Tecnologie e applicazioni web

WebSocket

Filippo Bergamasco (filippo.bergamasco@unive.it)
http://www.dais.unive.it/~bergamasco/
DAIS - Università Ca'Foscari di Venezia
Academic year: 2023/2024

WebSocket

Protocol that allows a simple **full-duplex** communication using an underlying TCP/IP connection.

Designed to be compatible with HTTP

- Same ports
- Initial handshake based on HTTP
- Proxy support

Full-duplex

Unlike the HTTP protocol, once a WebSocket connection has been established, the exchange of messages can take place indifferently and simultaneously between client and server

Overcomes the HTTP request-response model

WebSocket

Can be used inside the web browser or standalone.

WebSocket is designed to allow the transport of messages in a **bidirectional way** in web-based applications (therefore, within the browser)

WebSocket

Two high-level components:

1. Handshake protocol, based on HTTP, to negotiate the connection parameters and establish a communication channel
2. A framing mechanism to transfer binary/ascii data with the following features:
   a. Very low overhead
   b. Low latency

WebSocket handshake

The WebSocket communication channel is established from an existing HTTP connection.

- HTTP `Upgrade` header used to negotiate a protocol switch
- This mechanism allows the crossing of proxies supporting the WebSocket protocol
- Designed to prevent malicious attacks

Handshake: client->server

```
GET /socket HTTP/1.1
Host: thirdparty.com
Origin: http://example.com
Connection: Upgrade
Upgrade: websocket
Sec-WebSocket-Version: 13
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Protocol: appProtocol,appProtocol-v2
Sec-WebSocket-Extensions: x-webkit-deflate-
message, x-custom-extension
```

Handshake begins with a GET request to a specific server's resource

Handshake: client->server

Client asks for a protocol switch from HTTP to WebSocket

```
GET /socket HTTP/1.1
Host: thirdparty.com
Origin: http://example.com
Connection: Upgrade
Upgrade: websocket
Sec-WebSocket-Version: 13
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Protocol: appProtocol,appProtocol-v2
Sec-WebSocket-Extensions: x-webkit-deflate-
message, x-custom-extension
```

Handshake: client->server

Client's supported WebSocket version

```
GET /socket HTTP/1.1
Host: thirdparty.com
Origin: http://example.com
Connection: Upgrade
Upgrade: websocket
Sec-WebSocket-Version: 13
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Protocol: appProtocol,appProtocol-v2
Sec-WebSocket-Extensions: x-webkit-deflate-
message, x-custom-extension
```

Handshake: client->server

```
GET /socket HTTP/1.1
Host: thirdparty.com
Origin: http://example.com
Connection: Upgrade
Upgrade: websocket
Sec-WebSocket-Version: 13
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Protocol: appProtocol,appProtocol-v2
Sec-WebSocket-Extensions: x-webkit-deflate-
message, x-custom-extension
```

A random string encoded in base-64 is used to:

- Verify if the server supports the protocol
- Invalidate proxy caches and avoid duplicate handshakes

Handshake: client->server

```
GET /socket HTTP/1.1
Host: thirdparty.com
Origin: http://example.com
Connection: Upgrade
Upgrade: websocket
Sec-WebSocket-Version: 13
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Protocol: appProtocol,appProtocol-v2
Sec-WebSocket-Extensions: x-webkit-deflate-
message, x-custom-extension
```

List of sub-protocols and extension that might be used

Handshake: client<-server

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Access-Control-Allow-Origin: http://example.com
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: appProtocol-v2
Sec-WebSocket-Extensions: x-custom-extension
```

Response code to acknowledge the protocol switch

Handshake: client<-server

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Access-Control-Allow-Origin: http://example.com
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: appProtocol-v2
Sec-WebSocket-Extensions: x-custom-extension
```

Hash of the key sent on the previous request + a predefined string depending by the protocol

Handshake: client<-server

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Access-Control-Allow-Origin: http://example.com
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: appProtocol-v2
Sec-WebSocket-Extensions: x-custom-extension
```

Sub-protocol to use and supported extensions

Handshake

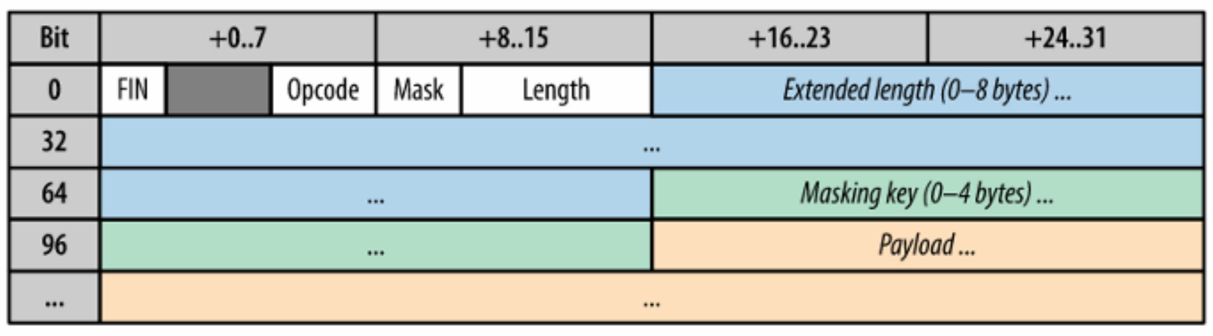
After the exchange of request-response messages:

- TCP (or SSL/TLS) connection is kept open
- New messages are exchanged according to WebSocket protocol (HTTP is not used anymore)

WebSocket messages

- Protocol allows the exchange of binary or text messages (UTF-8) of arbitrary length
- Communication is full-duplex. Both client and server can **pre-emptively** send a message when needed. The other peer is notified when a new message arrives
- Messages are divided into frames, each frame is sent sequentially and reassembled at the destination

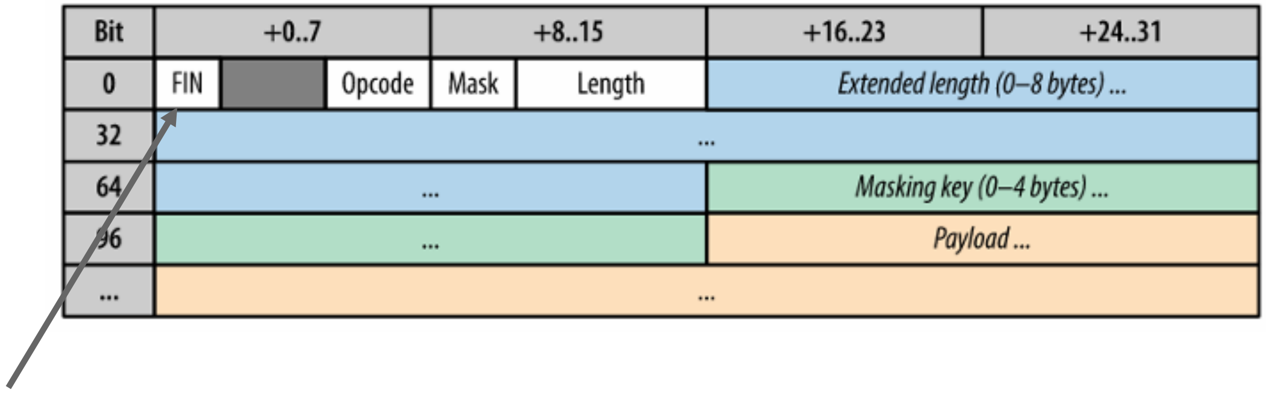
Frames



Bit	+0..7	+8..15	+16..23	+24..31
0	FIN ▨ Opcode	Mask Length	*Extended length (0–8 bytes) ...*	
32	...			
64	...		*Masking key (0–4 bytes) ...*	
96	...		*Payload ...*	
...	...			

Variable frame overhead (2 to 10 bytes). All messages sent by the client contain a masking key (0-4 bytes) , causing an additional overhead of 6 to 14 bytes

Frames

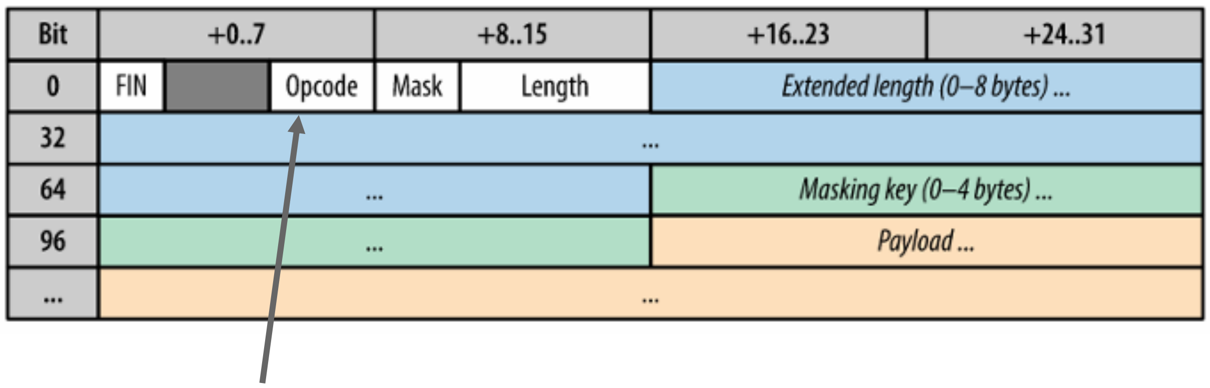


Bit	+0..7	+8..15	+16..23	+24..31
0	FIN ▨ Opcode	Mask Length	*Extended length (0–8 bytes) ...*	
32	...			
64	...		*Masking key (0–4 bytes) ...*	
96	...		*Payload ...*	
...	...			

0: Some other frames are needed to complete the message

1: Message is completed with this frame

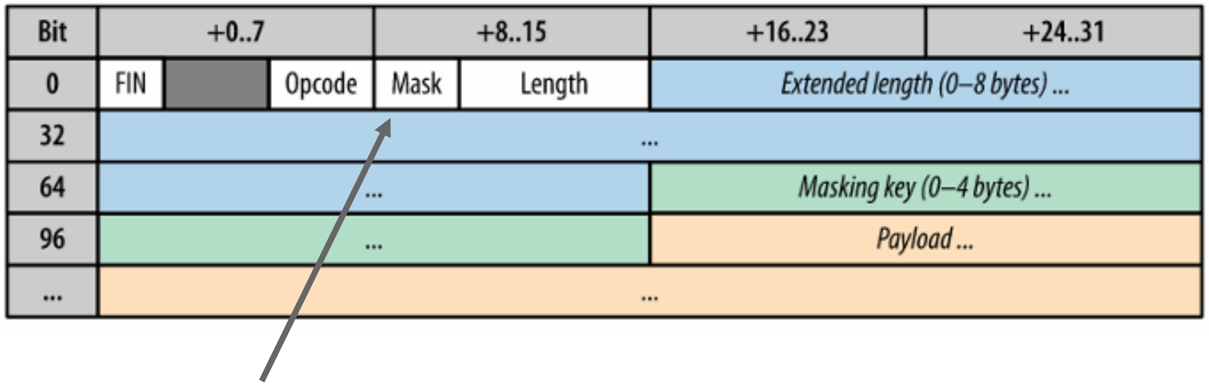
Frames



Bit	+0..7	+8..15	+16..23	+24..31
0	FIN | ▨ | Opcode	Mask | Length	*Extended length (0–8 bytes) ...*	
32	...			
64	...		*Masking key (0–4 bytes) ...*	
96	...		*Payload ...*	
...	...			

Message type: text (1), binary (2), close (8), ping (9), pong (10)

Frames

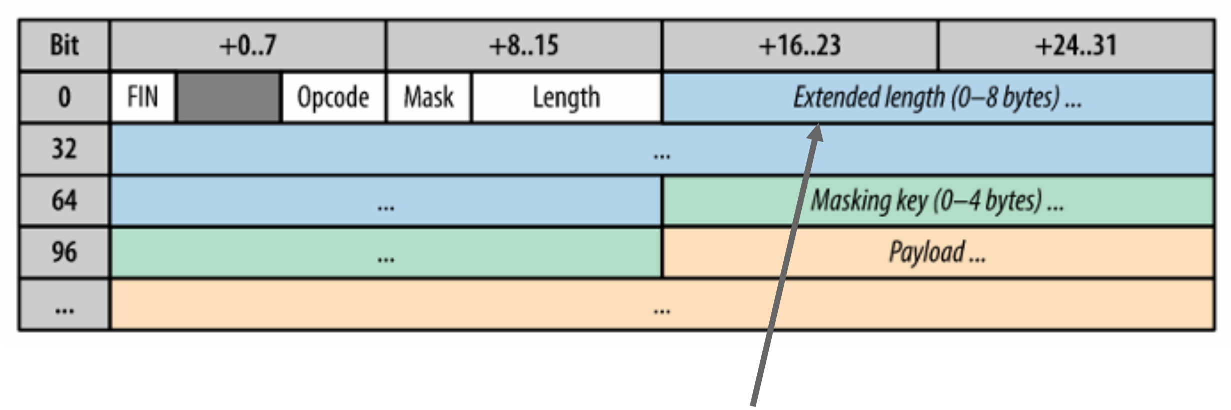


Bit	+0..7	+8..15	+16..23	+24..31
0	FIN ▨ Opcode	Mask Length	*Extended length (0–8 bytes) ...*	
32	...			
64	...		*Masking key (0–4 bytes) ...*	
96	...		*Payload ...*	
...	...			

0: Frame is NOT masked

1: Frame is masked

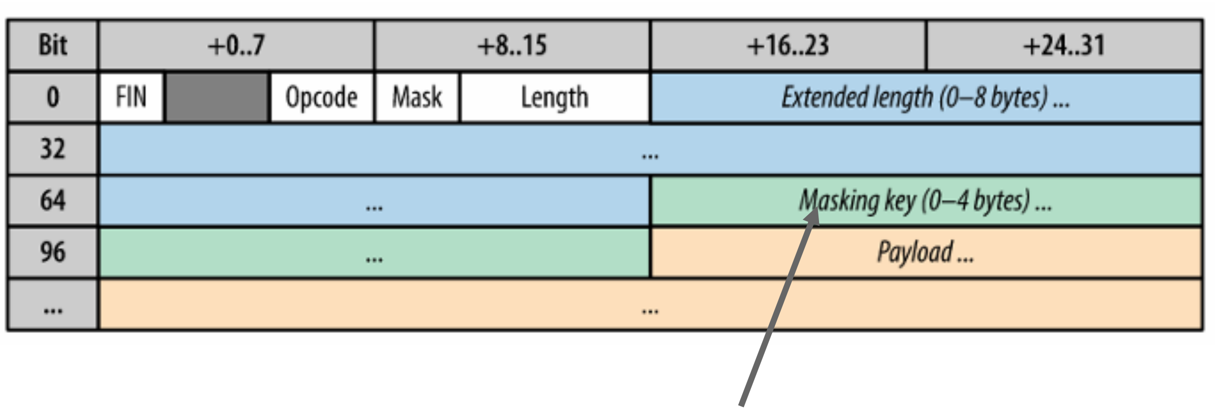
Frames



Bit	+0..7	+8..15	+16..23	+24..31
0	FIN ▨ Opcode	Mask Length	*Extended length (0–8 bytes) ...*	
32	...			
64	...		*Masking key (0–4 bytes) ...*	
96	...		*Payload ...*	
...	...			

Message length (one or more bytes)

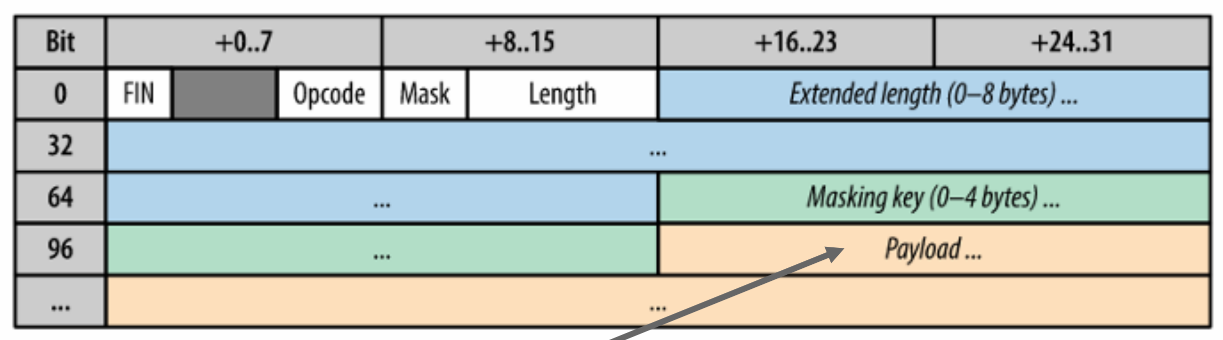
Frames



Bit	+0..7	+8..15	+16..23	+24..31
0	FIN ▨ Opcode	Mask Length	*Extended length (0–8 bytes) ...*	
32	...			
64	...		*Masking key (0–4 bytes) ...*	
96	...		*Payload ...*	
...	...			

Payloads of all client-initiated messages are masked (XOR) with this key to avoid «cache poisoning» attacks.

Frames



Bit	+0..7	+8..15	+16..23	+24..31
0	FIN ▨ Opcode	Mask Length	*Extended length (0–8 bytes) ...*	
32		...		
64	...		*Masking key (0–4 bytes) ...*	
96	...		*Payload ...*	
...		...		

Message payload

Framing

Messages are framed for two reasons:

1. Messages can be transferred without knowing their size in advance (infinite streams are also possible)
2. Frames belonging to different messages can be interleaved to reduce the latency (higher priority can be given to small messages)

WebSocket in JavaScript

WebSocket is supported in all modern web browsers.

Socket.io library simplifies the development of WebSocket applications in JavaScript

https://socket.io/

Allows the asynchronous exchange of «events» between client and server and vice versa

https://socketio-whiteboard-zmx4.herokuapp.com/