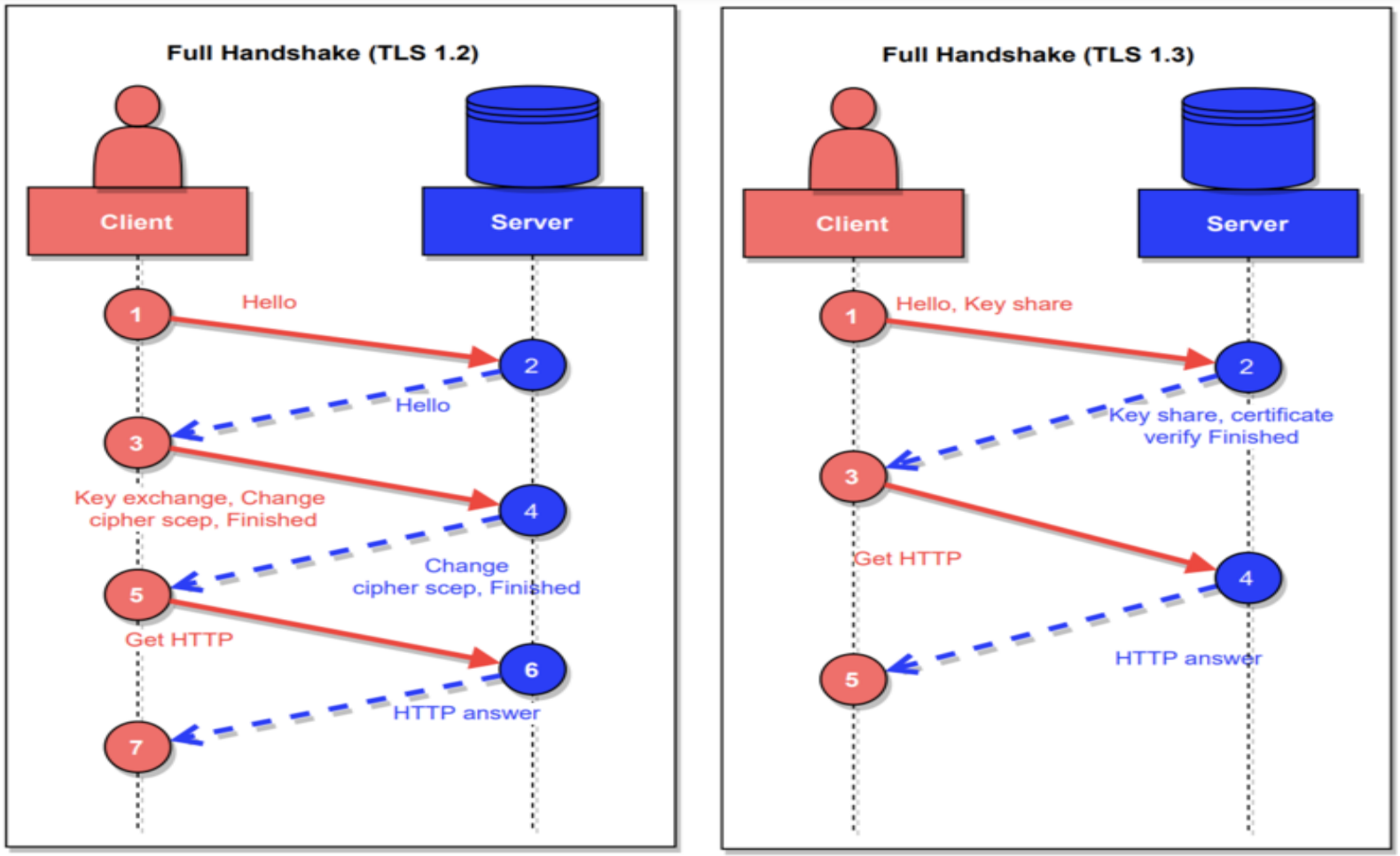


Image Ref: https://commons.m.wikimedia.org/wiki/File:Handshake_in_TLS.png

Certificates

- The HTTPS protocol is based on the server having a public key certificate, sometimes called an SSL certificate.
- When a secure connection is established, the server will send the client its TLS/SSL certificate.
- The client will then check the certificate against a trusted Certificate Authority, essentially validating the server's identity.
- A TLS/SSL certificate essentially binds an identity to a pair of keys which are then used by the server to encrypt as well as sign the data.
- A Certificate Authority is an entity that issues TLS/SSL or Digital certificates

References

- https://www.cloudflare.com/en-gb/learning/performance/http2-vs-http1.1/
- https://ably.com/topic/http3

Thank You!