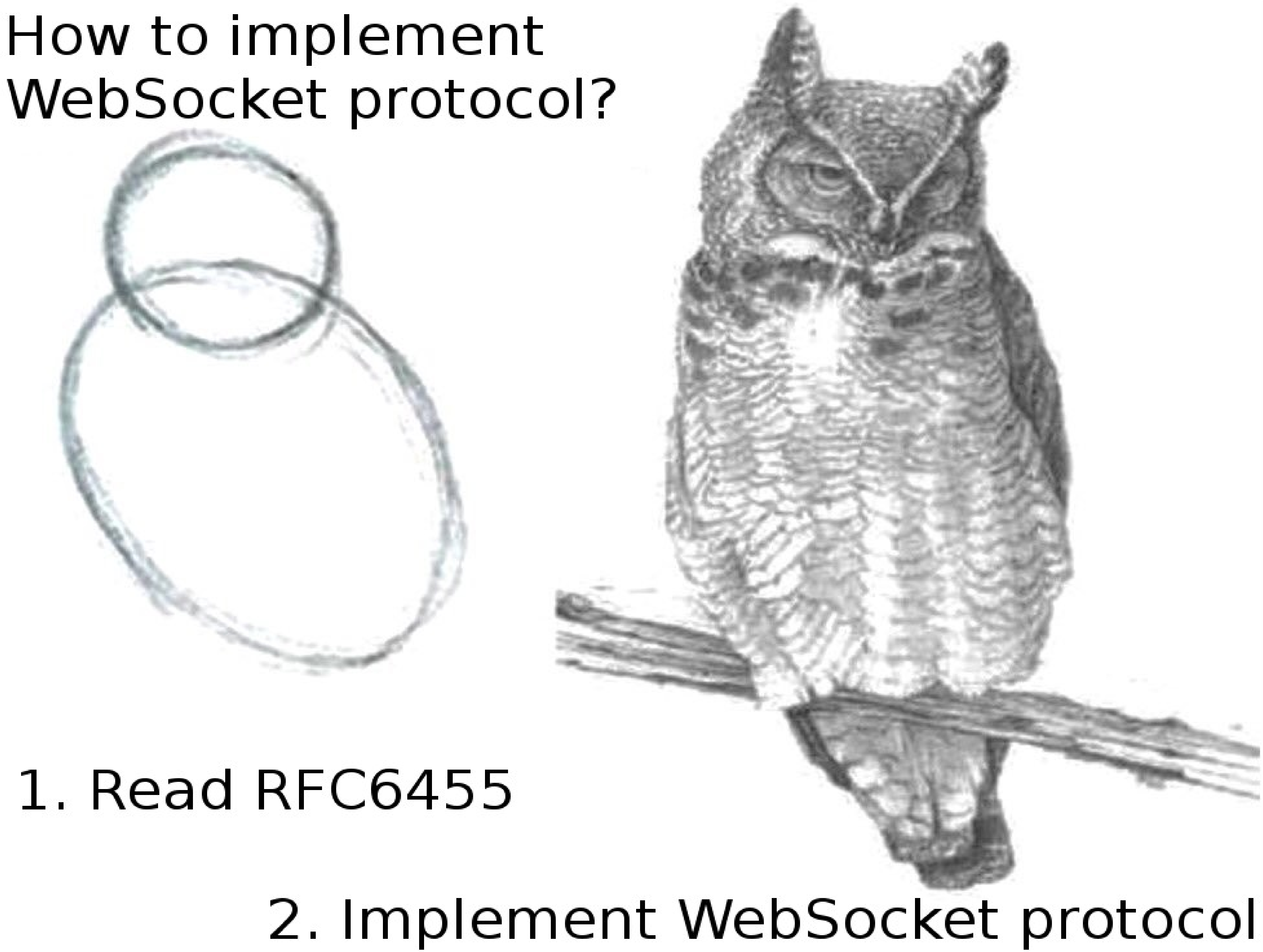
WebSocket protocol implementation in Node.JS (in details)

By Nicu Micleușanu


How to implement WebSocket protocol?
1. Read RFC6455
2. Implement WebSocket protocol

WebSocket History

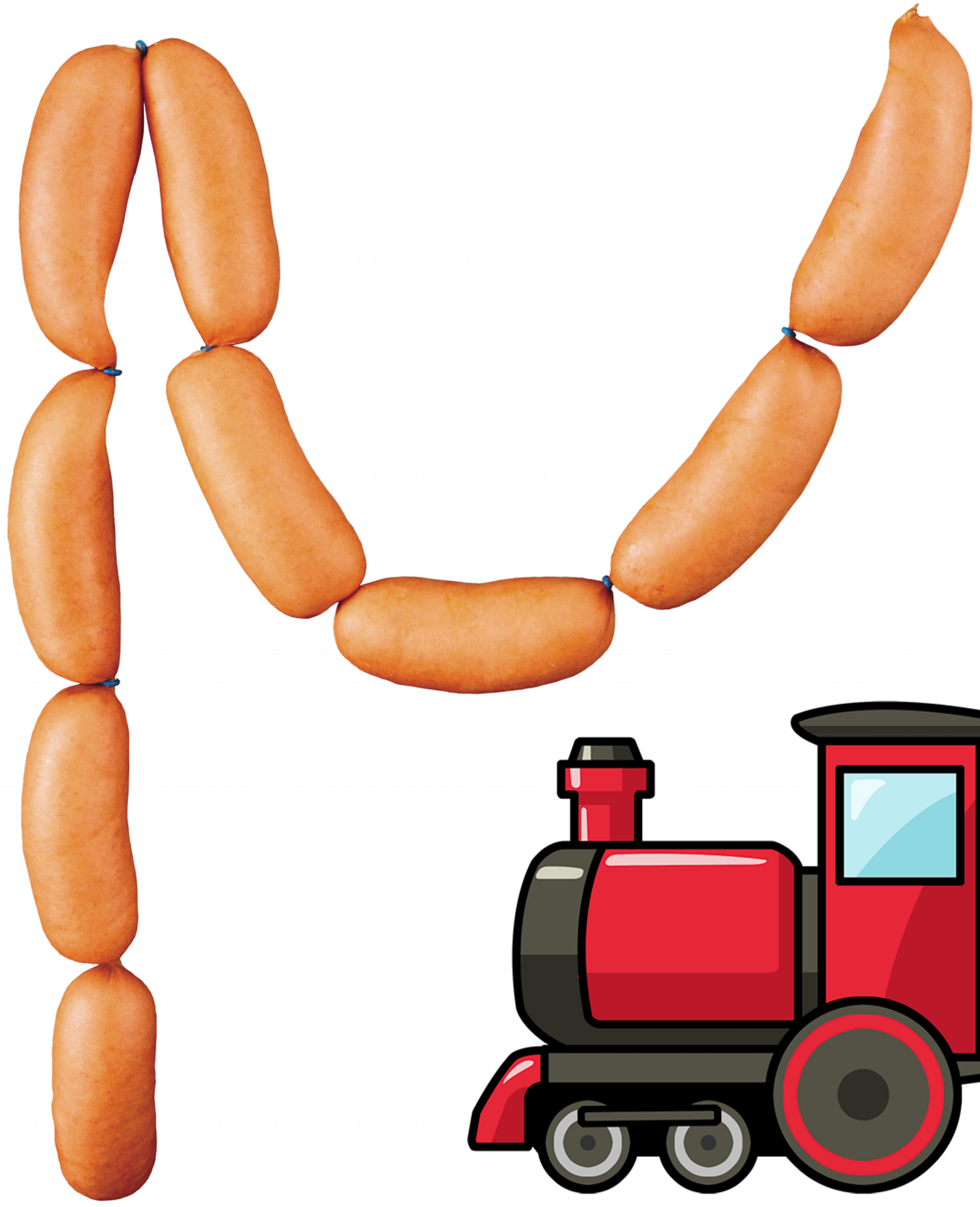
- First Idea – ~2008 as TCPConnection in HTML5
- First Implementation – June 2008

- hixie-75 – 4 February 2010 (CH4 / SF5)
- hixie-76 – 6 May 2010 (FF4 / CH6 / SF5)

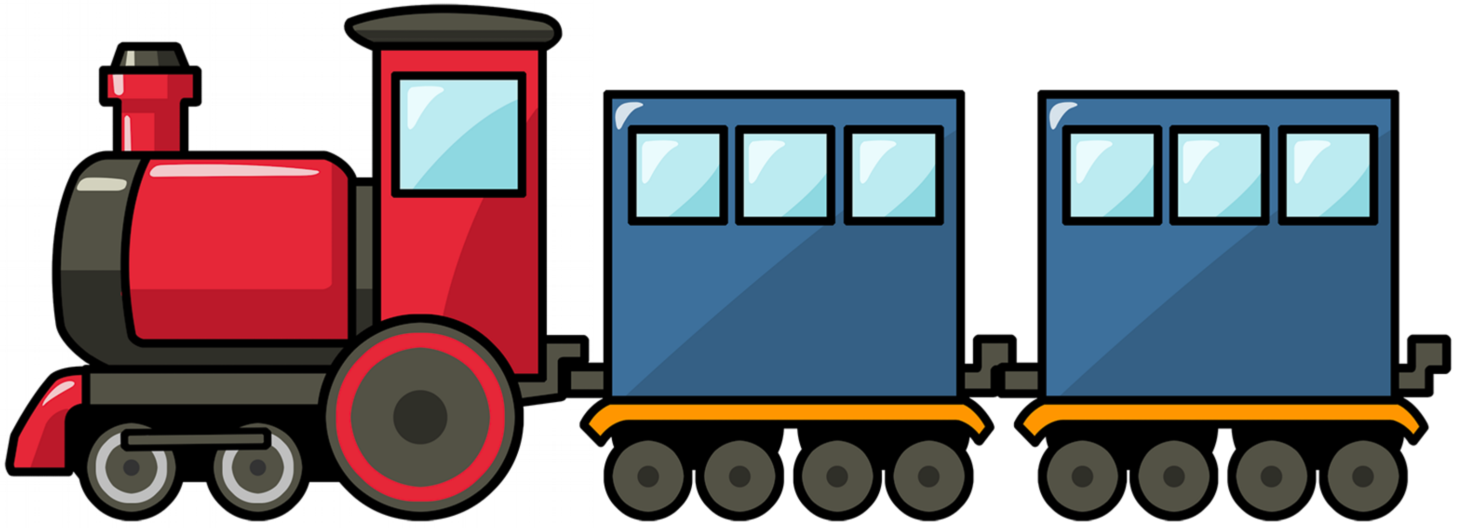
- hybi-00 – 23 May 2010
- 7 hybi-07 – 22 April 2011 (FF6)
- 8 hybi-10 – 11 July 2011 (FF7 / CH14)

- 13 RFC 6455 – December 2011
(IE10 / FF11 / CH 16)

What WebSocket really is?



Sausages or train?



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 ...                |
+---------------------------------------------------------------+
```

What do we need?

~~What we need is love!~~

Node.JS core modules:

http(s)

crypto

stream

Warning: DO NOT TRY THIS AT HOME !!!11

WebSocket HandShake

Client:

```
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
```

Server:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
```

Let’s begin

```
const http = require('http');

const server = http.Server();

server.on('request', (request, response) => {
  /* HTTP requests processing */
});

server.on('upgrade', (request, socket) => {
  /* WS requests processing */
});

server.listen(80);
```

WS Connection Listener

```
const connectionListener = (request, socket) => {

  const connection = new Connection(request, socket);

  if (validateConnection(connection)) {
    makeHandshake(connection);
    setupConnection(connection);
  }
};
```

Connection validation

```
const validateConnection = (connection) => {

  const headers = connection.headers;

  let error = '';
  let result = true;

  if (headers.upgrade.toLowerCase() !== 'websocket') {
    error = 'Invalid upgrade header';
  } else if (!headers['sec-websocket-key']) {
    error = 'No WebSocket handshake key';
  } else if (headers['sec-websocket-version'] !== '13') {
    error = 'Unsupported WebSocket version';
  }

  if (error) {
    result = false;
    connection.emit('error', new Error(error));
    connection.destroy();
  }

  return result;
};
```

Handshake response

```
const makeHandshake = (connection) => {

  const handshake = getHandshake(connection.headers['sec-websocket-key']);

  let head = '';

  head += 'HTTP/1.1 101 Switching Protocols\r\n';
  head += 'Connection: Upgrade\r\n';
  head += 'Upgrade: WebSocket\r\n';
  head += `Sec-WebSocket-Accept: ${handshake}\r\n`;
  head += '\r\n';

  connection.socket.write(head);
};
```

Handshake generation

```javascript
const crypto = require('crypto');

const wsGUID = '258EAFA5-E914-47DA-95CA-C5AB0DC85B11';

const getHandshake = (key) => {
  return crypto.Hash('sha1').update(key + wsGUID).digest('base64');
};
```

Prepare connection

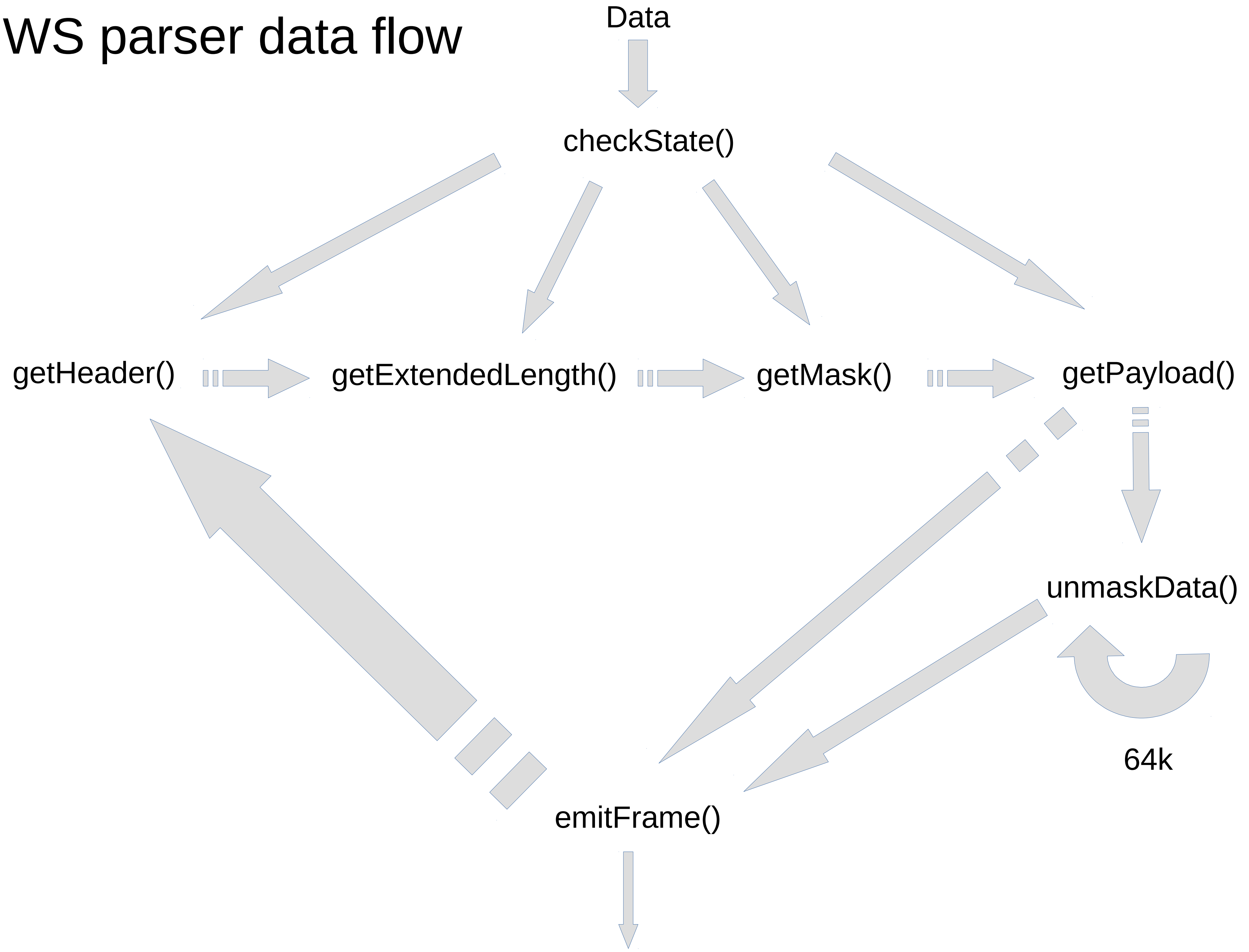
```
const ping = Buffer.from([0x89, 0x00]);
const timeout = 30 * 1000;

const setupConnection = (connection) => {

  const parser = new Parser(/* options */);

  connection.socket.setTimeout(timeout);
  connection.socket.write(ping);

  connection.on('close', () => {
    /* Remove connection */
  });

  /* Use connection */

  connection.pipe(connection.socket).pipe(parser);

  parser.on('frame', (frame) => {
    if (frame.opcode === 8) { // close
      connection.end();
    } else if (frame.opcode === 9) { // ping
      /* Send pong frame */
    } else if (frame.opcode !== 10) {
      prepareMessage(connection, frame);
    }
  });
};
```

Prepare message

```
const prepareMessage = (connection, frame) => {
  if (frame.opcode === 0) {
    frame.message.data = Buffer.concat([frame.message.data, frame.data]);
  } else {
    frame.message = {};

    frame.message.data = frame.data;

    if (frame.opcode === 1) {
      frame.message.type = 'text';
    } else {
      frame.message.type = 'binary';
    }
  }

  if (frame.fin) {
    if (frame.message.type === 'text') {
      frame.message.data = String(frame.message.data);
    }

    connection.emit('message', frame.message);
  }
};
```

Too much code ...

WS parser data flow



What’s next?

Text mode
Binary mode
Object mode

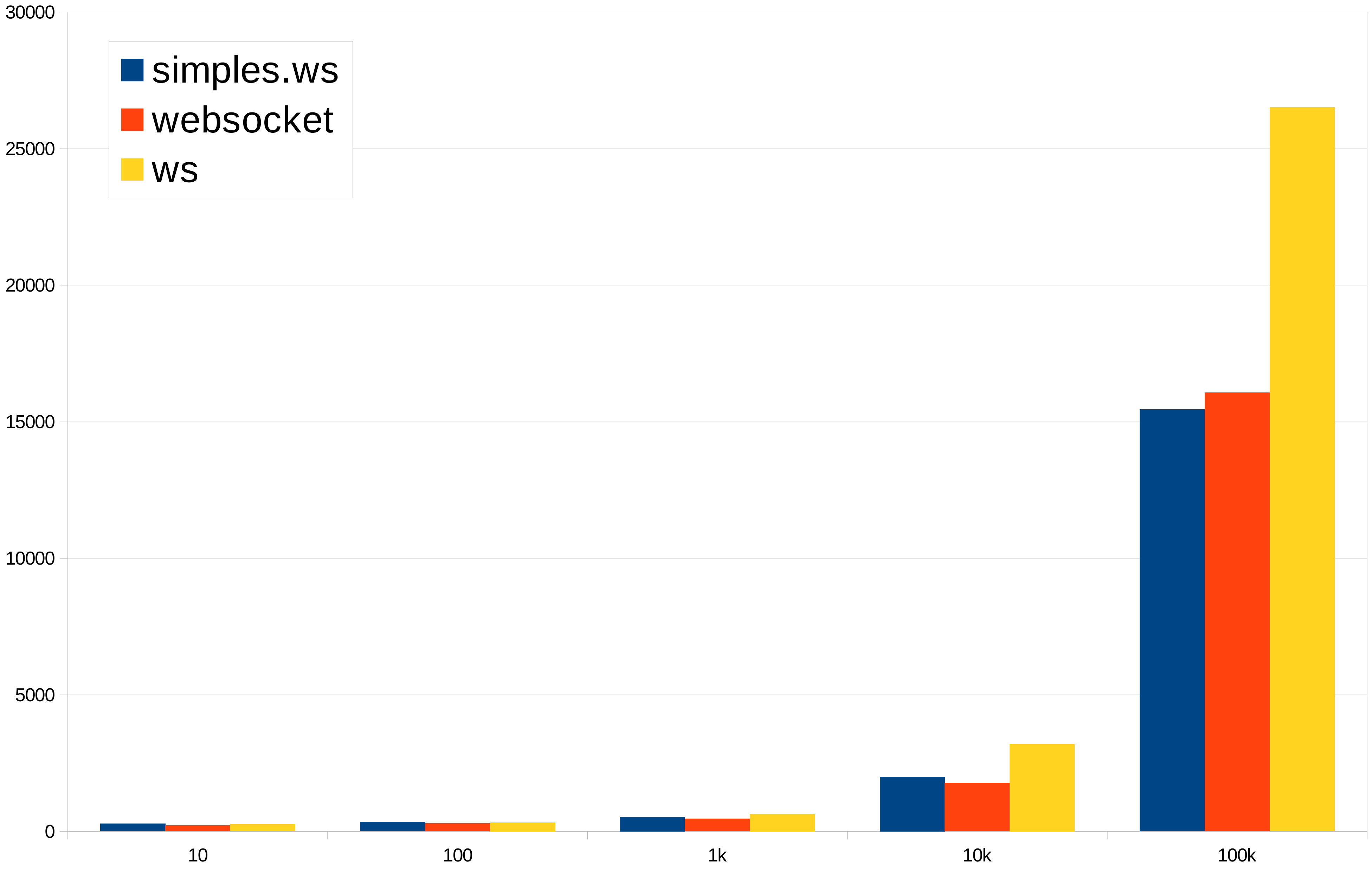
Messages broadcasting

Channels

Session

Template engine

Benchmark time



HAPPY END

Support simpleS with a star :)

https://github.com/micnic/simpleS/

https://www.npmjs.com/package/simples