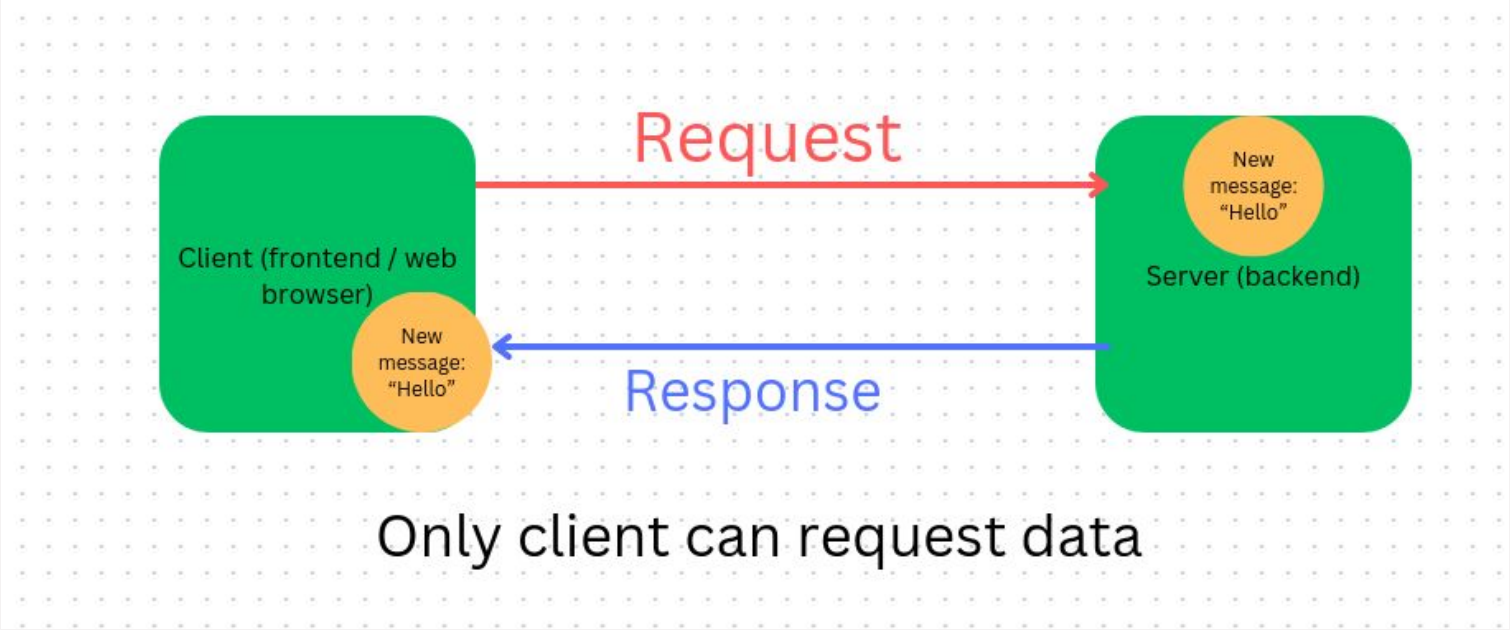
Homework #4

WebSocket

Websockets

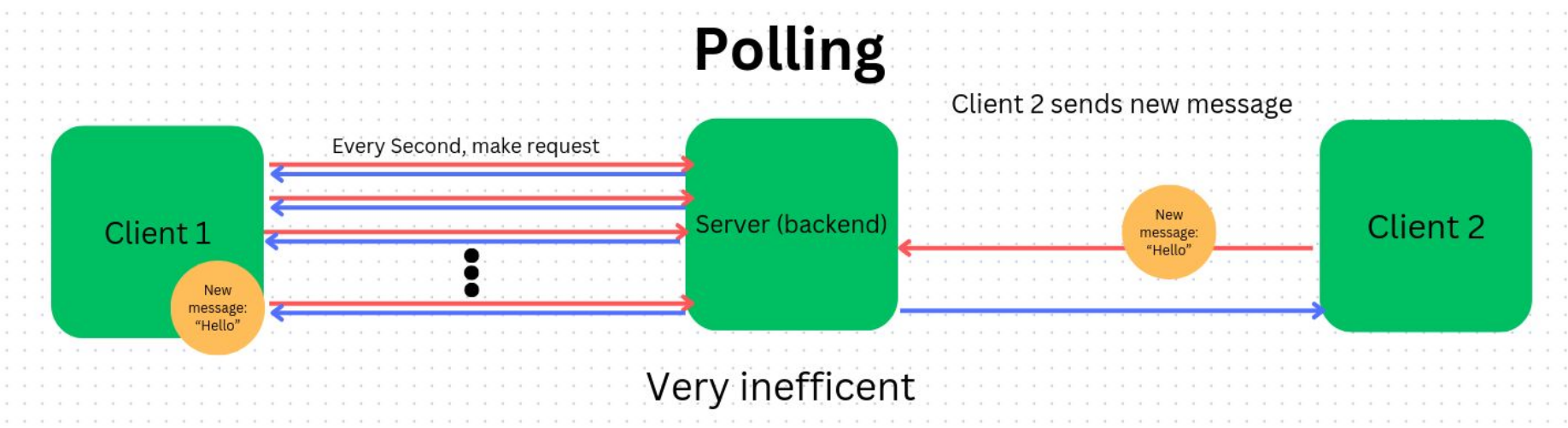
- Websocket Handshake
- Parse frames
- Echo
- Drawing Board
- DMs
- WebRTC

Websockets



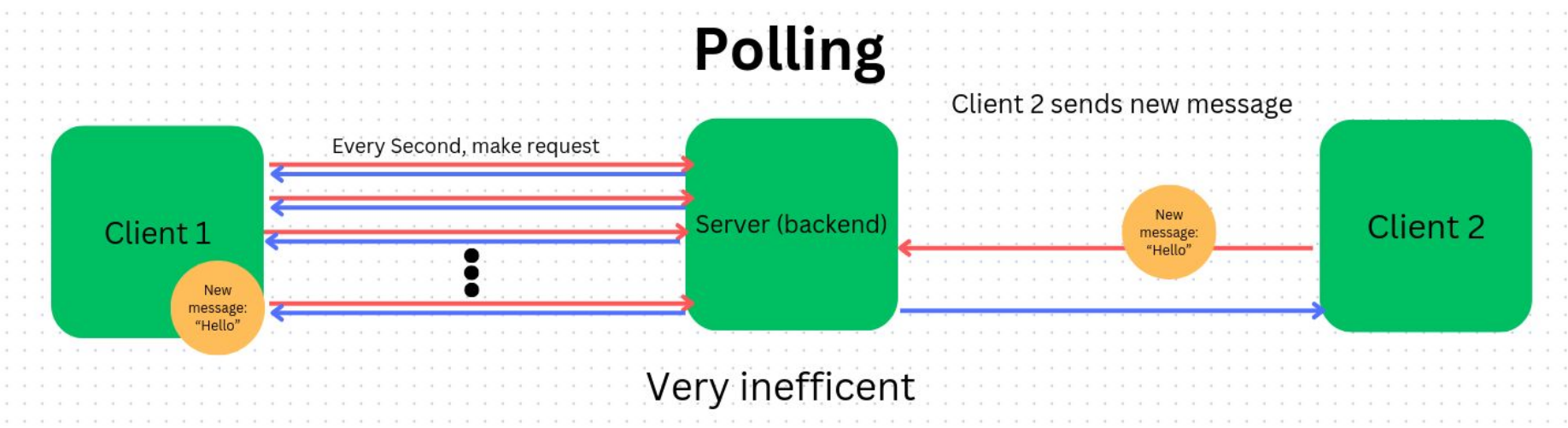
Http works on the idea of request and responses. But client that wants information has to request (has to ask for it first)

Websockets



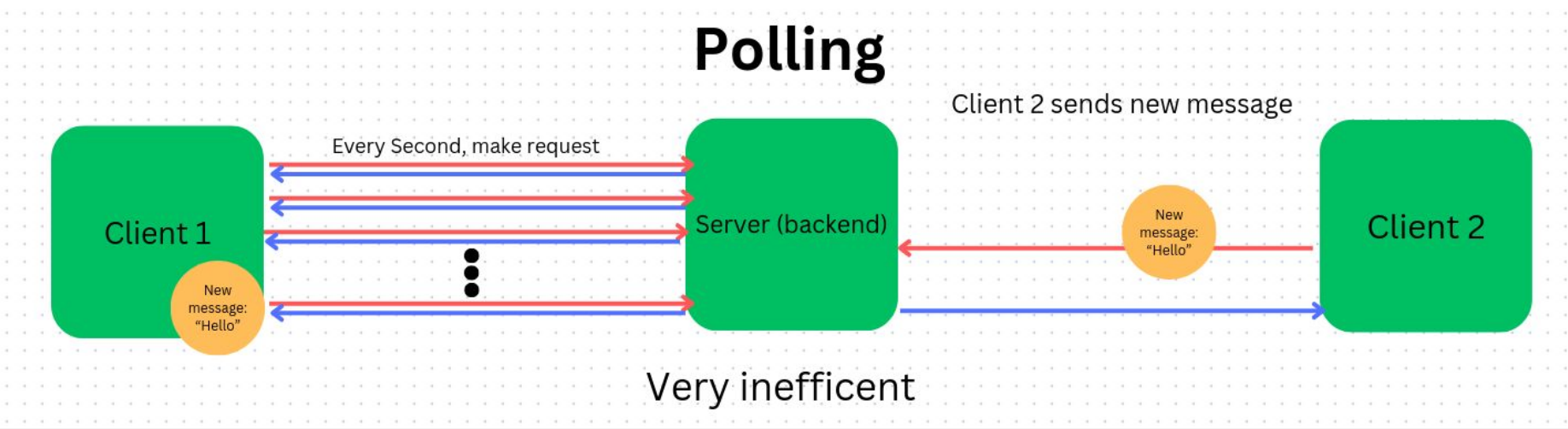
What if we have chat feature where we want information to be sent to client even when it doesn't ask for it. This is what your homework does

Websockets



Why can't the sever just send a update right away. That is just what http is limited to, server is not allowed to send events to client

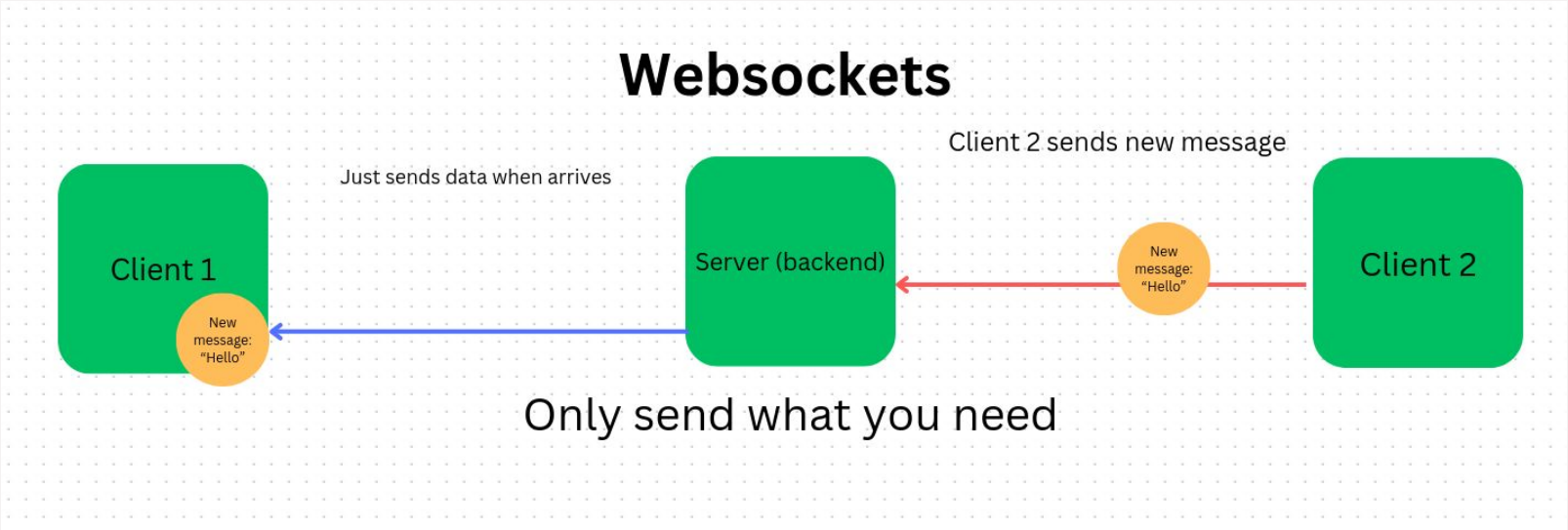
Websockets



Can have massive overhead with something like twitch, having all these users requesting messages.

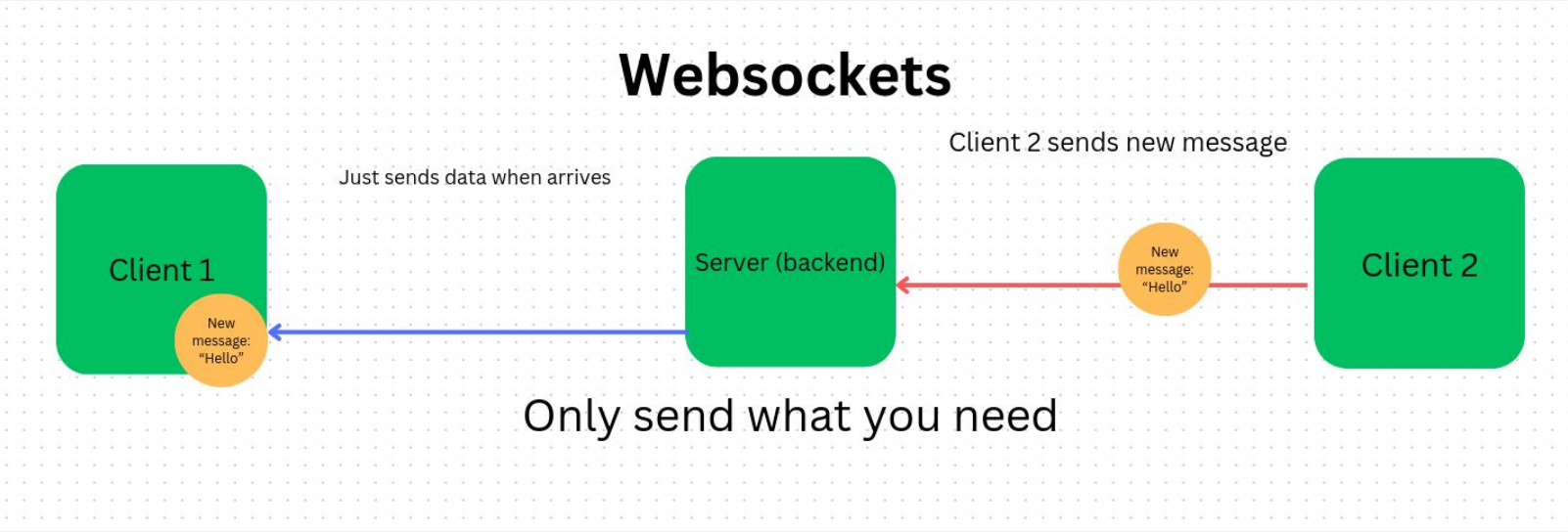
- Estimated $4-6 million a month to run twitch, 2.4 million average viewers
- Only want to have exactly as many requests as needed

Websockets



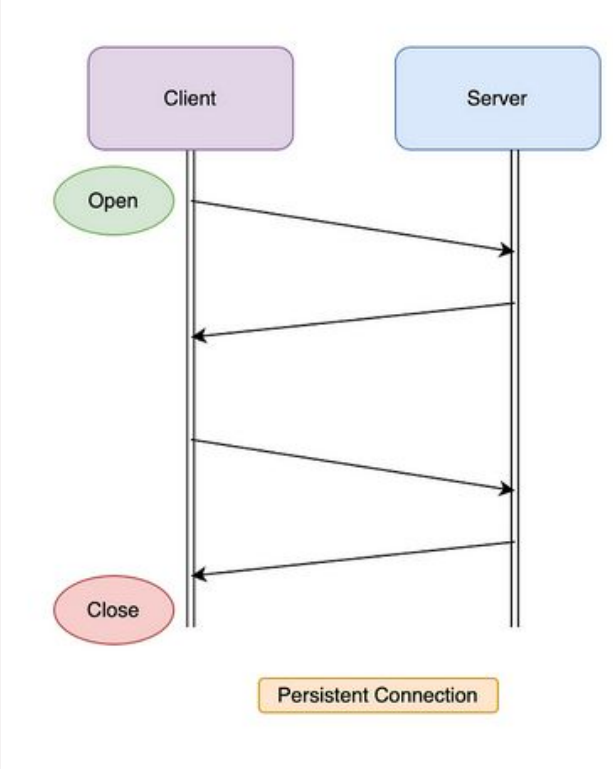
To solve this, we use websockets

Websockets

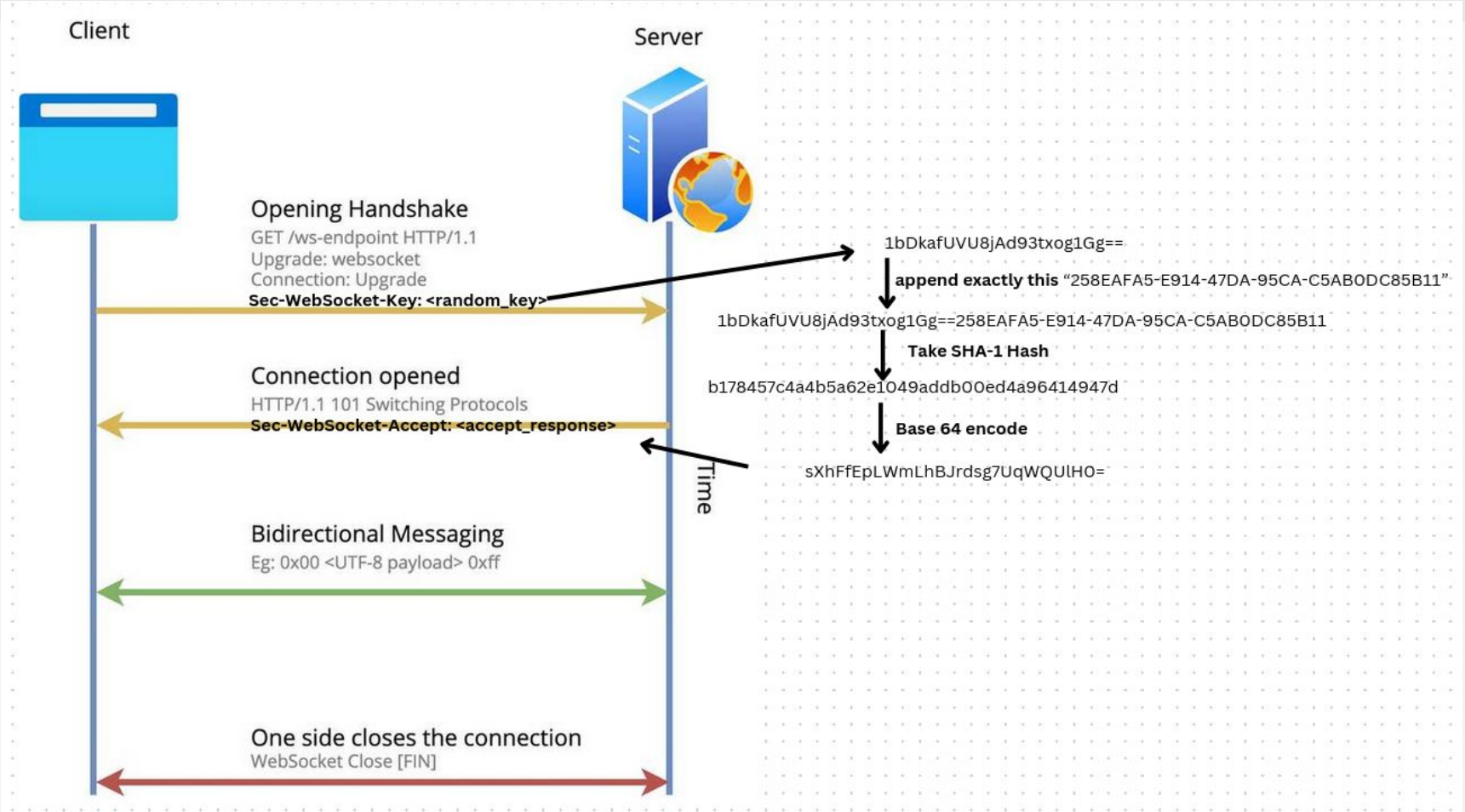


This works because websockets and HTTP both use TCP. TCP allows for bidirectonal communication, websockets takes advantage of that.

Websockets

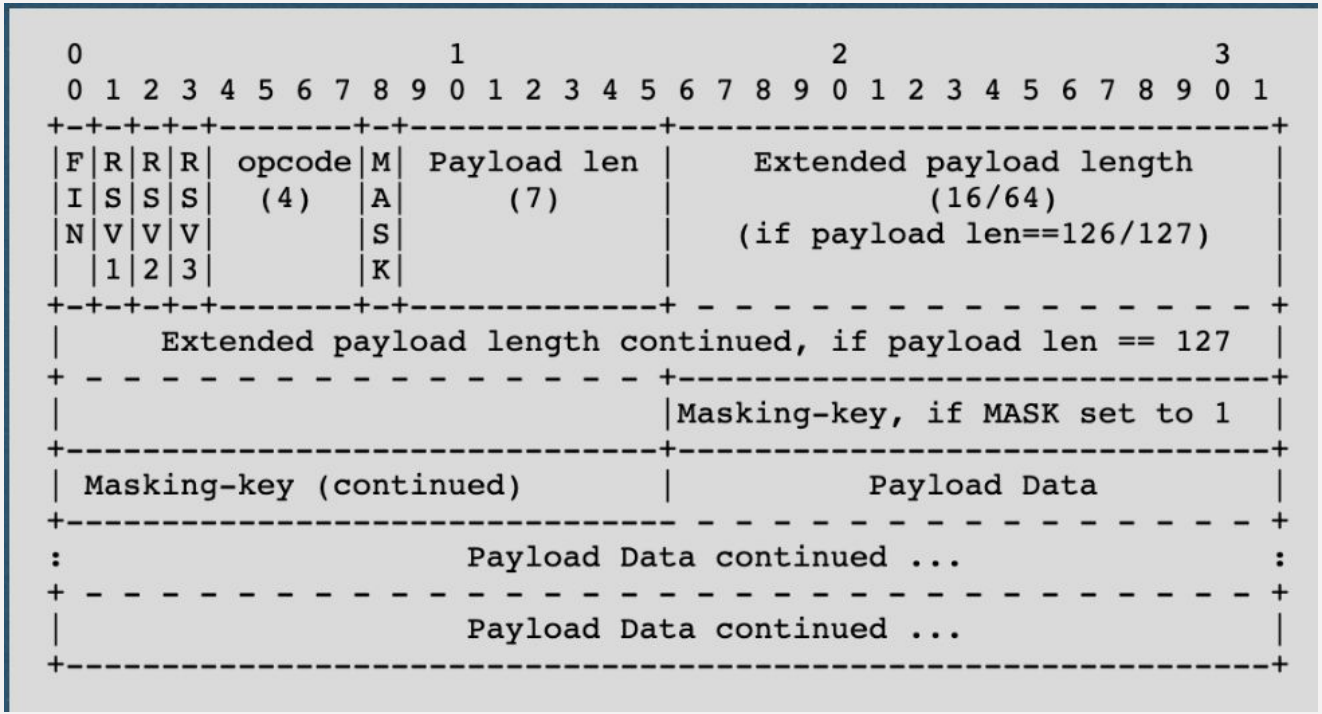


HTTP closes right away
Get your data (html, css, js) then it closes TCP connection



Not going to close the TCP connection the user requested with

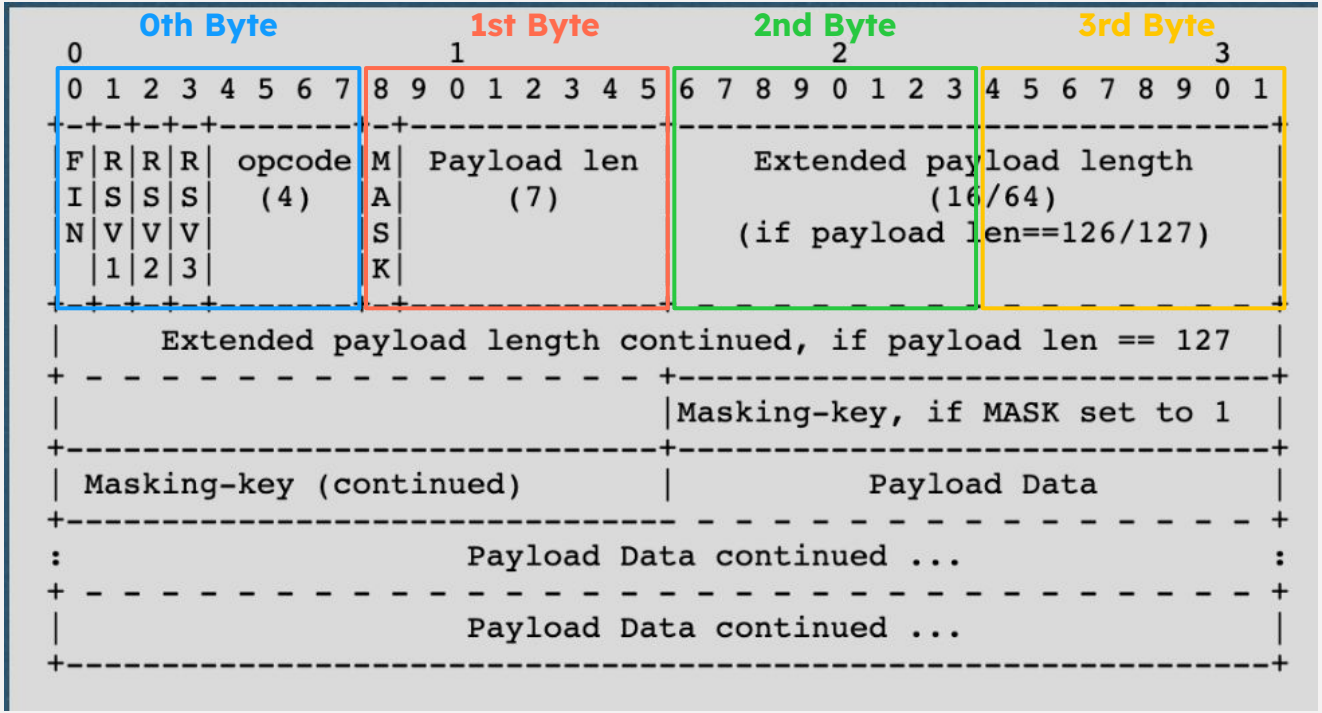
Websockets



Then all from there on this bidirectional connection sends frames.

ALL IN BITS

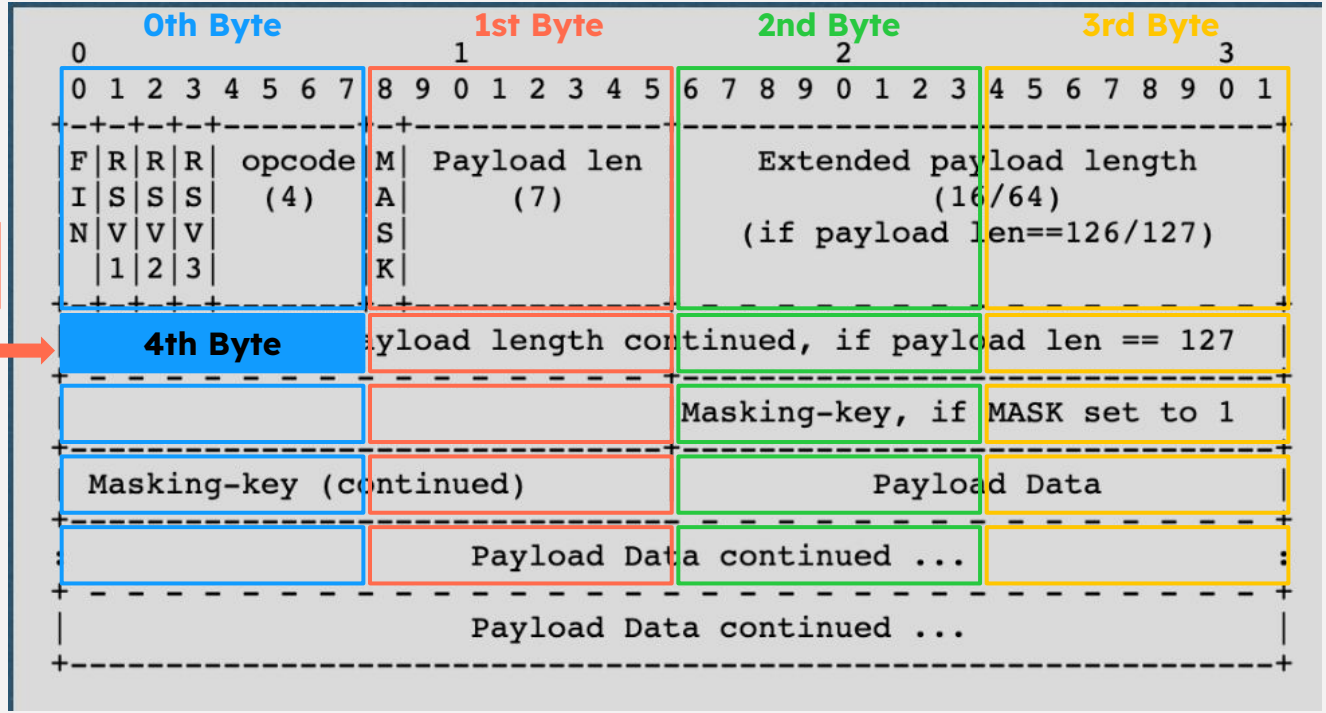
Websockets



The diagram from the [websockets rfc] numbers represent each **bit** not byte (Each row **32 bits** or **4 bytes**)

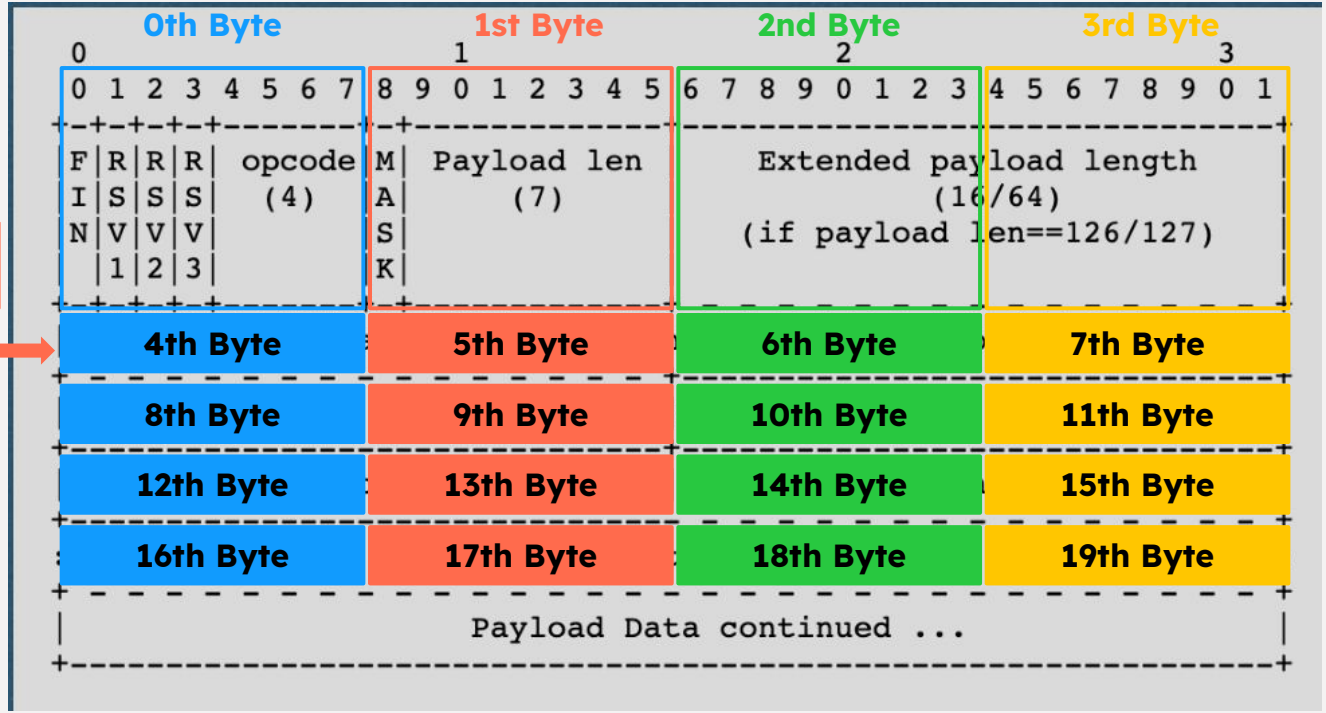
Websockets

Each row represents another set of 4 bytes



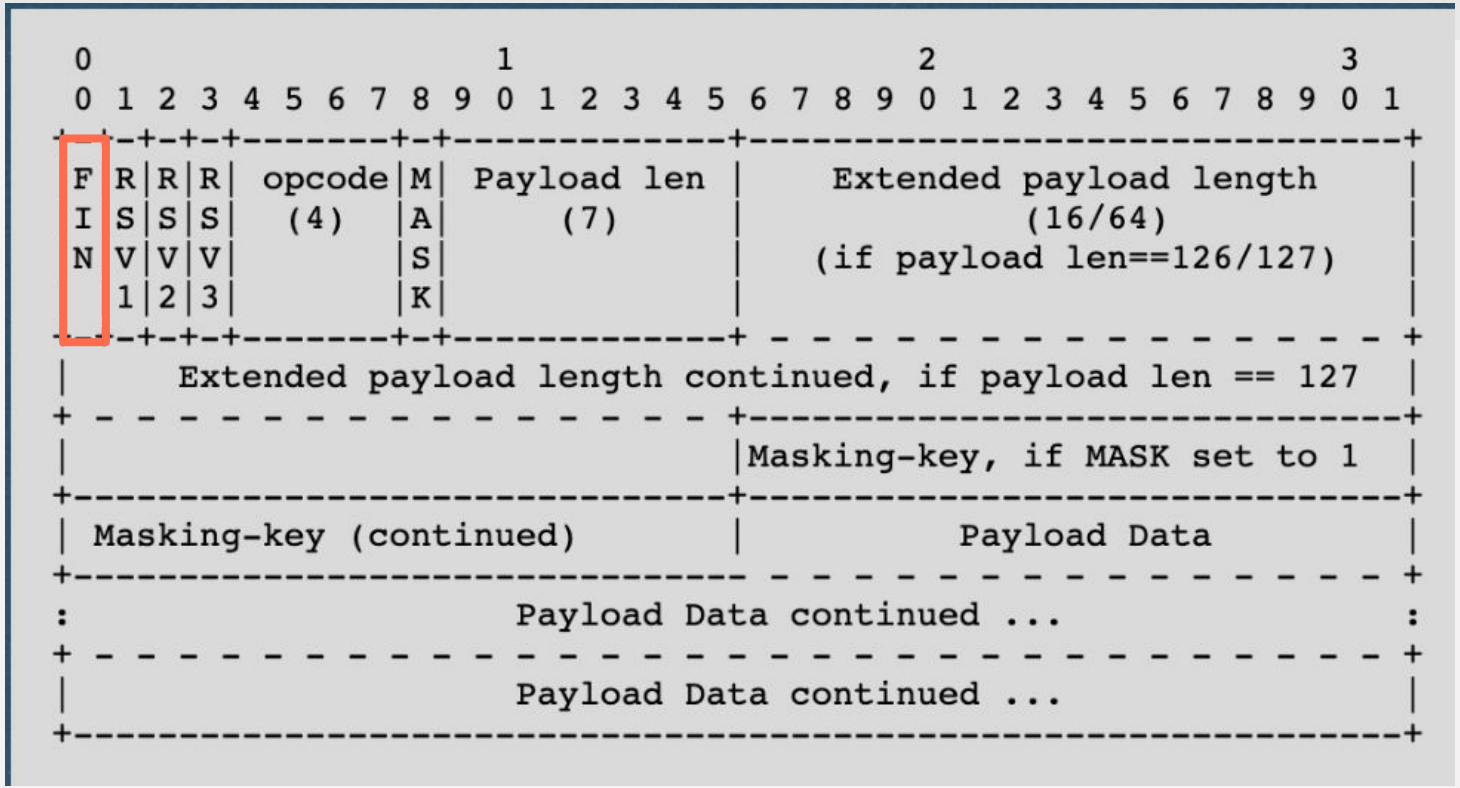
The diagram from the [websockets rfc] numbers represent each **bit** not byte (Each row **32 bits** or **4 bytes**)

Websockets



Each row represents another set of 4 bytes

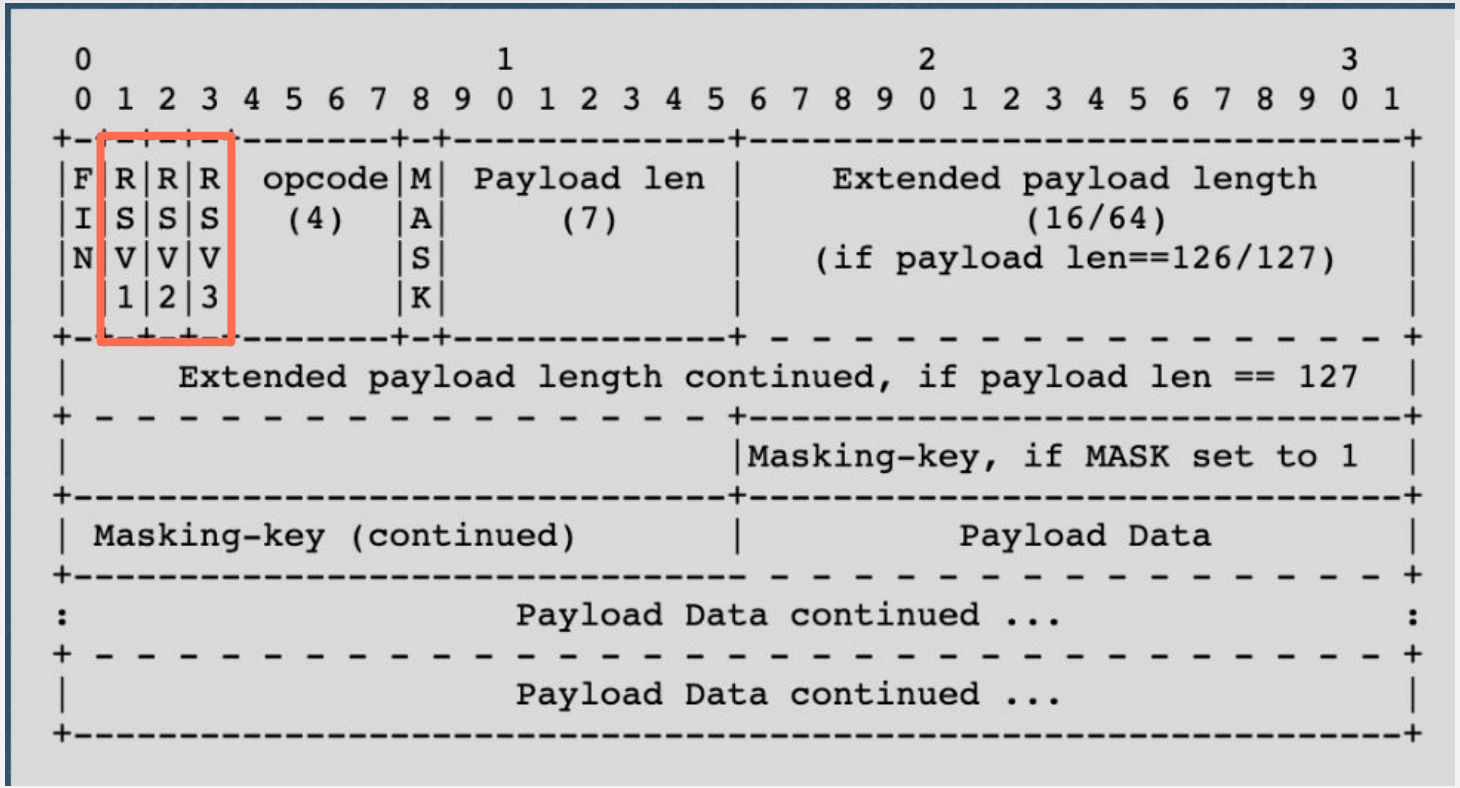
The diagram from the [websockets rfc] numbers represent each **bit** not byte (Each row **32 bits** or **4 bytes**)



FIN: The finish bit

1 - This is the last frame for this message

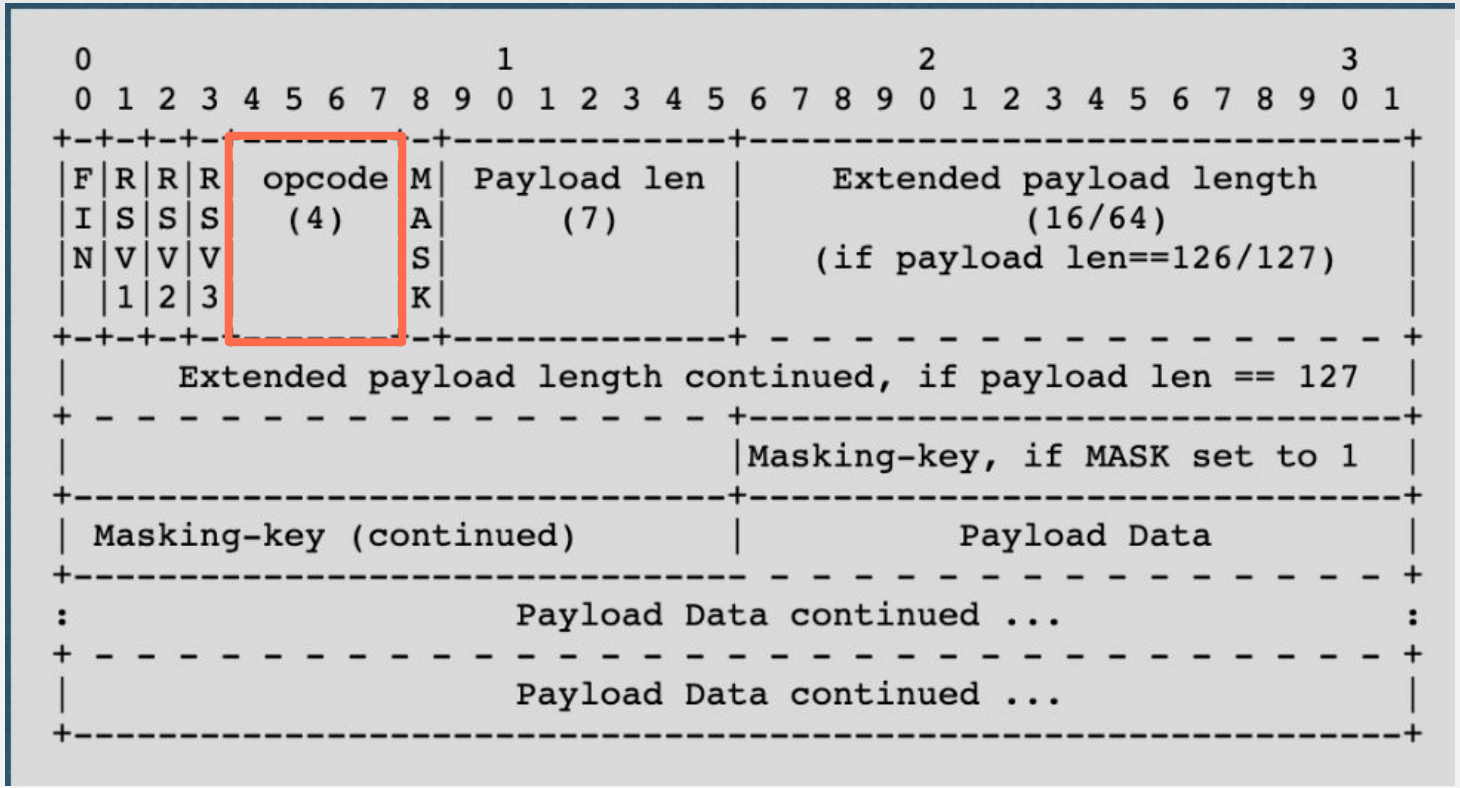
0 - There will be continuation frames containing more data for the same message



RSV: Reserved bits

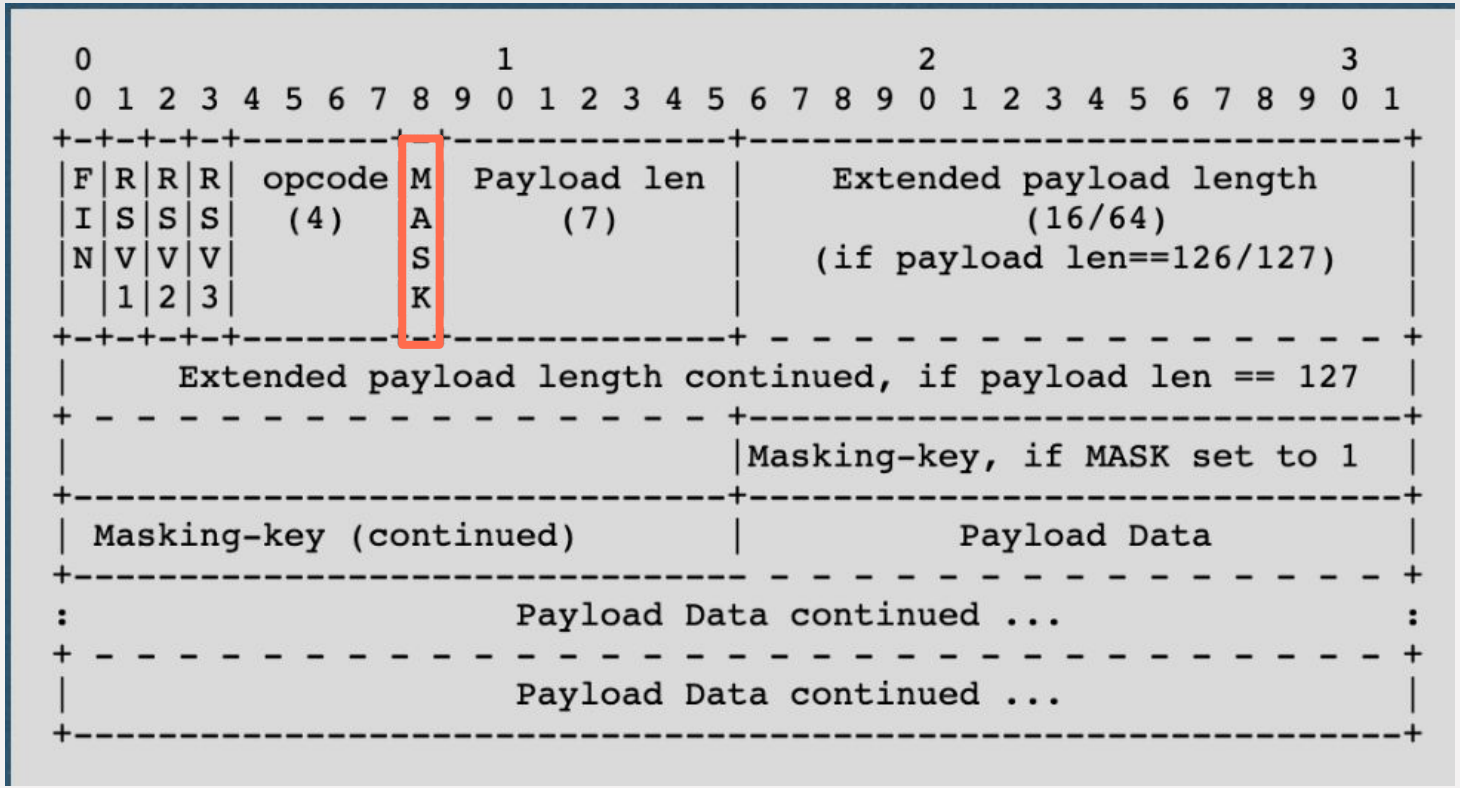
Used to specify any extensions being used

[You can assume these are always 000 for the HW]



Opcode: Operation code

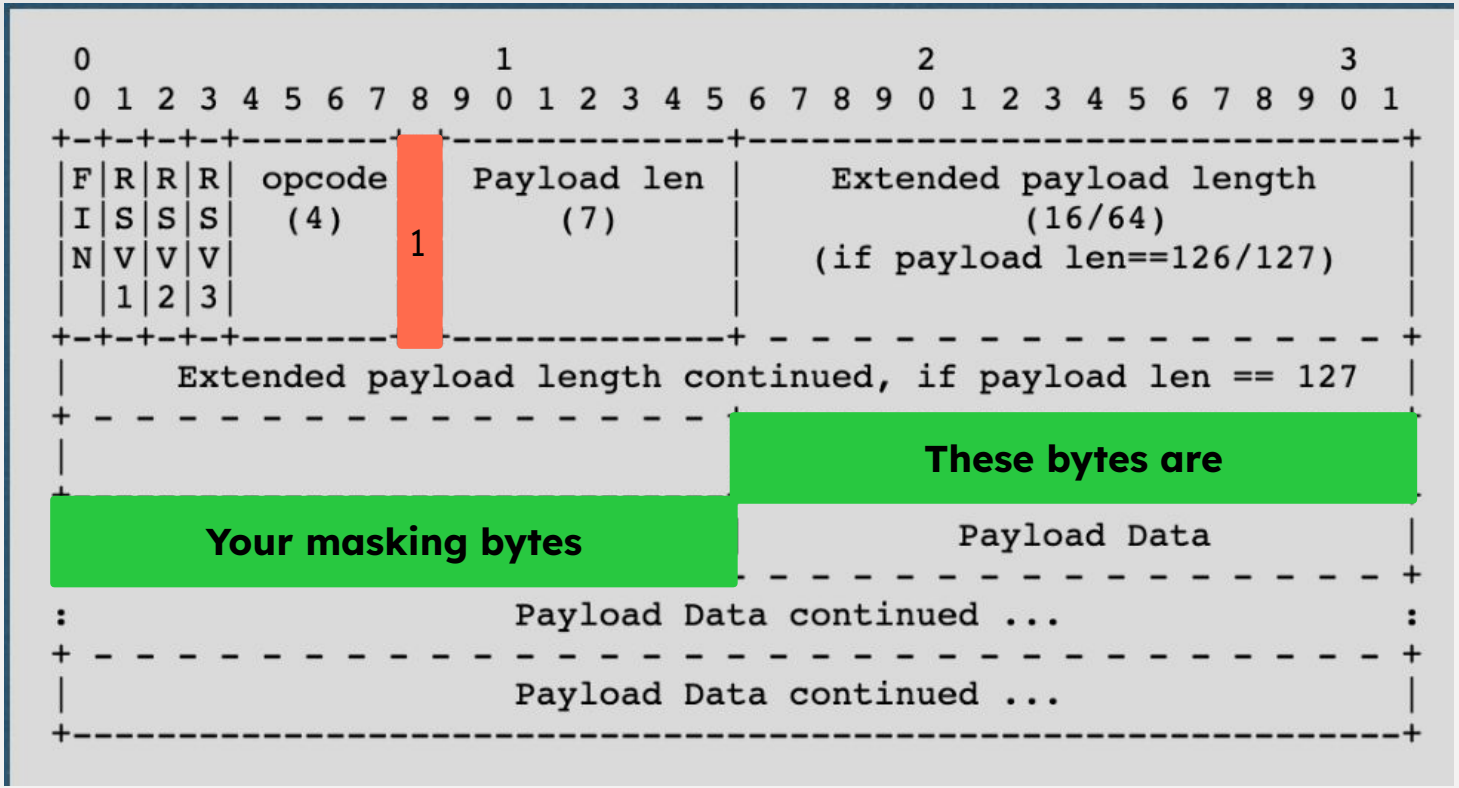
Specifies the type of information contained in the payload
Ex: 0001 for text, 0010 for binary, 1000 to close the connection, 0000 for continuation frame



MASK: Mask bit

Set to 1 if a mask is being used

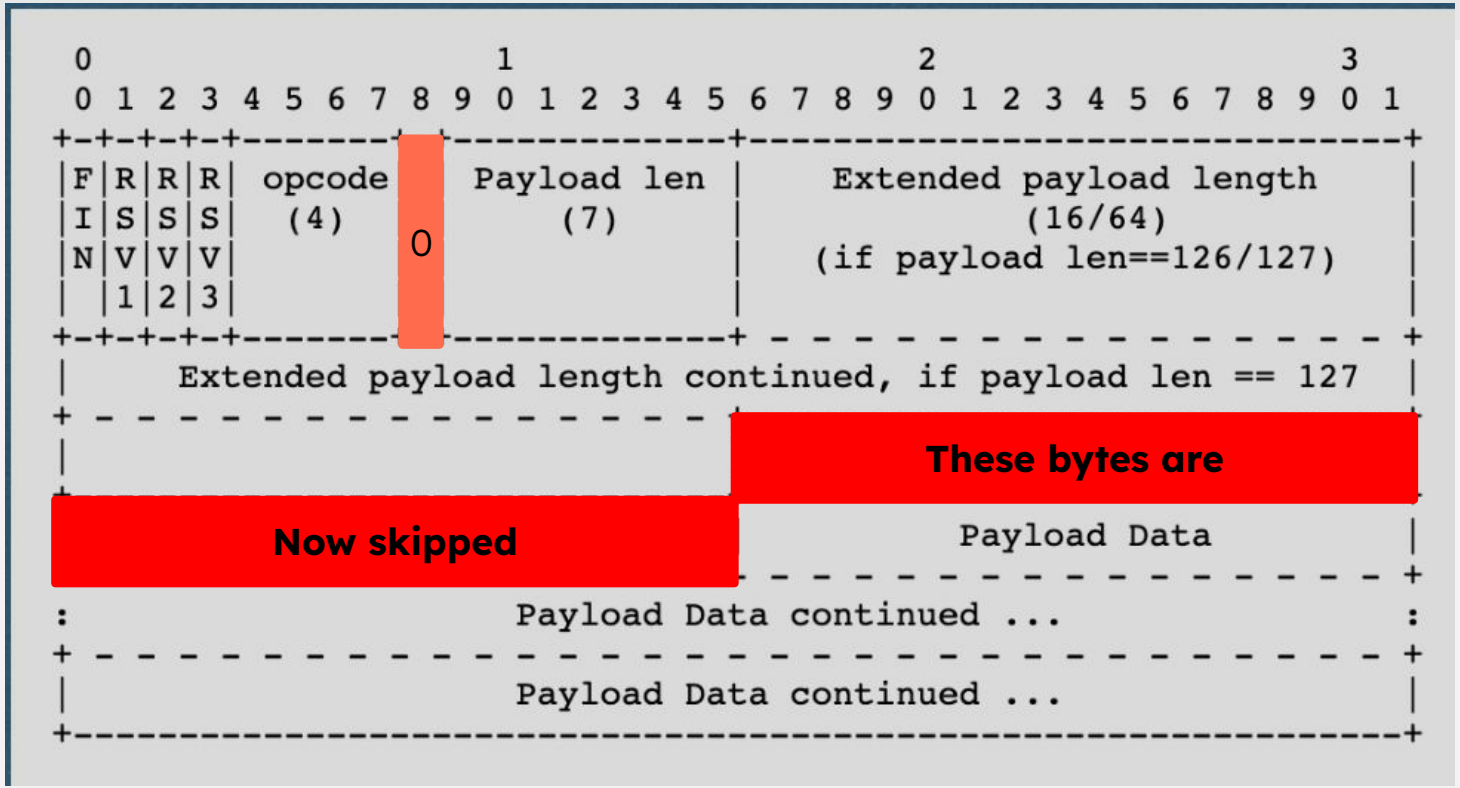
- Set to 0 if no mask is being used
- This will be 1 when receiving messages from a client



MASK: Mask bit

Set to 1 if a mask is being used

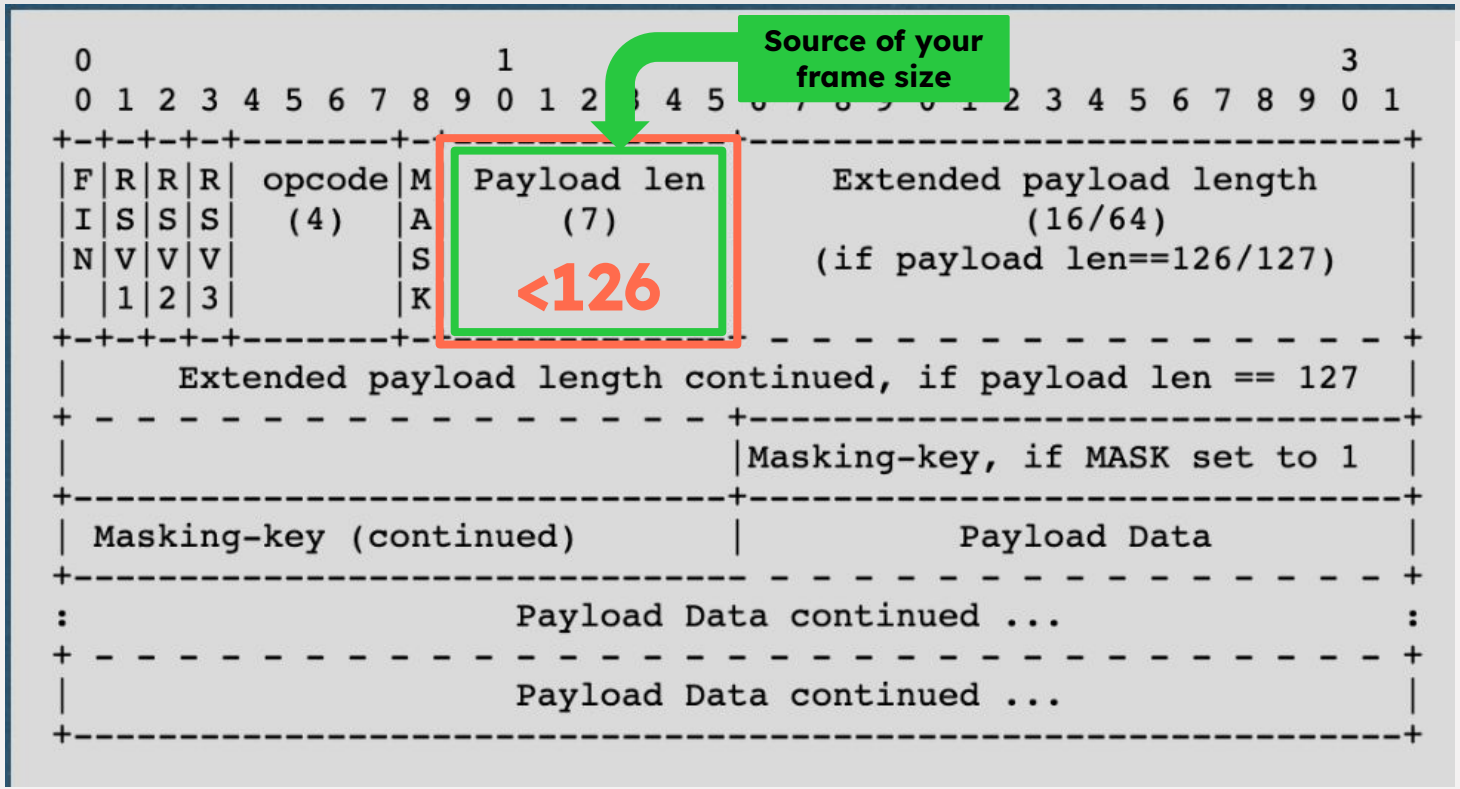
- Set to 0 if no mask is being used
- This will be 1 when receiving messages from a client



MASK: Mask bit

Set to 1 if a mask is being used

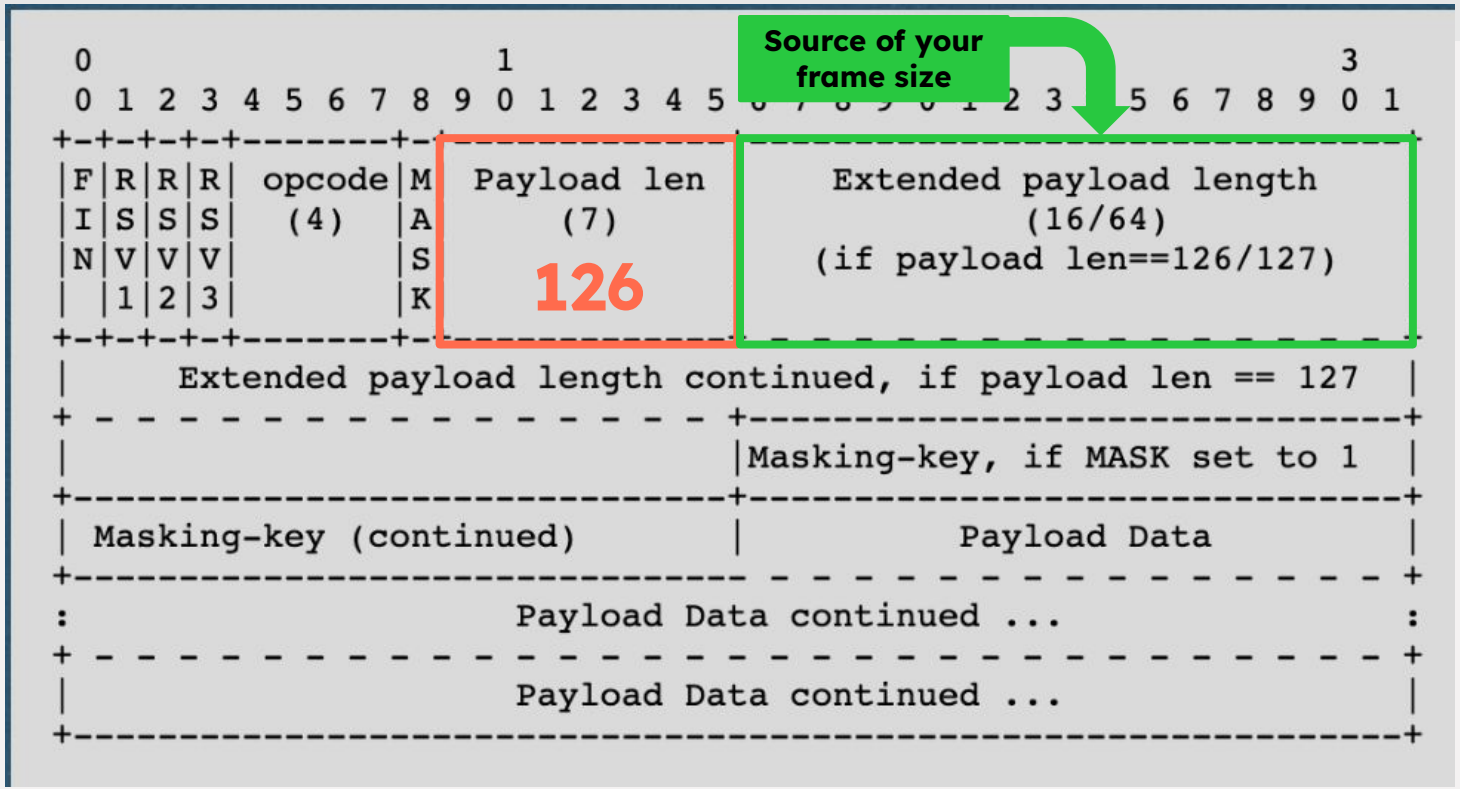
- Set to 0 if no mask is being used
- This will be 1 when receiving messages from a client



Payload (Frame) Length

If the length is <126 bytes

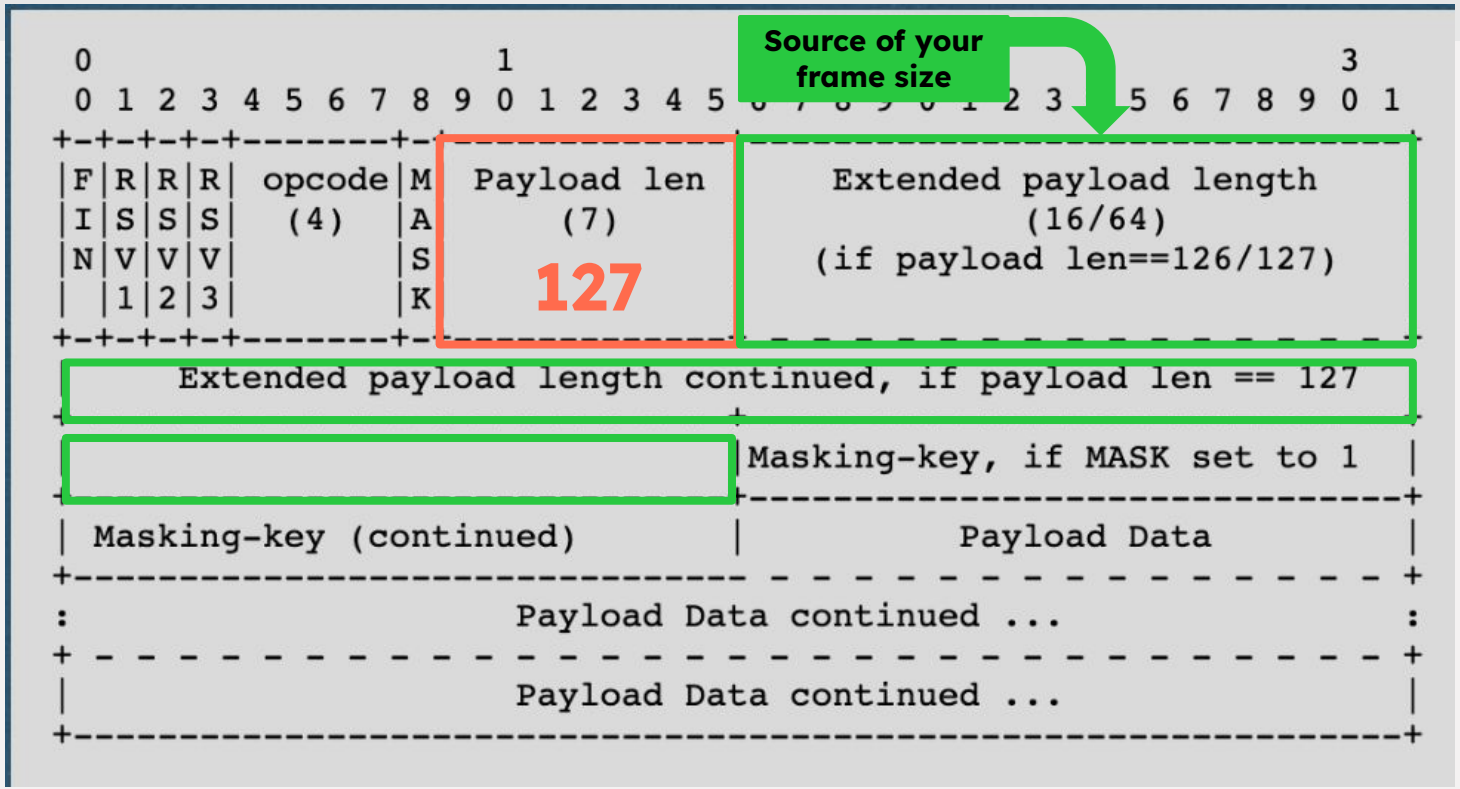
- The length is represented in 7 bits, sharing a byte with the MASK bit
- The next bit after the length is either the mask or payload



Payload (Frame) Length

If the length is >=126 and <65536 bytes

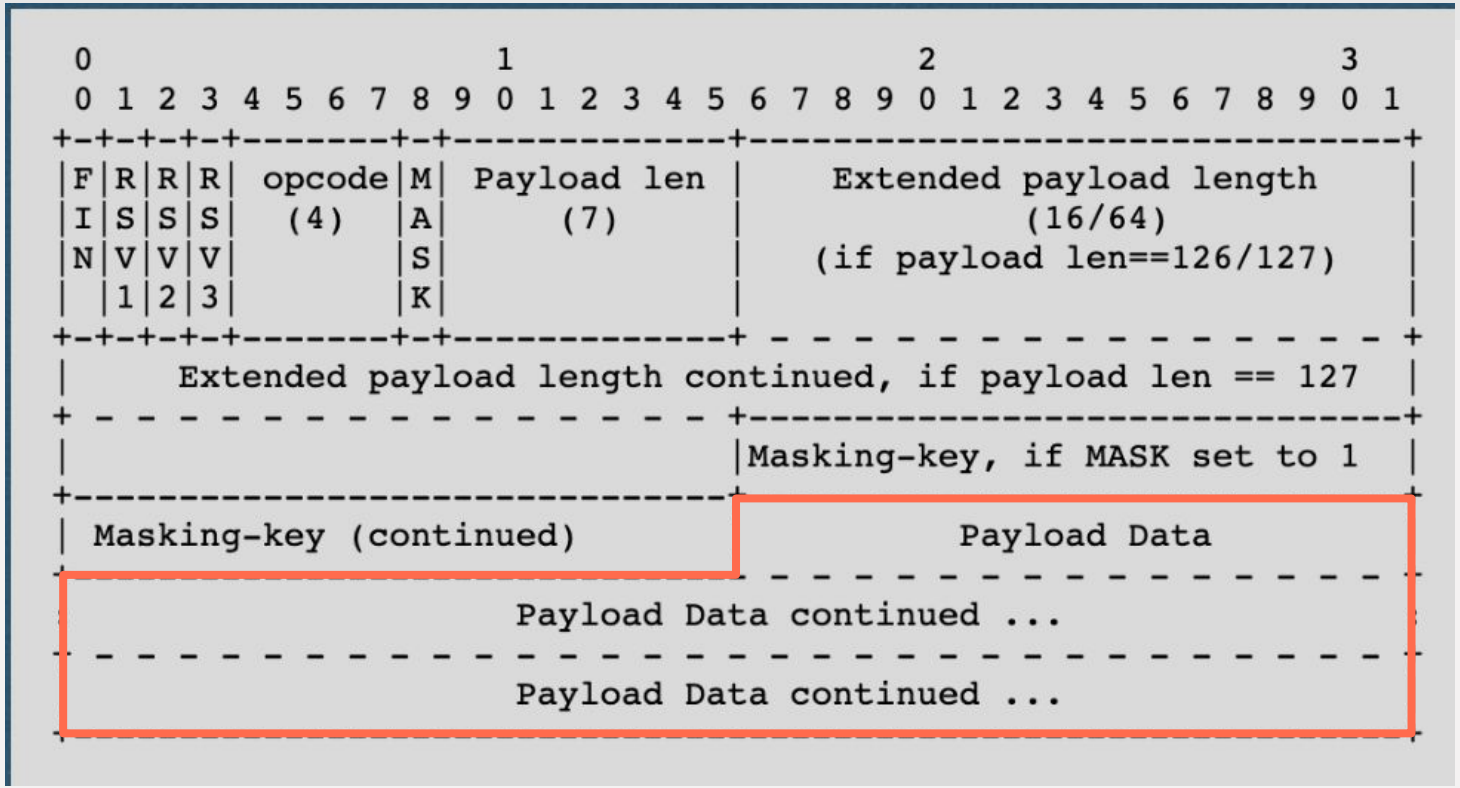
- The 7 bit length will be exactly 126 (1111110)
- The next 16 bits (2 byte) represents the payload length



Payload (Frame) Length

If the length is >=65536 bytes

- The 7 bit length will be exactly 127 (1111111)
- The next 64 bits (8 bytes) represents the payload length



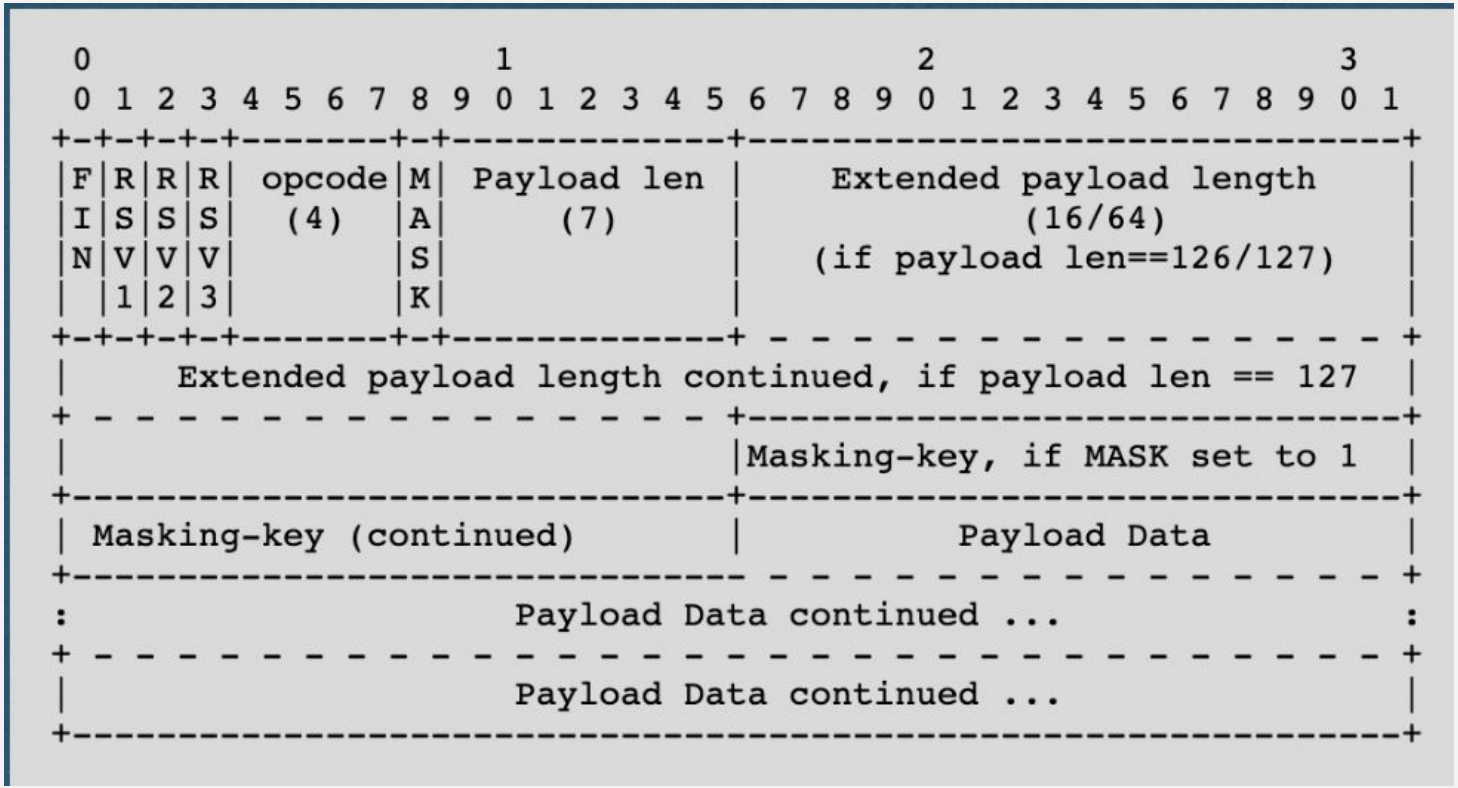
The Payload

The data you are trying to send in bytes

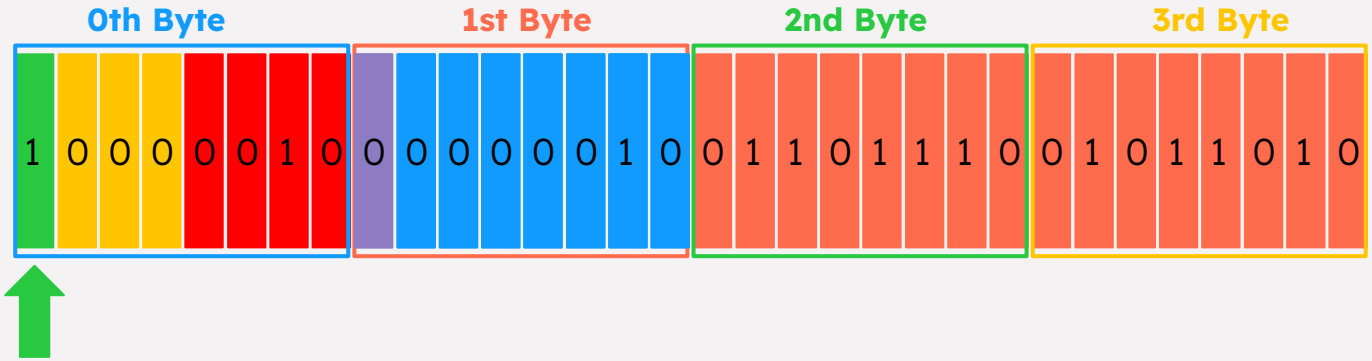
- If there is a mask, every 4 bytes will be XOR'd with the 4 bytes of the mask to get true payload

Examples

Websockets


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
FIN
RSV1
RSV2
RSV3
opcode (4)
MASK
Payload len (7)
Extended payload length (16/64)
(if payload len==126/127)
Extended payload length continued, if payload len == 127
Masking-key, if MASK set to 1
Masking-key (continued)
Payload Data
Payload Data continued ...
Payload Data continued ...

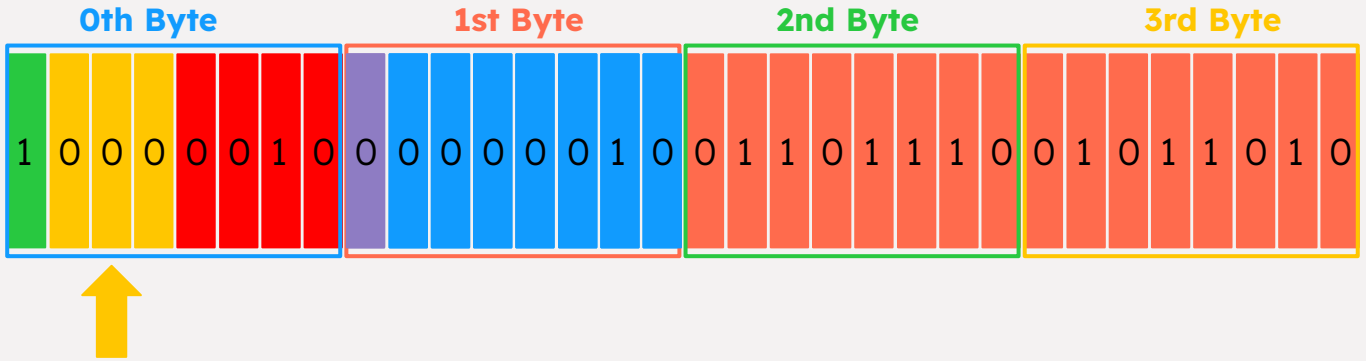
Websockets
Example #1



FIN: The finish bit

- 1 - This is the last frame for this message

Websockets
Example #1

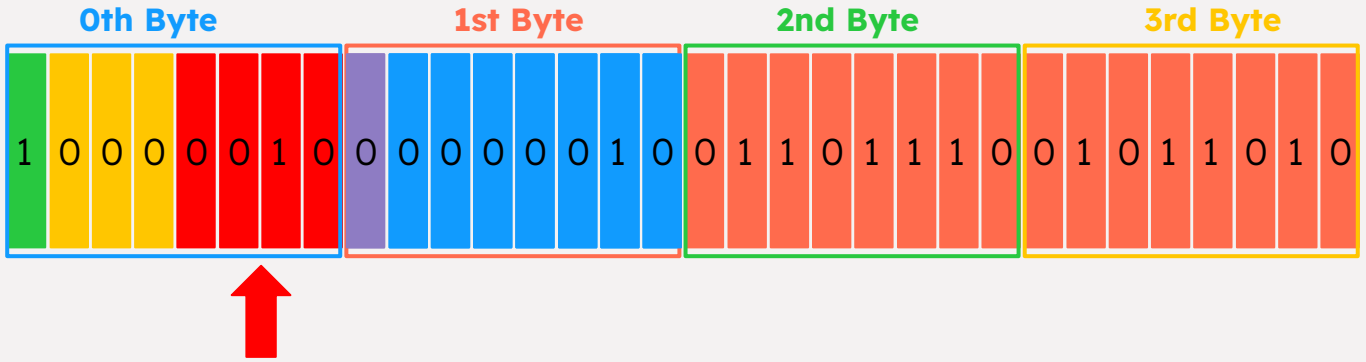


RSV: Reserved bits

- [You can assume these are always 000 for the HW]

Websockets

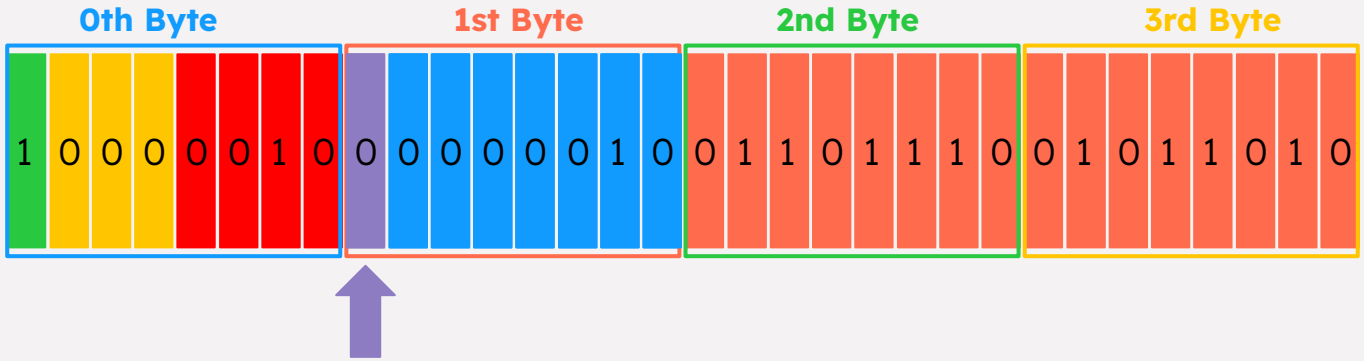
Example #1



opcode: Operation code

- 0010 for binary

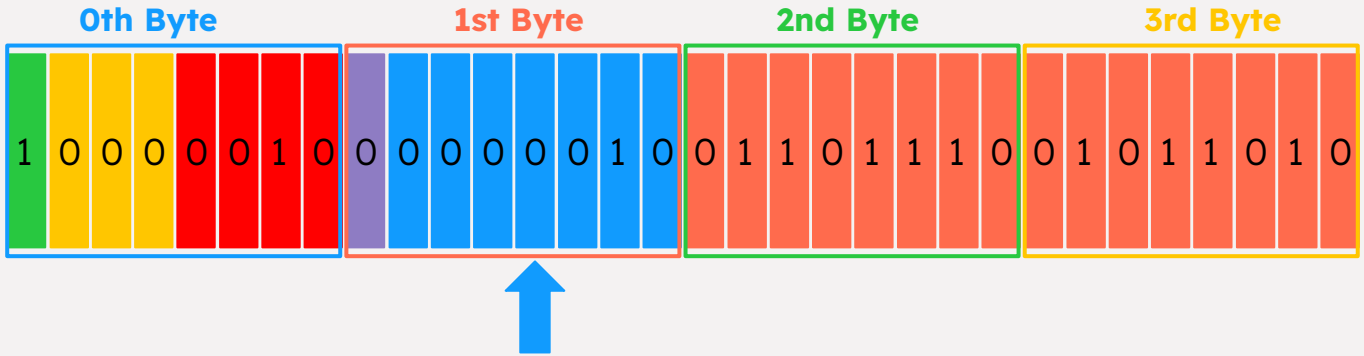
Websockets
Example #1



MASK: Mask bit

- Set to 0 no mask is being used

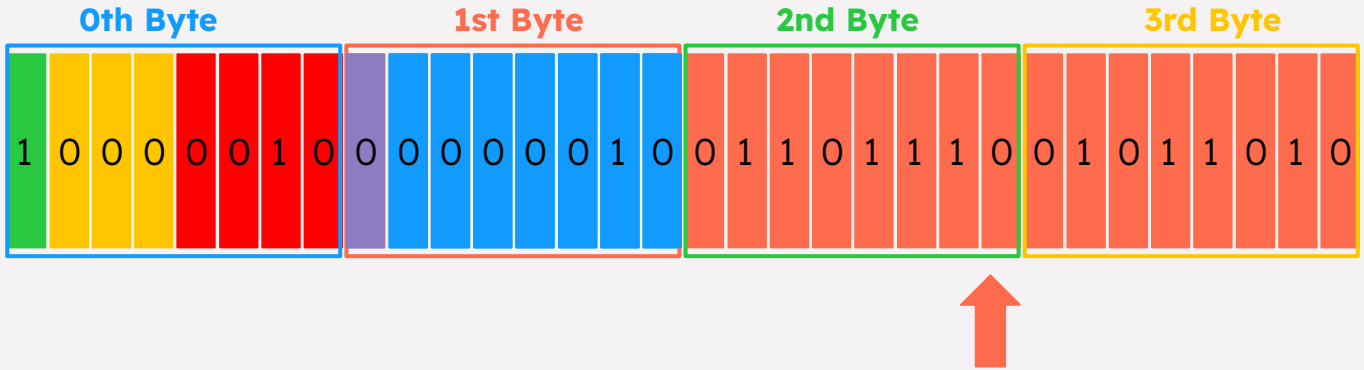
Websockets
Example #1



Payload (Frame) Length

- Length of 2 (0010000)

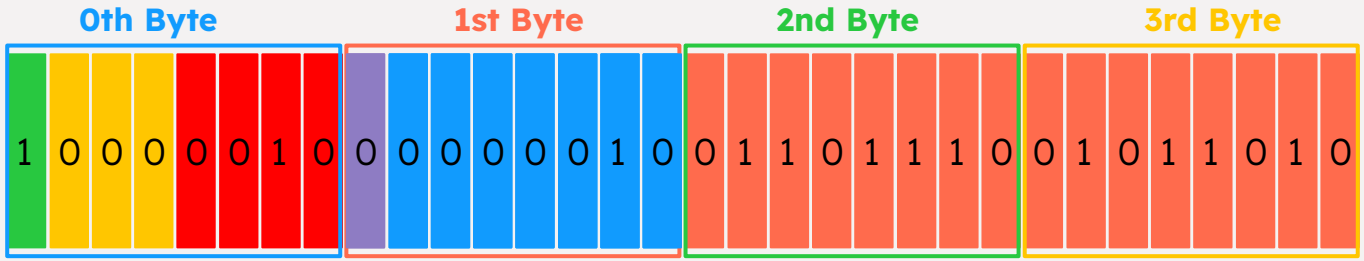
Websockets
Example #1



The Payload

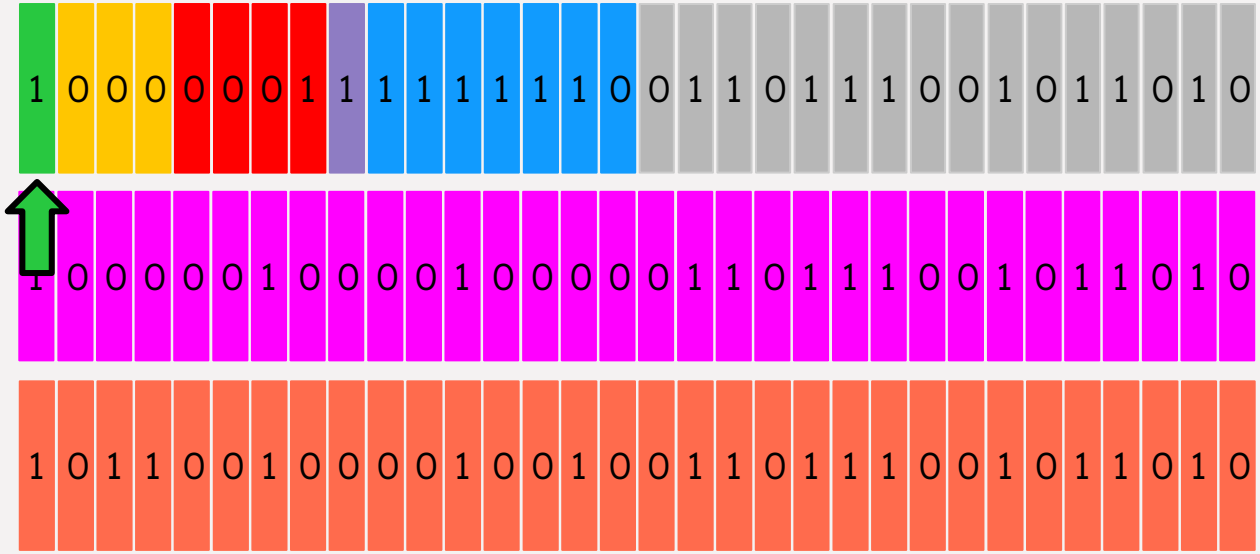
- Its two bytes of binary data

Websockets
Example #1



FIN: The finish bit - 1
RSV: Reserved bits - 000
opcode: Operation code - 0010 for binary
MASK: Mask bit - Set to 0 no mask is being used
Payload (Frame) Length - Length of 2 bytes (0000010)
The Payload - Its two bytes of binary data

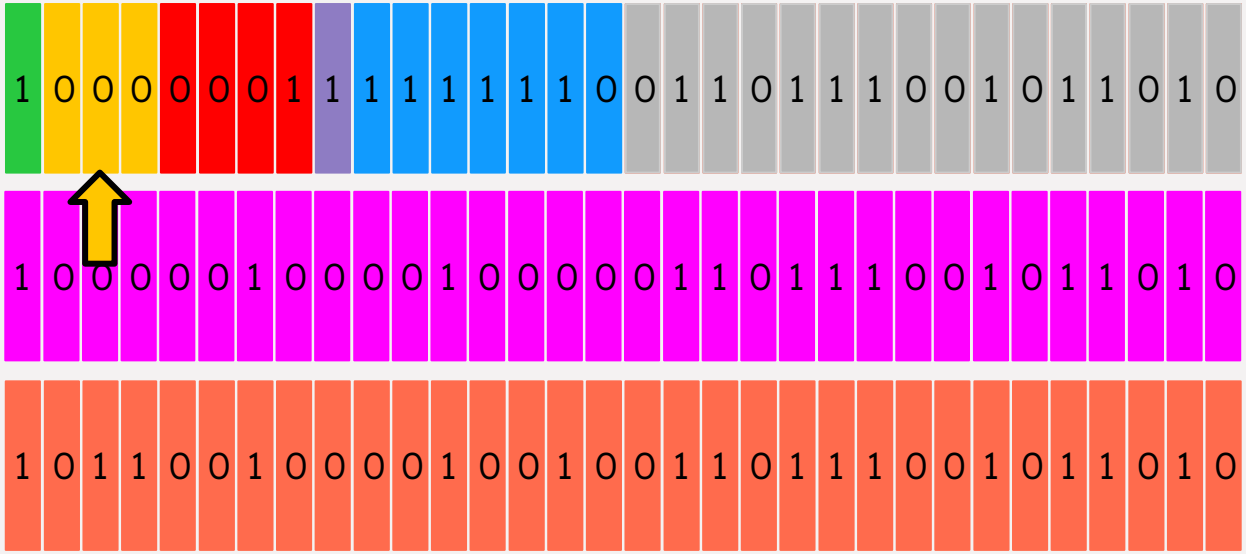
Websockets
Example #2



FIN: The finish bit

- 1 - This is the last frame for this message

Websockets
Example #2

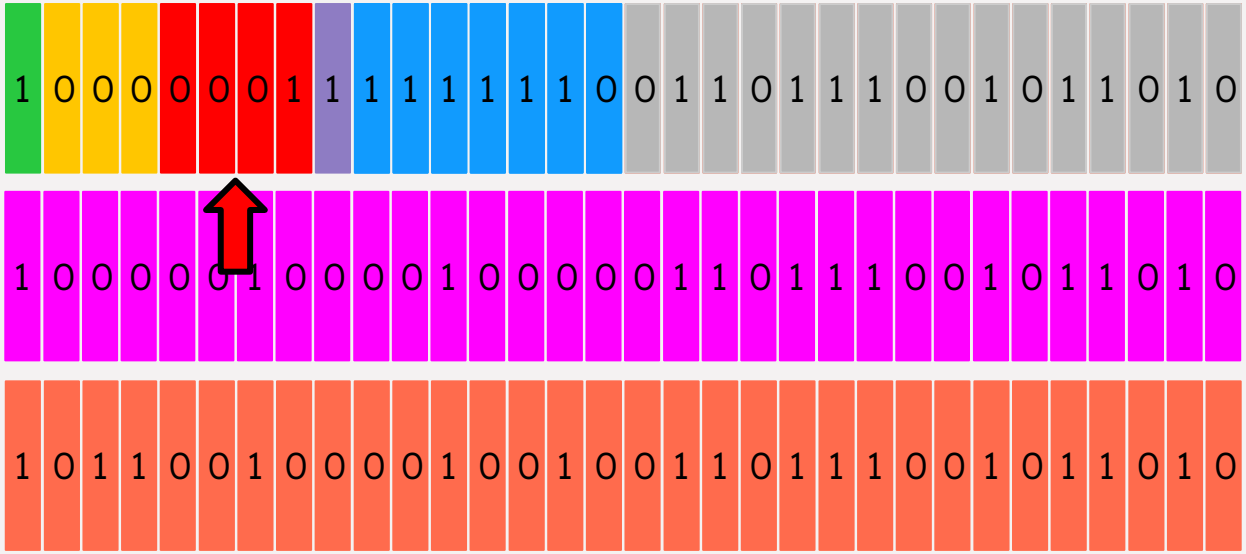


RSV: Reserved bits

- [You can assume these are always 000 for the HW]

Websockets

Example #2

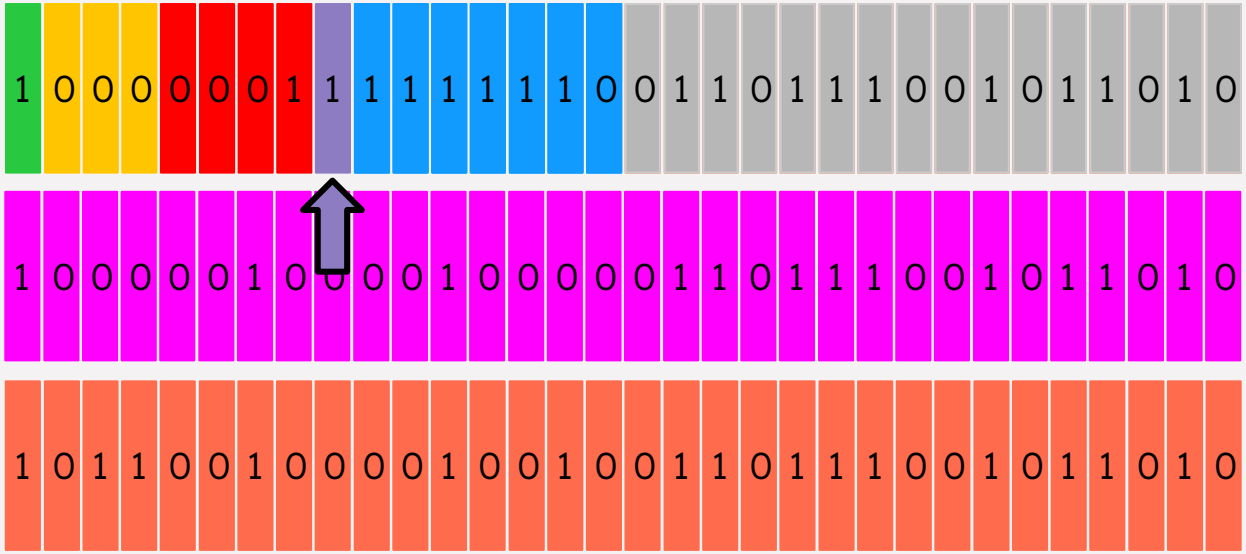


opcode: Operation code

- 0001 for text

Websockets

Example #2

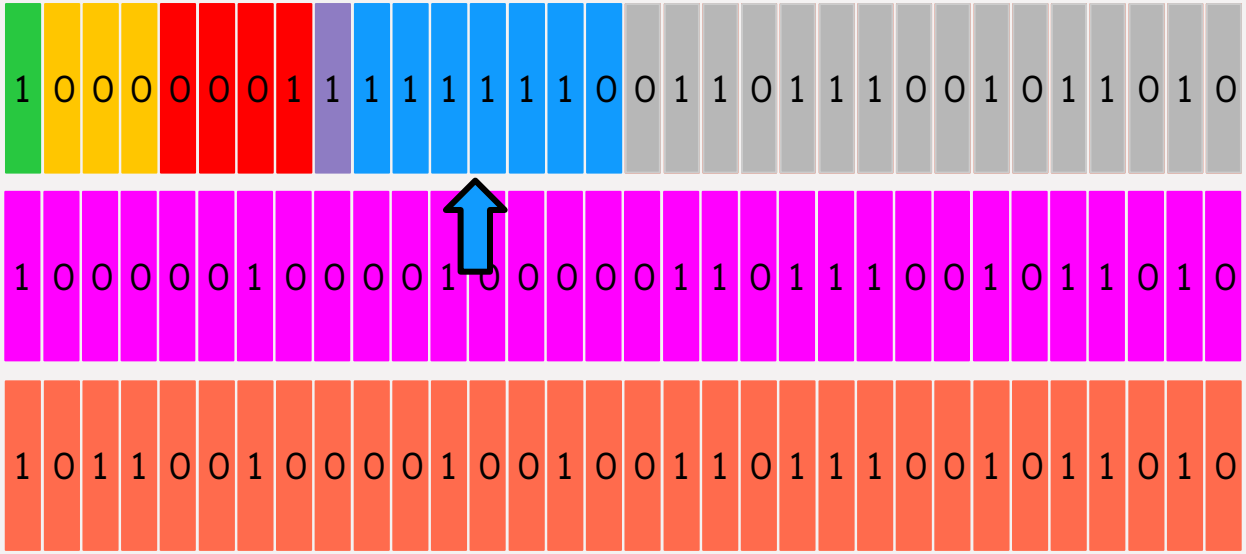


MASK: Mask bit

- Set to 1 mask is being used

Websockets

Example #2

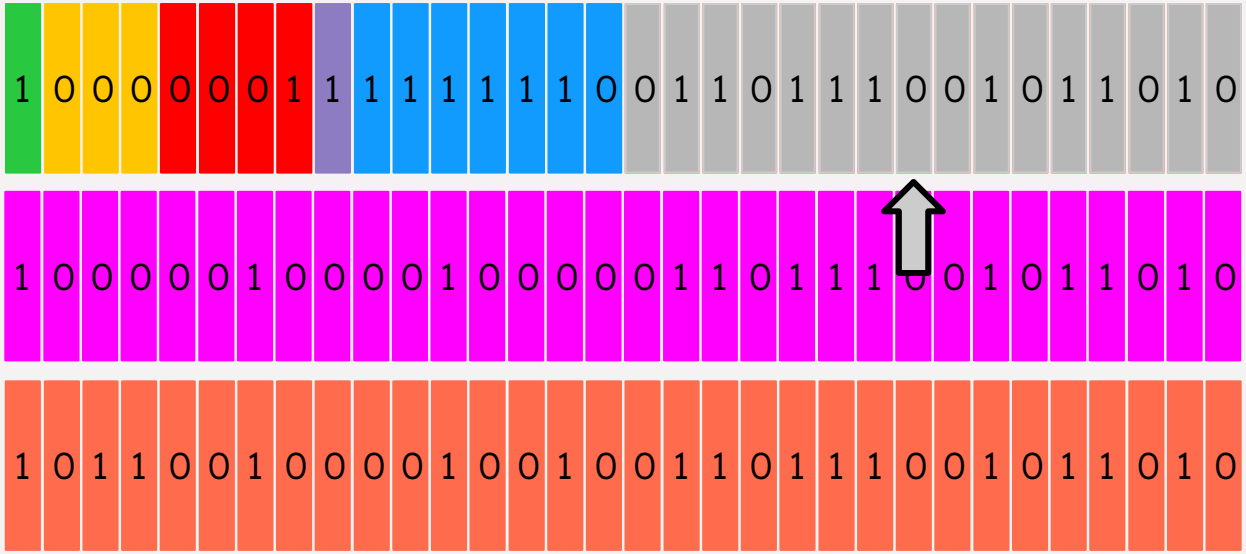


Payload (Frame) Length

- The 7 bit length is exactly 126 (1111110)
- length is >=126 and <65536 bytes

Websockets

Example #2

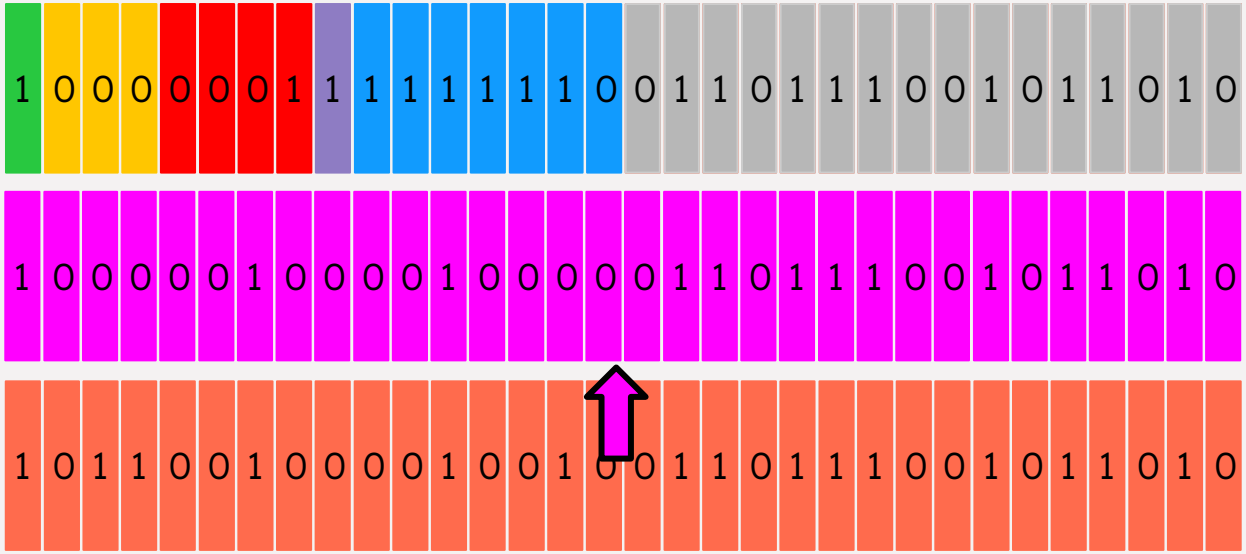


Extended Payload (Frame) Length

- Length is, 28250 bytes

Websockets

Example #2

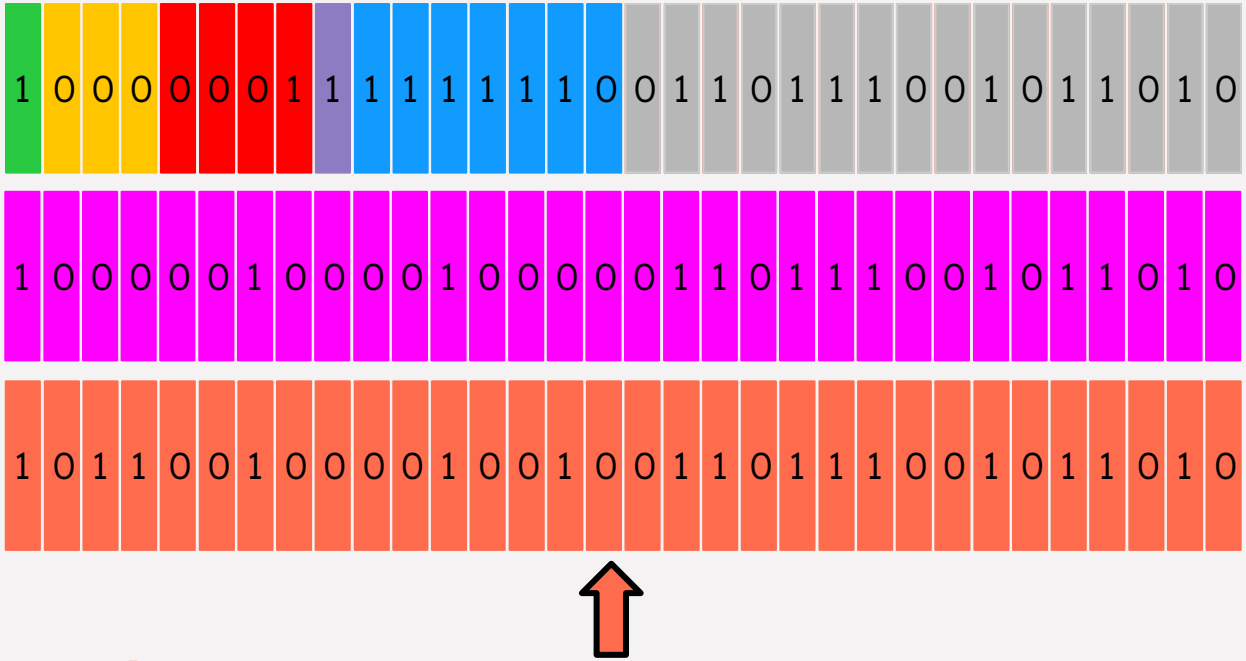


MASK

- This is the mask that will XOR with payload

Websockets

Example #2

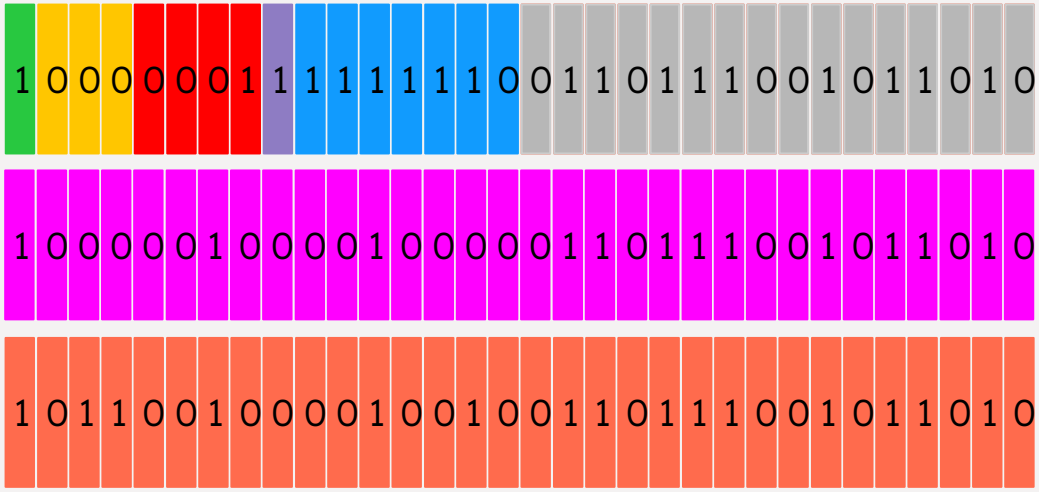


The Payload

- Only first 4 bytes of 28250, would continue on

Websockets

Example #2



FIN: The finish bit - 1
RSV: Reserved bits - 000
opcode: Operation code - 0010 for text
MASK: Mask bit - Set to 1 mask is being used
Payload Length - 126
Extended Payload (Frame) Length - Length of 28250 bytes
MASK - will XOR with payload
The Payload - 28250 bytes of data