Performance Evaluation of WebSocket Protocol for Implementation of Full-Duplex Web Streams

Oleg Bilovus

Università degli Studi di Salerno

1st Scalability Research Forum

Outline

Background

- *Historically*, creating web applications that need bidirectional communication between a client and a server has required an abuse of HTTP to poll the server for updates while sending upstream notifications as distinct HTTP calls.

Background

- *Historically*, creating web applications that need bidirectional communication between a client and a server has required an abuse of HTTP to poll the server for updates while sending upstream notifications as distinct HTTP calls.

Background

- *Historically*, creating web applications that need bidirectional communication between a client and a server has required an abuse of HTTP to poll the server for updates while sending upstream notifications as distinct HTTP calls.

Background

- *Historically*, creating web applications that need bidirectional communication between a client and a server has required an abuse of HTTP to poll the server for updates while sending upstream notifications as distinct HTTP calls.

Background

- *Historically*, creating web applications that need bidirectional communication between a client and a server has required an abuse of HTTP to poll the server for updates while sending upstream notifications as distinct HTTP calls.

Background

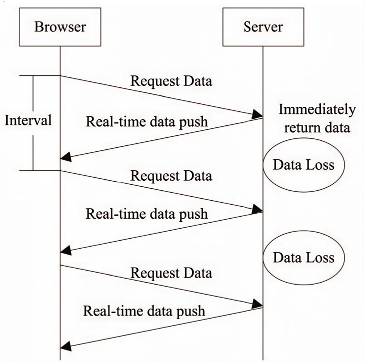
- *Historically*, creating web applications that need bidirectional communication between a client and a server has required an abuse of HTTP to poll the server for updates while sending upstream notifications as distinct HTTP calls.

Background

- *Historically*, creating web applications that need bidirectional communication between a client and a server has required an abuse of HTTP to poll the server for updates while sending upstream notifications as distinct HTTP calls.

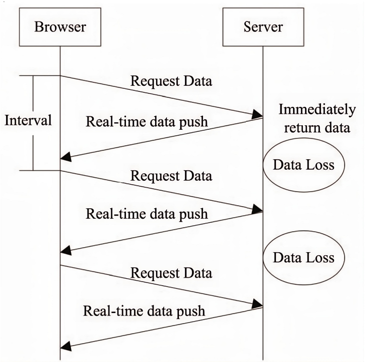
HTTP polling

Check whether the server is changed in a while, thereby performing incremental updates.



HTTP polling

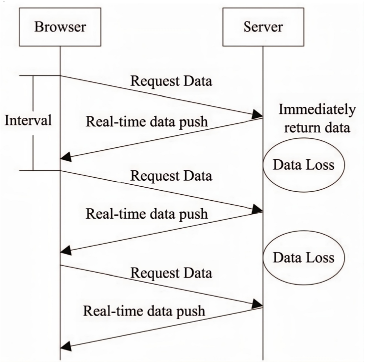
Check whether the server is changed in a while, thereby performing incremental updates.



- How often to query?

HTTP polling

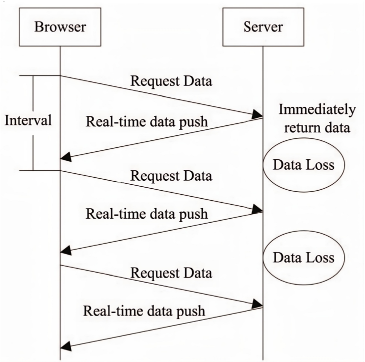
Check whether the server is changed in a while, thereby performing incremental updates.



- How often to query?
- Continuously short interval requests will be washed away the server.

HTTP polling

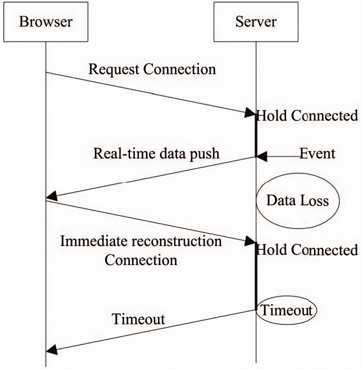
Check whether the server is changed in a while, thereby performing incremental updates.



- How often to query?
- Continuously short interval requests will be washed away the server.
- Long interval will require more time to reach the client, no real-time data.

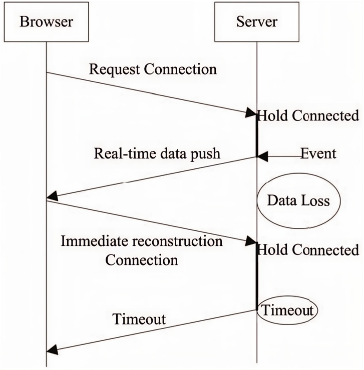
HTTP long polling

When a client sends a data request, the server will block the request until there is data transfer or timeout before returning.



HTTP long polling

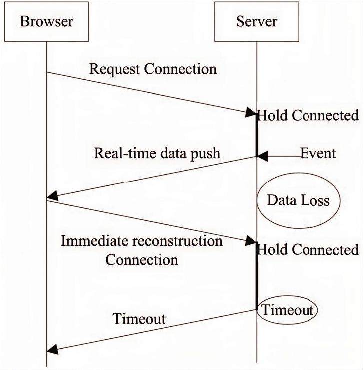
When a client sends a data request, the server will block the request until there is data transfer or timeout before returning.



- Solve the short polling frequency to access the server.

HTTP long polling

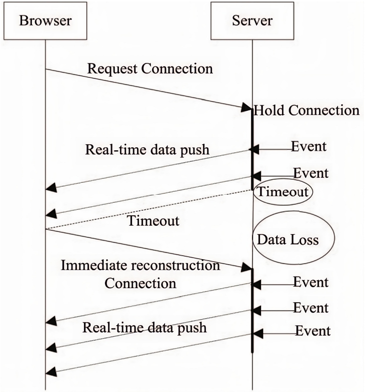
When a client sends a data request, the server will block the request until there is data transfer or timeout before returning.



- Solve the short polling frequency to access the server.
- No bidirectional communication, server push data.

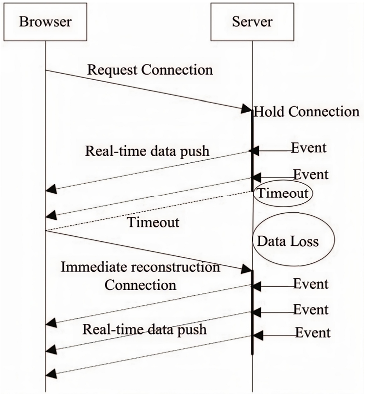
Streaming

Iframe embed a hidden frame in an HTML page, then set it as a long connection request, thus the server can send data to the clients constantly.



Streaming

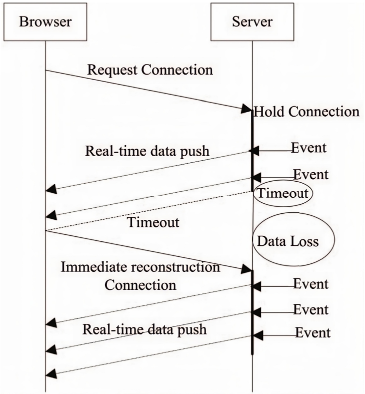
Iframe embed a hidden frame in an HTML page, then set it as a long connection request, thus the server can send data to the clients constantly.



- It can send multiple events from a single request.

Streaming

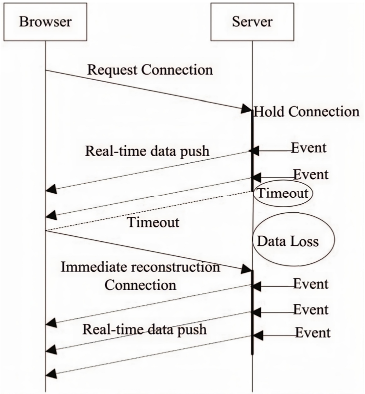
Iframe embed a hidden frame in an HTML page, then set it as a long connection request, thus the server can send data to the clients constantly.



- It can send multiple events from a single request.
- But, it increases the burden on the server, causing the server performance degradation, or even collapse.

Streaming

Iframe embed a hidden frame in an HTML page, then set it as a long connection request, thus the server can send data to the clients constantly.



- It can send multiple events from a single request.
- But, it increases the burden on the server, causing the server performance degradation, or even collapse.
- No bidirectional communication.

Outline

RFC 6455
Keywords

- The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code.

RFC 6455

Keywords

- The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code.

RFC 6455

Keywords

- The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code.

RFC 6455

Keywords

- The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code.

RFC 6455

Keywords

- The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code.

RFC 6455

Keywords

- The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code.
- The protocol consists of an opening handshake followed by basic message framing, layered over TCP.

RFC 6455

Keywords

- The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code.
- The protocol consists of an opening handshake followed by basic message framing, layered over TCP.

RFC 6455

Keywords

- The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code.
- The protocol consists of an opening handshake followed by basic message framing, layered over TCP.

RFC 6455

Keywords

- The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code.
- The protocol consists of an opening handshake followed by basic message framing, layered over TCP.

RFC 6455

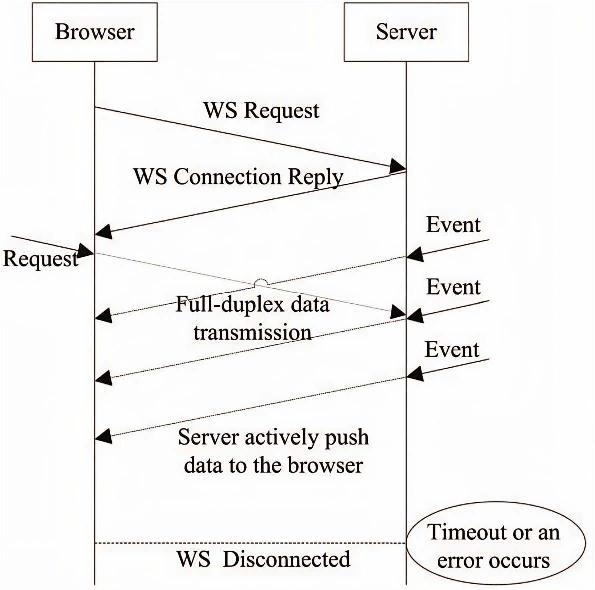
Keywords

- The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code.
- The protocol consists of an opening handshake followed by basic message framing, layered over TCP.
- The goal of this technology is to provide a mechanism for browser-based applications that need two-way communication with servers.

RFC 6455
Keywords

- The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code.
- The protocol consists of an opening handshake followed by basic message framing, layered over TCP.
- The goal of this technology is to provide a mechanism for browser-based applications that need two-way communication with servers.

WebSocket


Browser
Server
WS Request
WS Connection Reply
Event
Request
Event
Full-duplex data
transmission
Event
Server actively push
data to the browser
Timeout or an
error occurs
WS Disconnected

Handshake

- For WebSocket-based communication, a WebSocket session should be established first.

Handshake

- For WebSocket-based communication, a WebSocket session should be established first.
- To establish a session, client sends a WebSocket Upgrade Request to the server, upon which server responds with a WebSocket Upgrade Response.

Handshake

- For WebSocket-based communication, a WebSocket session should be established first.
- To establish a session, client sends a WebSocket Upgrade Request to the server, upon which server responds with a WebSocket Upgrade Response.
- From this point forward, the client and server can send data back and forth in asynchronous full-duplex mode.

WebSocket Upgrade Request

```
GET /chat HTTP/1.1
Host:  server.example.com
Upgrade:  WebSocket
Connection:  Upgrade
Sec-WebSocket-Key:
dGhlIHNhbXBsZSBub25jZQ==
Origin:  http://example.com
Sec-WebSocket-Protocol:
chat, superchat
Sec-WebSocket-Version:  13
```

WebSocket Upgrade Request

- HTTP GET request.

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: WebSocket
Connection: Upgrade
Sec-WebSocket-Key:
dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol:
chat, superchat
Sec-WebSocket-Version: 13
```

WebSocket Upgrade Request

```
GET /chat HTTP/1.1
Host:  server.example.com
Upgrade:  WebSocket
Connection:  Upgrade
Sec-WebSocket-Key:
dGhlIHNhbXBsZSBub25jZQ==
Origin:  http://example.com
Sec-WebSocket-Protocol:
chat, superchat
Sec-WebSocket-Version:  13
```

- HTTP GET request.
- URI to identify endpoint.

WebSocket Upgrade Request

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: WebSocket
Connection: Upgrade
Sec-WebSocket-Key:
dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol:
chat, superchat
Sec-WebSocket-Version: 13
```

- HTTP GET request.
- URI to identify endpoint.
- Headers indicating the will to switch from regular HTTP to WebSocket.

WebSocket Upgrade Request

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: WebSocket
Connection: Upgrade
Sec-WebSocket-Key:
dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol:
chat, superchat
Sec-WebSocket-Version: 13
```

- HTTP GET request.
- URI to identify endpoint.
- Headers indicating the will to switch from regular HTTP to WebSocket.
- A key the server has to use to prove that it can use WebSockets.

WebSocket Upgrade Request

```
GET /chat HTTP/1.1
Host:  server.example.com
Upgrade:  WebSocket
Connection:  Upgrade
Sec-WebSocket-Key:
dGhlIHNhbXBsZSBub25jZQ==
Origin:  http://example.com
Sec-WebSocket-Protocol:
chat, superchat
Sec-WebSocket-Version:  13
```

- HTTP GET request.
- URI to identify endpoint.
- Headers indicating the will to switch from regular HTTP to WebSocket.
- A key the server has to use to prove that it can use WebSockets.
- WebSocket protocols.

WebSocket Upgrade Request

```
GET /chat HTTP/1.1
Host:  server.example.com
Upgrade:  WebSocket
Connection:  Upgrade
Sec-WebSocket-Key:
dGhlIHNhbXBsZSBub25jZQ==
Origin:  http://example.com
Sec-WebSocket-Protocol:
chat, superchat
Sec-WebSocket-Version:  13
```

- HTTP GET request.
- URI to identify endpoint.
- Headers indicating the will to switch from regular HTTP to WebSocket.
- A key the server has to use to prove that it can use WebSockets.
- WebSocket protocols.
- WebSocket version.

WebSocket Upgrade Response

```
HTTP/1.1 101 Switching
protocols
Upgrade:  WebSocket
Connection:  Upgrade
Sec-WebSocket-Accept:
dGhlIHNhbXBsZSBub25jZQ==
Origin:  http://example.com
Sec-WebSocket-Protocol:  chat
```

WebSocket Upgrade Response

```
HTTP/1.1 101 Switching
protocols
Upgrade: WebSocket
Connection: Upgrade
Sec-WebSocket-Accept:
dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat
```

- Server confirms it supports WebSocket.

WebSocket Upgrade Response

```
HTTP/1.1 101 Switching
protocols
Upgrade: WebSocket
Connection: Upgrade
Sec-WebSocket-Accept:
dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat
```

- Server confirms it supports WebSocket.
- Server proves that it can use WebSocket. Client checks it.

WebSocket Upgrade Response

```
HTTP/1.1 101 Switching
protocols
Upgrade:  WebSocket
Connection:  Upgrade
Sec-WebSocket-Accept:
dGhlIHNhbXBsZSBub25jZQ==
Origin:  http://example.com
Sec-WebSocket-Protocol:  chat
```

- Server confirms it supports WebSocket.
- Server proves that it can use WebSocket. Client checks it.
- Server tells which protocol it supports.

WebSocket Frame

- After the handshake is successful, client and server can communicate in full-duplex by using frames.

WebSocket Frame

- After the handshake is successful, client and server can communicate in full-duplex by using frames.
- The added overhead to the payload data is minimal because it does not send all the HTTP headers for each frame.

WebSocket Frame

- After the handshake is successful, client and server can communicate in full-duplex by using frames.
- The added overhead to the payload data is minimal because it does not send all the HTTP headers for each frame.
- Each frame adds at least 2 bytes of overhead to the payload data. Depending on the length of the payload data and the direction of the communication, the length of the overhead may increase up to 14 bytes.

WebSocket Frame Structure

```
 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
+-+-+-+-+-------+-+-------------+-------------------------------+
|F|R|R|R| opcode|M| Payload len |    Extended payload length    |
|I|S|S|S|  (4)  |A|     (7)     |             (16/64)           |
|N|V|V|V|       |S|             |   (if payload len==126/127)   |
| |1|2|3|       |K|             |                               |
+-+-+-+-+-------+-+-------------+ - - - - - - - - - - - - - - - +
|     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 ...                |
+---------------------------------------------------------------+
```

WebSocket API

The API is defined by its states of readiness, responses to a networking or messaging event.

Callback	Description
`onopen`	invoked when `WebSocket session` `is established`, signalizes that the protocol is ready to transfer payload data
`onerror`	invoked whenever an `error occurs`
`onclose`	invoked when one of the peers has `terminated the session`
`onmessage`	invoked when an `incoming message` from another peer has arrived

Outline

Background
- HTTP polling
- HTTP long polling
- Streaming

WebSocket protocol
- Definition
- Handshake
- Frame
- API

Performance vs TCP Socket
- Performance Evaluation
- WebSocket TCP
- Network traffic

Performace Evaluation

- Performance evaluation of the WebSocket and the TCP Socket protocol consists of:

Performace Evaluation

- Performance evaluation of the WebSocket and the TCP Socket protocol consists of:
  - Network traffic

Performace Evaluation

- Performance evaluation of the WebSocket and the TCP Socket protocol consists of:
  - Network traffic
  - Data transfer time

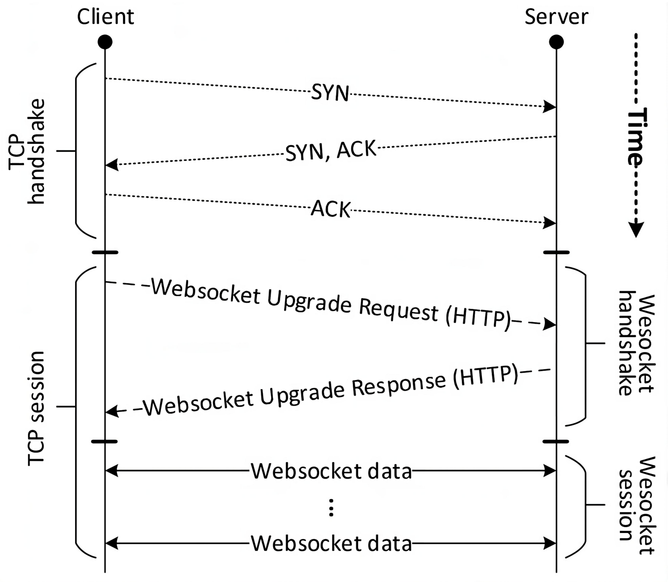
Performace Evaluation

- Performance evaluation of the WebSocket and the TCP Socket protocol consists of:
  - Network traffic
  - Data transfer time
- Network traffic is *evaluated analytically* using the protocol specifications.

Performace Evaluation

- Performance evaluation of the WebSocket and the TCP Socket protocol consists of:
  - Network traffic
  - Data transfer time
- Network traffic is *evaluated analytically* using the protocol specifications.
- Data transfer time is *evaluated experimentally* in a laboratory test bed.

WebSocket TCP sequence diagram



Analytical Evaluation of Network Traffic

- Both protocols will have the lower level protocols fields overhead such as *Ethernet, IP and TCP header*.

Analytical Evaluation of Network Traffic

- Both protocols will have the lower level protocols fields overhead such as *Ethernet, IP and TCP header*.
- For this reason, the analysis consider only the overhead the WebSocket incurs:

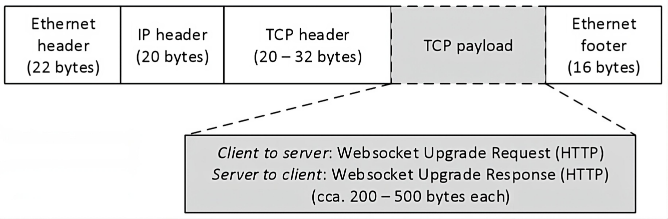
Analytical Evaluation of Network Traffic

- Both protocols will have the lower level protocols fields overhead such as *Ethernet, IP and TCP header*.
- For this reason, the analysis consider only the overhead the WebSocket incurs:
  - Handshake

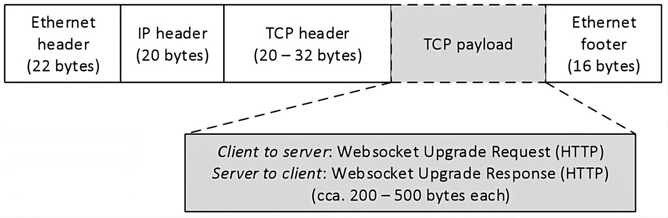
Analytical Evaluation of Network Traffic

- Both protocols will have the lower level protocols fields overhead such as *Ethernet, IP and TCP header*.
- For this reason, the analysis consider only the overhead the WebSocket incurs:
  - Handshake
  - Frame header for each frame

Handshake overhead

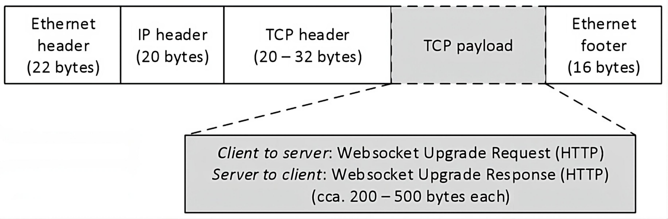


Handshake overhead



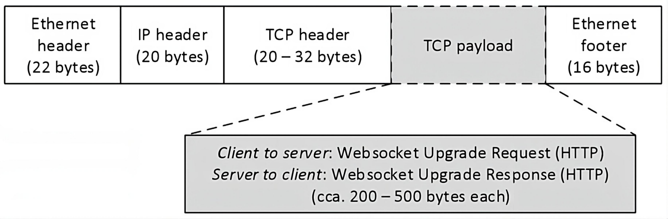
- The overhead is fixed in length and typically counts few hundreds of bytes.

Handshake overhead



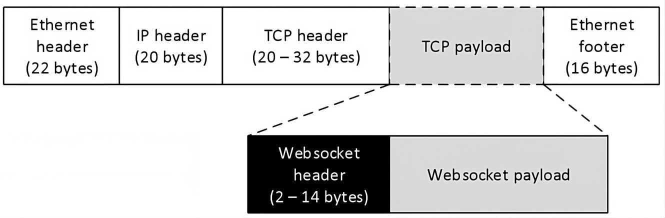
- The overhead is fixed in length and typically counts few hundreds of bytes.
- It is performed only once per session.

Handshake overhead

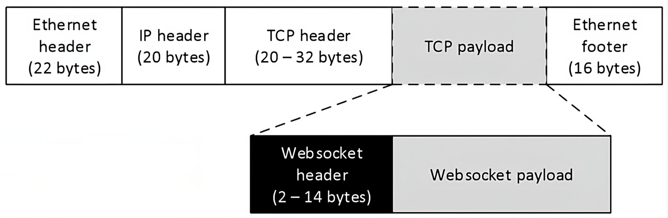


- The overhead is fixed in length and typically counts few hundreds of bytes.
- It is performed only once per session.
- Its significance decreases with the increasing number of frames sent over the same session. Thus, the evaluation is focused on long-running sessions.

Frame overhead



Frame overhead



- The overhead counts 2 to 14 bytes for each frame.

Network Traffic Overhead

- When the data are transferred with TCP Socket, they are directly embedded as TCP Payload.

Network Traffic Overhead

- When the data are transferred with TCP Socket, they are directly embedded as TCP Payload.
- With WebSocket, the TCP Payload consists of both data and Frame header.

Network Traffic Overhead

- When the data are transferred with TCP Socket, they are directly embedded as TCP Payload.
- With WebSocket, the TCP Payload consists of both data and Frame header.
- This relation can be written as:

$$P_{TCP} = data \tag{1}$$

$$P_{WS} = data + H \tag{2}$$

where:

P = payload
$data$ = data to send
H = length of frame's header

Network Traffic Overhead

- When the data are transferred with TCP Socket, they are directly embedded as TCP Payload.
- With WebSocket, the TCP Payload consists of both data and Frame header.
- This relation can be written as:

$$P_{TCP} = data \tag{1}$$

$$P_{WS} = data + H \tag{2}$$

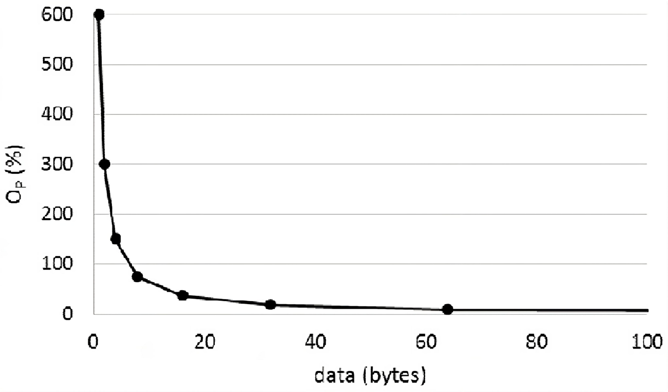
where:

$$\begin{aligned} P &= \text{payload} \\ data &= \text{data to send} \\ H &= \text{length of frame's header} \end{aligned}$$

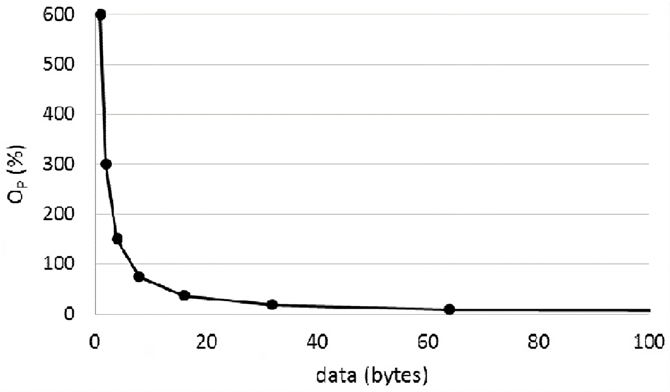
- We can now define the network traffic overhead O_p a WebSocket has over a TCP Socket:

$$O_p = \frac{P_{WS} - P_{TCP}}{P_{TCP}} \cdot 100\% \tag{3}$$

Network Traffic Overhead

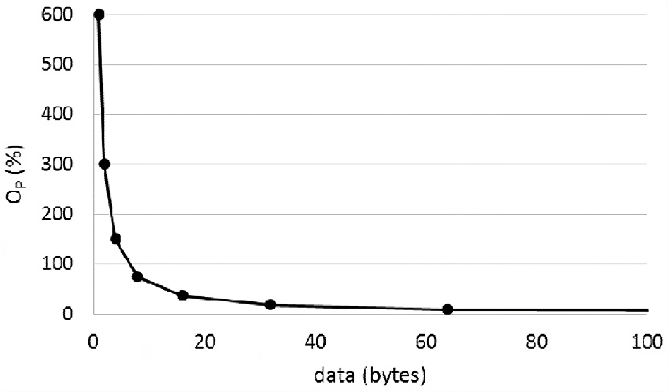


Network Traffic Overhead



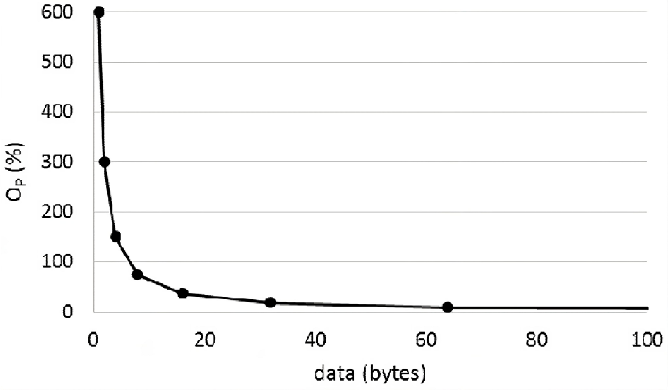
- Significant difference in performance only for tiny data.

Network Traffic Overhead



- Significant difference in performance only for tiny data.
- For biggest messages, the WebSocket frame size converges very fast towards the TCP Socket size.

Network Traffic Overhead



- Significant difference in performance only for tiny data.
- For biggest messages, the WebSocket frame size converges very fast towards the TCP Socket size.
- Except for the *initial* WebSocket Handshake, the amount of network traffic generated is comparable to that generated by to the TCP Socket.

References

- Alexey Melnikov and Ian Fette, *The WebSocket Protocol*, RFC 6455, December 2011.
- D. Skvorc, M. Horvat, and S. Srbljic, *Performance evaluation of websocket protocol for implementation of full-duplex web streams*, 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014, pp. 1003–1008.
- Lijing Zhang and Xiaoxiao Shen, *Research and development of real-time monitoring system based on websocket technology*, Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC), 2013, pp. 1955–1958.