WebSocket

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

Oleg Bilovus

Università degli Studi di Salerno

1st Scalability Research Forum

We will talk about WebSockets and compare its performance with TCP Socket. But, before diving into analyzing the performance we need to understand why we needed WebSockets and what they are.

- *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.

Bidirectional means the server and the client can send data to each other at any time

- *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.

Bidirectional means the server and the client can send data to each other at any time

- *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.

Bidirectional means the server and the client can send data to each other at any time

- *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.

Bidirectional means the server and the client can send data to each other at any time

- *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.

Bidirectional means the server and the client can send data to each other at any time

- *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.

Bidirectional means the server and the client can send data to each other at any time

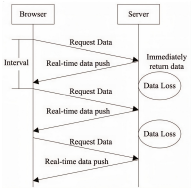
Background

- *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.

Bidirectional means the server and the client can send data to each other at any time

HTTP polling

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

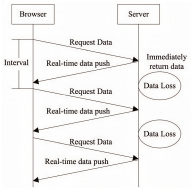


A client can send data and ask for data at the same time. But, if client has no data and server has no data, a request and response will still be generated with all the HTTP headers and thus wasting resources.No real-time data because while the client waits, an event could occur and the client will know about it only when the timeout expires.

HTTP polling

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

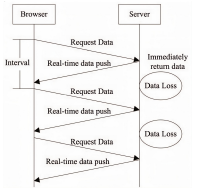
- How often to query?



A client can send data and ask for data at the same time. But, if client has no data and server has no data, a request and response will still be generated with all the HTTP headers and thus wasting resources.No real-time data because while the client waits, an event could occur and the client will know about it only when the timeout expires.

HTTP polling

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

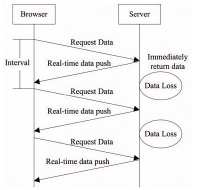


- How often to query?
- Continuously short interval requests will be washed away the server.

A client can send data and ask for data at the same time. But, if client has no data and server has no data, a request and response will still be generated with all the HTTP headers and thus wasting resources.No real-time data because while the client waits, an event could occur and the client will know about it only when the timeout expires.

HTTP polling

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

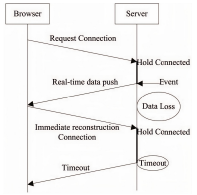


- How often to query?
- Continuously short interval requests will be washed away the server.
- Long interval will require more time to reach the client, no real-time data.

A client can send data and ask for data at the same time. But, if client has no data and server has no data, a request and response will still be generated with all the HTTP headers and thus wasting resources.No real-time data because while the client waits, an event could occur and the client will know about it only when the timeout expires.

HTTP long polling

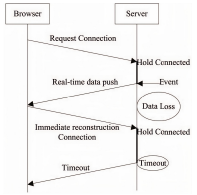
When a client sends a data request, the server will block the request until there is data transfer or timeout before returning.



Can hold the connection up to a certain time, after that a timeout is exceeded and need a new connection.No bidirectional because the client may only send data the first time, but then it will only receive until a timeout and another request is made.In the normal polling we could have bidirectional because the interval was shorter.

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.

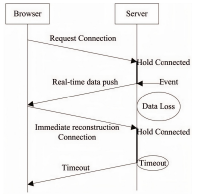


- Solve the short polling frequency to access the server.

Can hold the connection up to a certain time, after that a timeout is exceeded and need a new connection.No bidirectional because the client may only send data the first time, but then it will only receive until a timeout and another request is made.In the normal polling we could have bidirectional because the interval was shorter.

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.

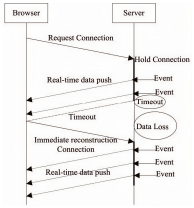


- Solve the short polling frequency to access the server.
- No bidirectional communication, server push data.

Can hold the connection up to a certain time, after that a timeout is exceeded and need a new connection.No bidirectional because the client may only send data the first time, but then it will only receive until a timeout and another request is made.In the normal polling we could have bidirectional because the interval was shorter.

Streaming

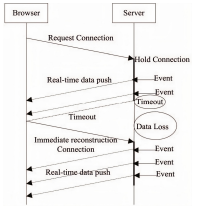
Iframe embed a hidden frame in an HTML page, then set it as a long connection request, thus the server can send data to the clients constantly.



iframe is a html page inside another.Because the server need to keep the connections alive.

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.

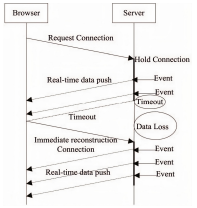


- It can send multiple events from a single request.

iframe is a html page inside another.Because the server need to keep the connections alive.

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.

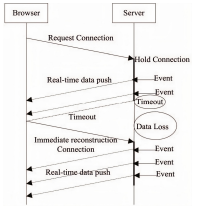


- It can send multiple events from a single request.
- But, it increases the burden on the server, causing the server performance degradation, or even collapse.

iframe is a html page inside another.Because the server need to keep the connections alive.

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.

iframe is a html page inside another.Because the server need to keep the connections alive.

- 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.

opted-in is important because with polling any HTTP server would accept it, but here additional steps are needed.Handshake means client and server have to agree that they can both use the protocol and the server has to prove it.Message framing because we do not want to send every time the headers.TCP means it is reliable, no messages will be lost.

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.

opted-in is important because with polling any HTTP server would accept it, but here additional steps are needed.Handshake means client and server have to agree that they can both use the protocol and the server has to prove it.Message framing because we do not want to send every time the headers.TCP means it is reliable, no messages will be lost.

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.

opted-in is important because with polling any HTTP server would accept it, but here additional steps are needed.Handshake means client and server have to agree that they can both use the protocol and the server has to prove it.Message framing because we do not want to send every time the headers.TCP means it is reliable, no messages will be lost.

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.

opted-in is important because with polling any HTTP server would accept it, but here additional steps are needed.Handshake means client and server have to agree that they can both use the protocol and the server has to prove it.Message framing because we do not want to send every time the headers.TCP means it is reliable, no messages will be lost.

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.

opted-in is important because with polling any HTTP server would accept it, but here additional steps are needed.Handshake means client and server have to agree that they can both use the protocol and the server has to prove it.Message framing because we do not want to send every time the headers.TCP means it is reliable, no messages will be lost.

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.

opted-in is important because with polling any HTTP server would accept it, but here additional steps are needed.Handshake means client and server have to agree that they can both use the protocol and the server has to prove it.Message framing because we do not want to send every time the headers.TCP means it is reliable, no messages will be lost.

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.

opted-in is important because with polling any HTTP server would accept it, but here additional steps are needed.Handshake means client and server have to agree that they can both use the protocol and the server has to prove it.Message framing because we do not want to send every time the headers.TCP means it is reliable, no messages will be lost.

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.

opted-in is important because with polling any HTTP server would accept it, but here additional steps are needed.Handshake means client and server have to agree that they can both use the protocol and the server has to prove it.Message framing because we do not want to send every time the headers.TCP means it is reliable, no messages will be lost.

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.

opted-in is important because with polling any HTTP server would accept it, but here additional steps are needed.Handshake means client and server have to agree that they can both use the protocol and the server has to prove it.Message framing because we do not want to send every time the headers.TCP means it is reliable, no messages will be lost.

RFC 6455
Keywords

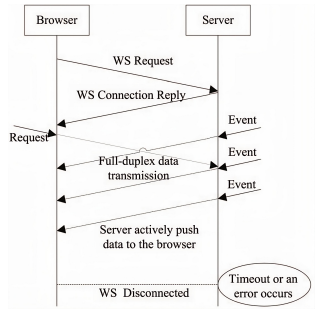
- The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code.
- The protocol consists of an opening handshake followed by basic message framing, layered over TCP.
- The goal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

opted-in is important because with polling any HTTP server would accept it, but here additional steps are needed.Handshake means client and server have to agree that they can both use the protocol and the server has to prove it.Message framing because we do not want to send every time the headers.TCP means it is reliable, no messages will be lost.

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.

opted-in is important because with polling any HTTP server would accept it, but here additional steps are needed.Handshake means client and server have to agree that they can both use the protocol and the server has to prove it.Message framing because we do not want to send every time the headers.TCP means it is reliable, no messages will be lost.



There is the initial handshake, after that, client and server can send and receive data at any moment without further interaction.There is no timeout. If it disconnects, it is because of an error and to establish the connection, the handshake has to be done again.

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

With the Upgrade Response, the server proves that it can communicate with WebSockets.

- 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.

With the Upgrade Response, the server proves that it can communicate with WebSockets.

- 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.

With the Upgrade Response, the server proves that it can communicate with WebSockets.

WebSocket
WebSocket protocol
Handshake
WebSocket Upgrade Request

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
```

Different URI can be used to identify different endpoints. A URI can be regular HTTP, another can be WebSocket.

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
```

Different URI can be used to identify different endpoints. A URI can be regular HTTP, another can be 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.

Different URI can be used to identify different endpoints. A URI can be regular HTTP, another can be 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.

Different URI can be used to identify different endpoints. A URI can be regular HTTP, another can be 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.

Different URI can be used to identify different endpoints. A URI can be regular HTTP, another can be 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 protocols.

Different URI can be used to identify different endpoints. A URI can be regular HTTP, another can be 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 protocols.
- WebSocket version.

Different URI can be used to identify different endpoints. A URI can be regular HTTP, another can be 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
```

There is a specific algorithm to generate this Header from a key.

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.

There is a specific algorithm to generate this Header from a key.

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.

There is a specific algorithm to generate this Header from a key.

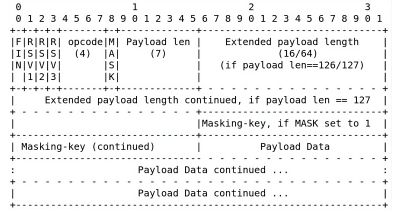
WebSocket Upgrade Response

```
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.

There is a specific algorithm to generate this Header from a key.

WebSocket Frame Structure



We will not go into the details because it is out of the scope of this presentation and, as mentioned earlier, the added overhead to the payload data is minimal.