Networking 101

A Beginner's Guide to OSI Model, HTTP, HTTPS, WS, WSS, and SSL/TLS

www.embedthreads.in

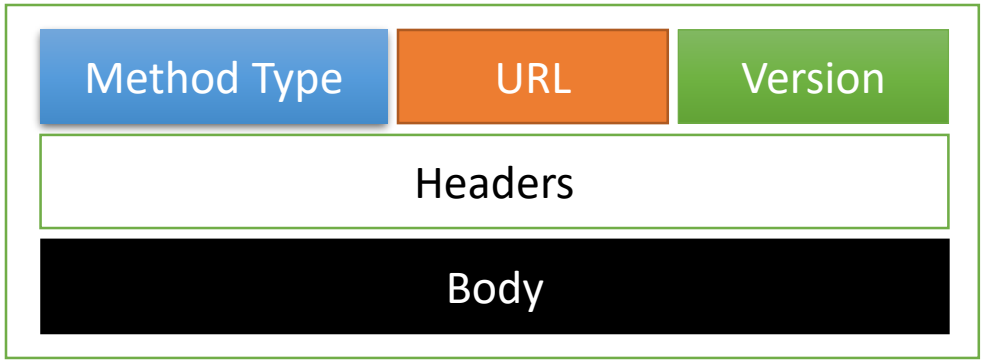
OSI Model: Introduction

Head of Model

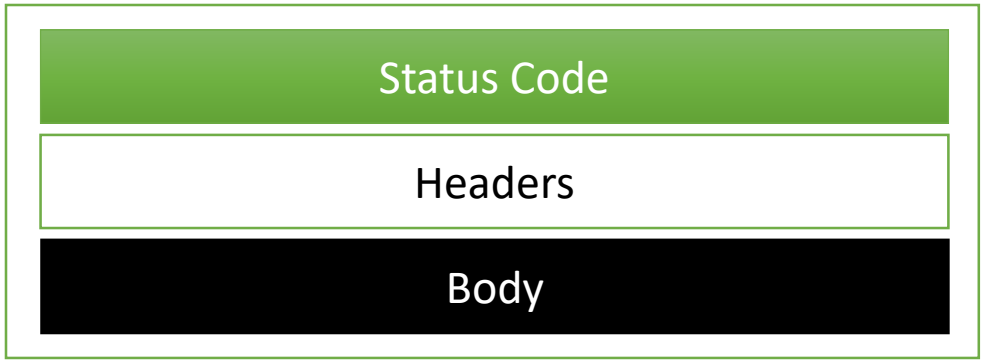
APPLICATION	→	Protocols Definition (HTTP/S, SMTP, FTP and etc.)	
PRESENTATION	→	Encryption, Compression, Conversion	SSL/TLS
SESSION	→	Establish and Terminate Communications	
TRANSPORT	→	Divides data into Segment, Error Correction	TCP/UDP
NETWORK	→	Logical addressing, Routing	IP
Data Link	→	Formats data into frames, MAC addressing	Media
PHYSICAL	→	Electrical Signals, Bits, DAC/ADC	

HTTP: Introduction

- **HTTP and Web Socket** both are communication protocols used in **client-server communication**.
- HTTP is **stateless**, **unidirectional**, and runs on top of TCP. Clients send requests, servers respond, and the connection closes after the response.
- The most commonly used HTTP request methods are **GET, POST, PUT, PATCH, and DELETE**.
- HTTP messages are **encoded in ASCII** and contain information such as protocol version, methods, headers, and message body.
- **Versions**: HTTP1.0, HTTP1.1, HTTP2.0, HTTP3.0

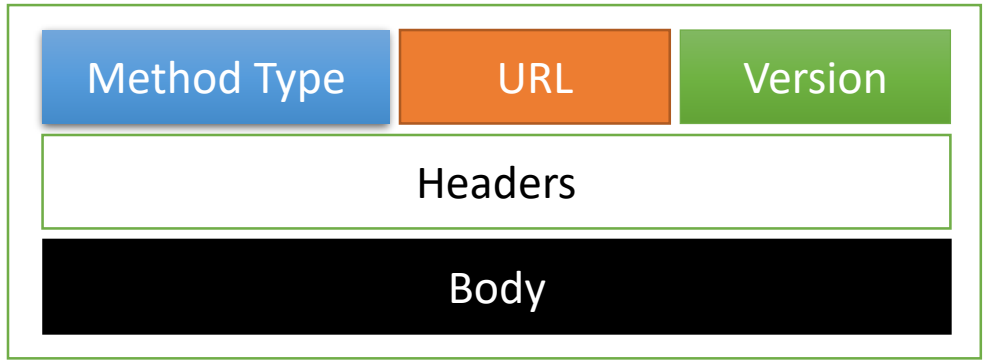


Request Packet Structure



Response Packet Structure

HTTP POST Request & Response



method URI http version

```
POST /create-user HTTP/1.1

Host: localhost:3000
Connection: keep-alive
Content-type: application/json

{ "name": "John", "age: 35 }
```

header

body

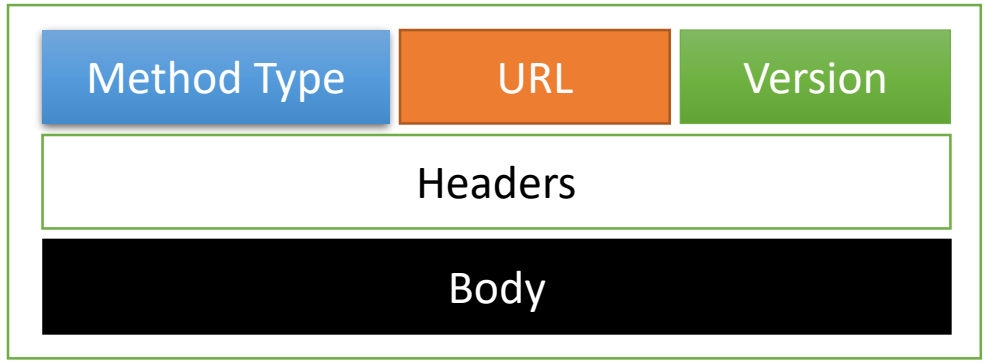
http ver. status

```
HTTP/1.1 200 OK

Date: 2017-01-10 12:28:53 GMT
Server: Apache/2.2.14
Content-type: text/html

<h1>Hello World</h1>
```

header

body

HTTP GET Request & Response



method URI http version

```
GET / users HTTP/1.1

Host: localhost:3000
Connection: keep-alive
Content-type: application/json

Empty
```

header

body

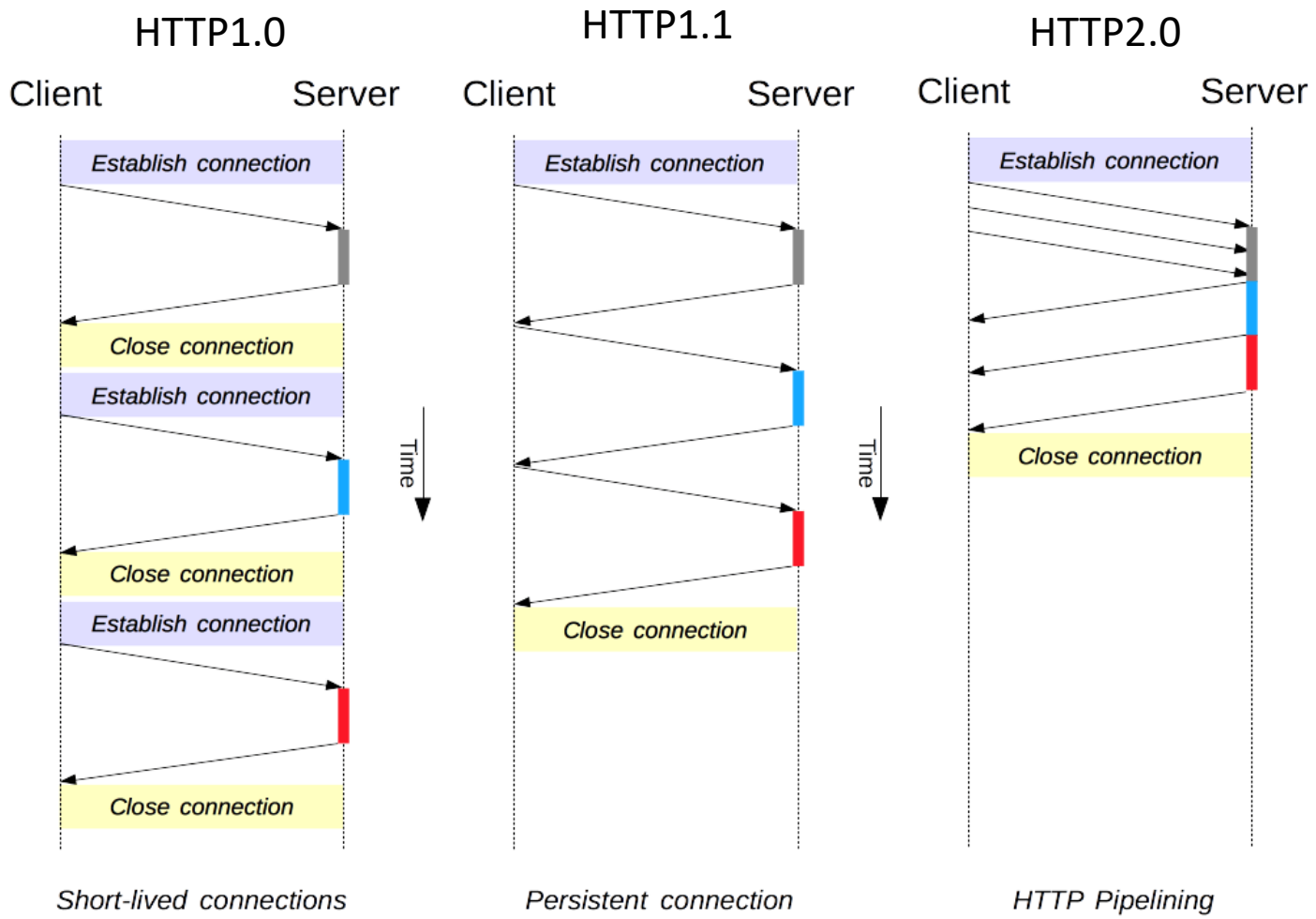
http ver. status

```
HTTP/1.1 200 OK

Date: 2017-01-10 12:28:53 GMT
Server: Apache/2.2.14
Content-type: application/json

{ "name": "John", "age: 35 }
```

header

body

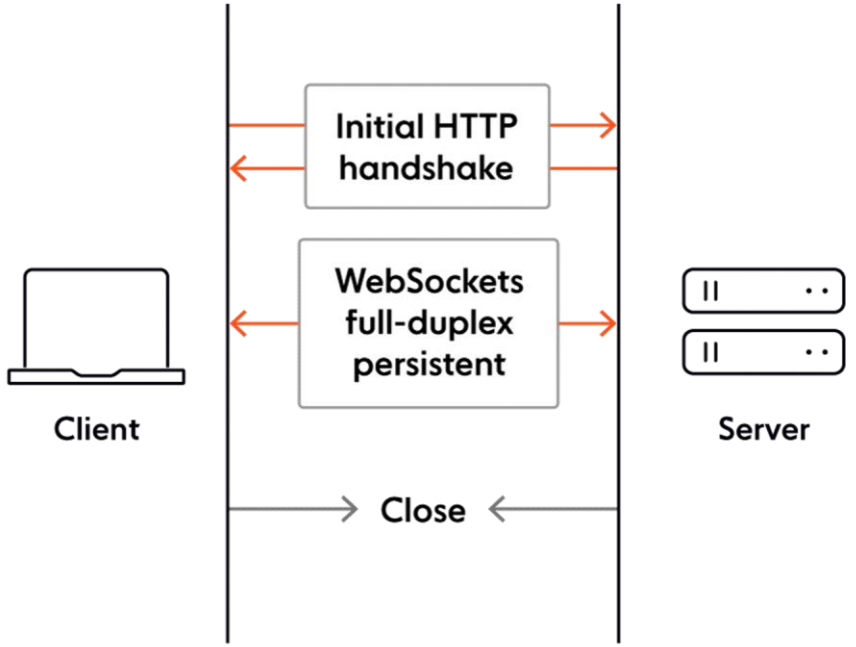
HTTP Different Versions



Why we need Web Socket ???

WebSocket: Introduction

- WebSocket is a **bidirectional**, full-duplex protocol used for client-server communication. Unlike HTTP, it starts with "ws://" or "wss://" and maintains a **stateful** connection.
- WebSocket is initiated through handshaking between the client and server, creating a new connection.
- The status code 101 indicates the protocol switch to WebSocket.
- The connection remains active until terminated by either the client or server.
- WebSocket uses a **lightweight** message format for sending and receiving data of various types, allowing independent transmission without waiting for a response.



When can a web socket be used

- **Real-time web application:** Real-time web application uses a web socket to show the data at the client end, which is continuously being sent by the backend server. In WebSocket, data is continuously pushed/transmitted into the same connection which is already open, that is why WebSocket is faster and improves the application performance.
  For e.g. in a trading website or bitcoin trading, for displaying the price fluctuation and movement data is continuously pushed by the backend server to the client end by using a WebSocket channel.

- **Gaming application:** In a Gaming application, you might focus on that, data is continuously received by the server, and without refreshing the UI, it will take effect on the screen, UI gets automatically refreshed without even establishing the new connection, so it is very helpful in a Gaming application.

- **Chat application:** Chat applications use WebSockets to establish the connection only once for exchange, publishing, and broadcasting the message among the subscribers. It reuses the same WebSocket connection, for sending and receiving the message and for one-to-one message transfer.

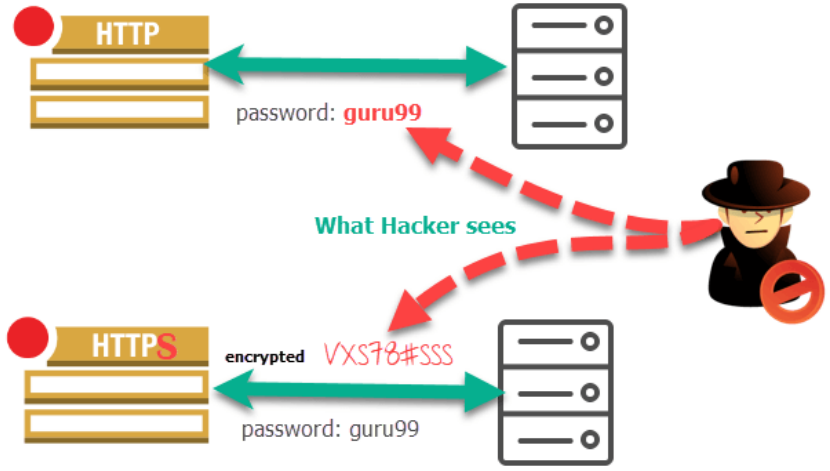
Difference between HTTP and WebSocket

WebSocket Connection	HTTP Connection
• WebSocket is a bidirectional communication protocol that can send the data from the client to the server or from the server to the client by reusing the established connection channel. The connection is kept alive until terminated by either the client or the server.	• The HTTP protocol is a unidirectional protocol that works on top of TCP protocol which is a connection-oriented transport layer protocol, we can create the connection by using HTTP request methods after getting the response HTTP connection get closed.
• Almost all the real-time applications like (trading, monitoring, notification) services use WebSocket to receive the data on a single communication channel.	• Simple RESTful application uses HTTP protocol which is stateless.
• All the frequently updated applications used WebSocket because it is faster than HTTP Connection.	• When we do not want to retain a connection for a particular amount of time or reuse the connection for transmitting data; An HTTP connection is slower than WebSockets.

Why we need secure communication ???

Secure communication is crucial

- **Data Privacy:** It ensures the confidentiality of sensitive information, protecting it from unauthorized access.
- **User Trust:** Secure communication builds confidence among users, fostering trust in web applications and online services.
- **Cyber Threat Mitigation:** Secure channels mitigate risks like data tampering, session hijacking, and unauthorized access.
- **Compliance:** Many industries require secure communication to meet regulatory and compliance standards.



SSL and TLS: Introduction

- SSL (Secure Sockets Layer) and TLS (Transport Layer Security) are **cryptographic** protocols that provide secure communication over the internet.
- Apply the **encryption algorithm** to the plaintext data and the encryption key. The encryption algorithm performs a series of mathematical operations on the **plaintext**, transforming it into **ciphertext.**
- TLS used two types of encryption techniques:

    1) Symmetric (Single Key)

    2) Asymmetric (Public and Private Keys)
- Symmetric **encryption algorithms are: AES, RC4, Blowfish and etc.**
- Asymmetric **encryption algorithms are: RSA, Diffie Hellman, Elgamal, and etc.**
- **TLS Versions:** TLS1.2 and TLS1.3 (most popular)
- WebSocket and HTTP uses **port 80** for unsecured connections (ws:// and http://) and **port 443** for secured connections (wss:// and https://).

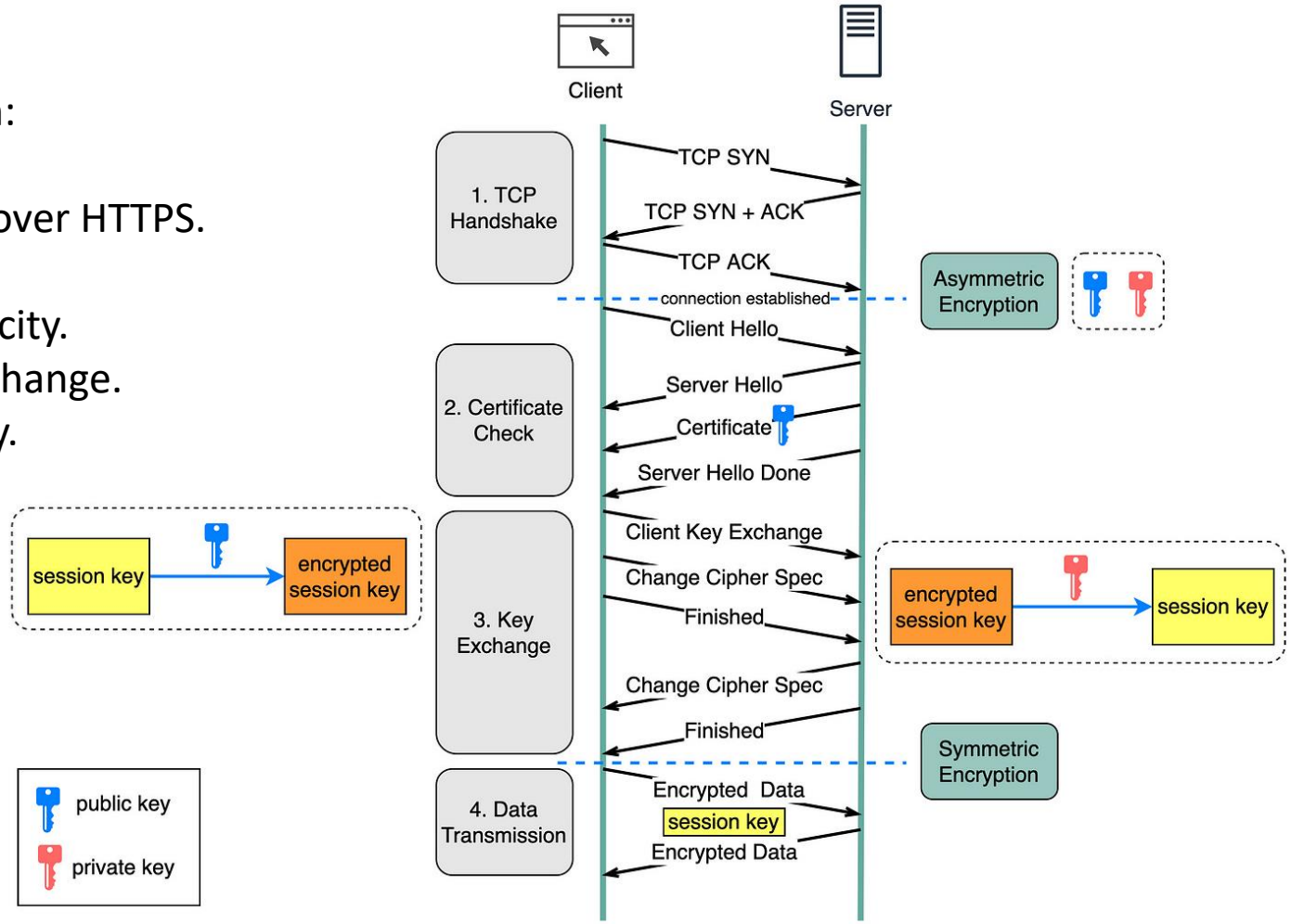
SSL / TLS Steps

1. The client (browser) and the server establish a TCP connection.
2. The client sends a “client hello” to the server. The message contains a set of necessary encryption algorithms (cipher suites) and the latest TLS version it can support. The server responds with a “server hello” so the browser knows whether it can support the algorithms and TLS version. The server then sends the SSL certificate to the client. The certificate contains the public key, hostname, expiry dates, etc. The client validates the certificate.
3. After validating the SSL certificate, the client generates a session key and encrypts it using the public key. The server receives the encrypted session key and decrypts it with the private key.
4. Now that both the client and the server hold the same session key (symmetric encryption), the encrypted data is transmitted in a secure bi-directional channel.

Establishing a Secure HTTPS Connection

Steps to establish a secure HTTPS connection:

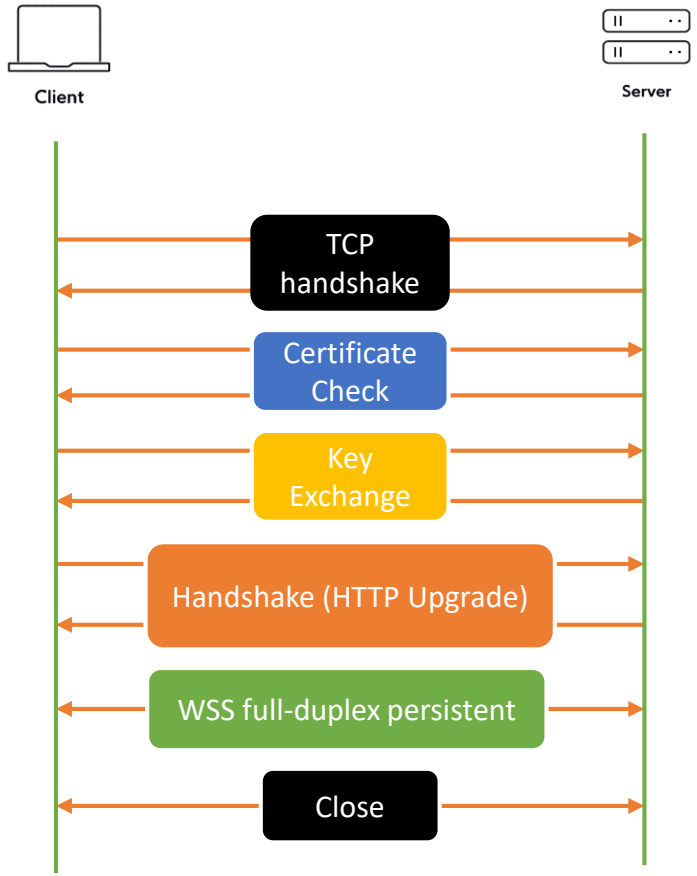
1. Client initiates a connection to the server over HTTPS.
2. Server presents its digital certificate.
3. Client validates the certificate for authenticity.
4. Client and server perform a secure key exchange.
5. Data is encrypted and transmitted securely.
6. Connection is gracefully terminated.



Establishing a Secure WSS Connection

Steps to establish a secure WebSocket (WSS) connection:

1.Establish a secure HTTPS connection.
2.Client sends a WebSocket handshake request.
3.Server validates and responds to the handshake.
4.Handshake confirmation and connection establishment.
5.Bidirectional data transfer and communication.
6.Closure initiation and acknowledgment.



Secured Connection

Thank You!