



Hochschule
Albstadt-Sigmaringen

Albstadt-Sigmaringen University

WebSockets

→ bspw. Notifications
→ real time Fähigkeit



Dipl. Ing. Sven Eppler (FH)
sodge IT GmbH

A white speech bubble with a dark blue outline, pointing downwards and to the left. The bubble is centered horizontally and contains the text "Welcome to the real time web!".

Welcome to the real time web!

Wozu WebSockets?

- Real-Time Updates zwischen Client und Server
 - HTTP als Request-Response-Protokoll erlaubt keine Server-to-Client Kommunikation
 - Ausgenommen: HTTP/2 ServerSendEvents *→ aber sehr umständlich*
→ trotzdem Web-Sockets
→ über keep alive (hacky)
- Verhindern von (long) Polling
 - Immer wieder Fragen „Gibt es was neues für mich?“
 - Schlechte Performance, hohe Last auch wenn nichts passiert.

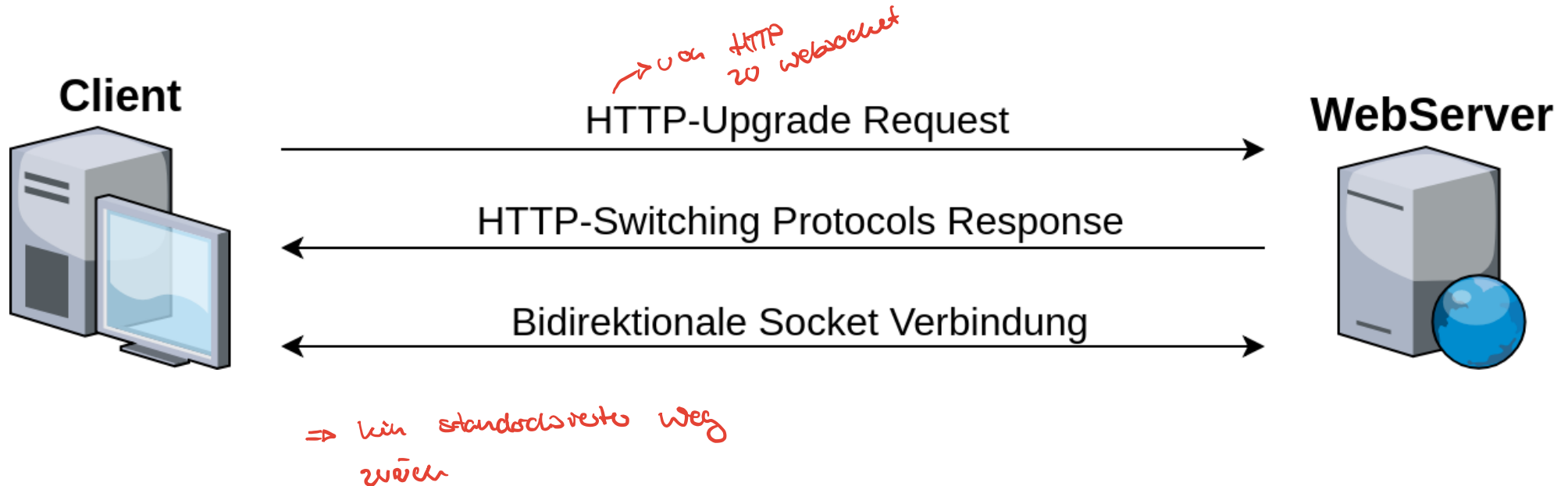
Was sind WebSockets

- WebSockets ist ein Protokoll um über eine HTTP-Verbindung eine bidirektionale Socket-Verbindung zu initiieren
 - ↳ NW-Verbindung auf OS-Ebene → immer TCP/IP → wird dann auf Sdpw HTTP gewechselt
- Client und Server können dann jederzeit Daten senden bzw. Daten empfangen
- Spezifiziert in RFC6455 im Dezember 2011

⇒ sehr weit verbreitet

WebSocket Handshake

- Der WebSocket Handshake ist ein spezieller HTTP-Request-Response Zyklus der zu einer bidirektionalen Socket Verbindung führt



WebSocket Handshake

- HTTP Upgrade Request

- Wird vom Client zum Server geschickt um eine WebSocket-Verbindung zu starten

```
GET /guestbook-websocket/echo HTTP/1.1
Connection: Upgrade      (instead of "keep-alive")
Upgrade: websocket      → wohin upgraden
Host: localhost:8080
Sec-WebSocket-Version: 13
Sec-WebSocket-Key: EchPITmg0D5XsQiAQb1xDw==
Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits
User-Agent: Mozilla/5.0 (X11; Linux x86_64) OPR/60.0.3255.59
Origin: http://localhost:8080
```

WebSocket Handshake

- HTTP Switching Protocols Response

- Wird vom Server zum Client geschickt, wenn der Server die WebSocket-Verbindung annehmen möchte

```
HTTP/1.1 101 Switching Protocols
Server: Apache-Coyote/1.1
Upgrade: websocket
Connection: upgrade
Sec-WebSocket-Accept: VcEH0BKcMSH8rqBZw+KcTGS31hk=
Sec-WebSocket-Extensions: permessage-deflate;client_max_window_bits=15
Date: Sun, 26 May 2019 18:04:36 GMT
```

↳ Ext. auf die man sich
geeignet hat

WebSocket Handshake

- Besondere Header

- Sec-WebSocket-Version

- Welche Version des WebSocket Protokolls wird benutzt

- Sec-WebSocket-Key

- Base64 Encoded Zufallswert des Clients

- Sec-WebSocket-Accept

- Antwort auf den Sec-WebSocket-Key:

$\text{Base64}(\text{SHA1}(\text{Sec-WebSocket-Key} + \text{FixedUUID}))$
 $\text{FixedUUID} = 258EAF55-E914-47DA-95CA-C5AB0DC85B11$

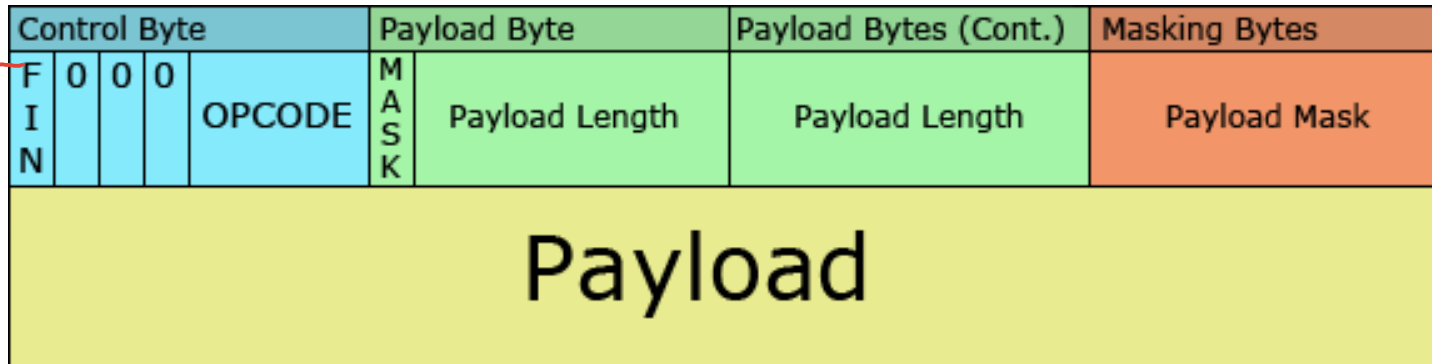
*Proof of knowledge
→ damit Server nicht nur Header spiegelt
und websocket Req nicht versteht.*

- Sec-WebSocket-Extensions

- Welche Extensions der Client/Server unterstützt

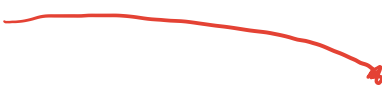
WebSocket Frame Format

- **WebSocket Frame Format** *→ im Binär-Format*
 - OPCODE: Text/Binär/Close/Ping/Pong Message
 - Payload Length: Wie viele Bytes kommen im Payload (max 2^{15} ~~63~~ *15 Bit* byte) *→ selbe Nachricht steht anders aus*
 - Masking Bytes: Bytes zur XOR Maskierung des Payloads um Cache-Poisoning-Attacks zu verhindern *(zufällige Wert)* *→ web socket Nachrichten von anderen lesen*



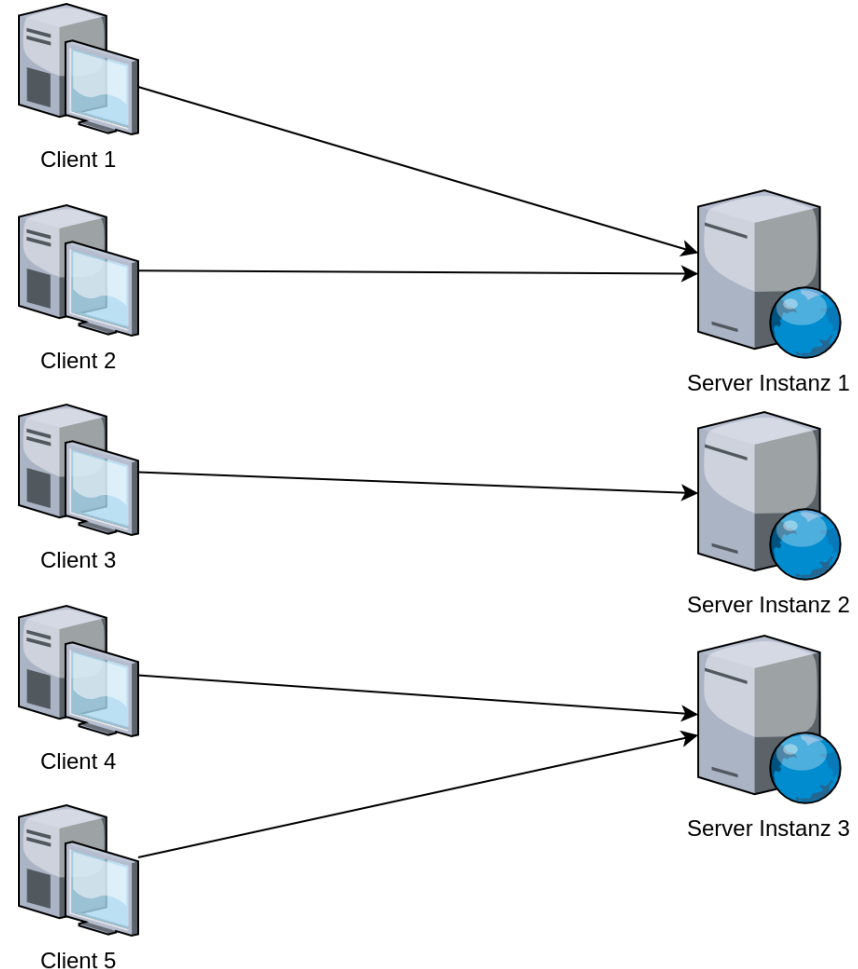
Quelle: <https://github.com/OpenSuede/Suede/wiki/Websocket-Structure>

Architektur Probleme mit WebSockets

- WebSockets brechen mit dem stateless Prinzip, daher wird die Skalierung erschwert
 - Eine WebSocket-Verbindung ist an einen Server-Instanz gebunden
 - Erschwert den LoadBalancer Einsatz!
 - Kommen mehrere Instanzen zum Einsatz, müssen diese miteinander kommunizieren können
 - Dafür benötigt es ein übergeordnetes InterProcessCommunication (IPC) System
- 

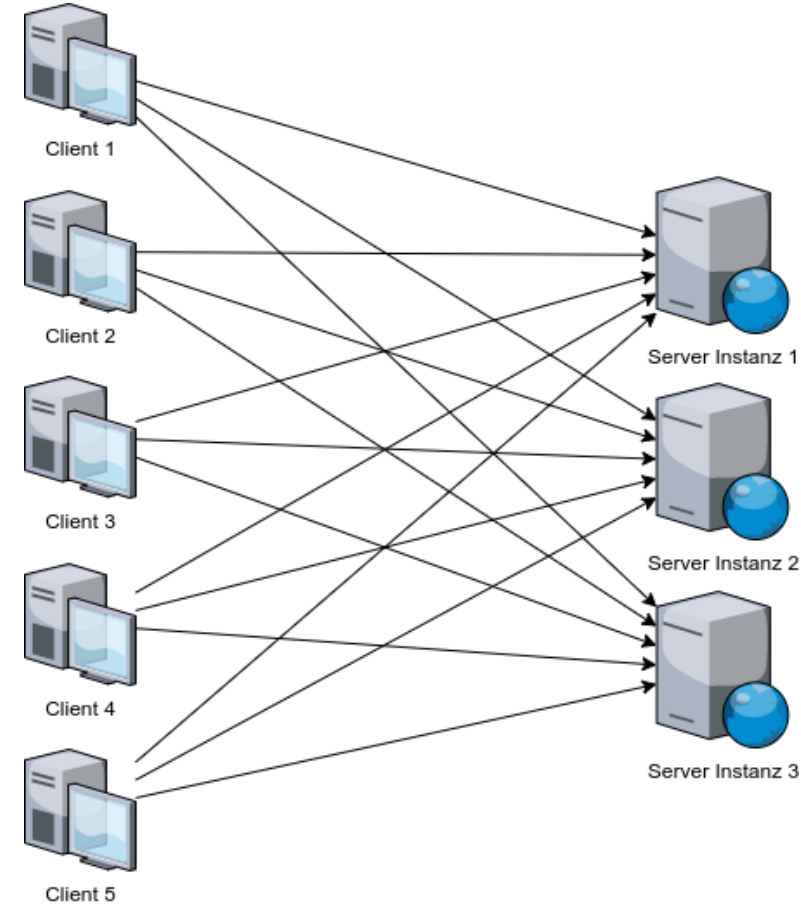
Architektur Probleme mit WebSockets

- Wie kann Client 1 eine Nachricht an Client 5 schicken?
 - In diesem Setup: Gar nicht!
- Die Server können untereinander die Nachrichten nicht weiterleiten!



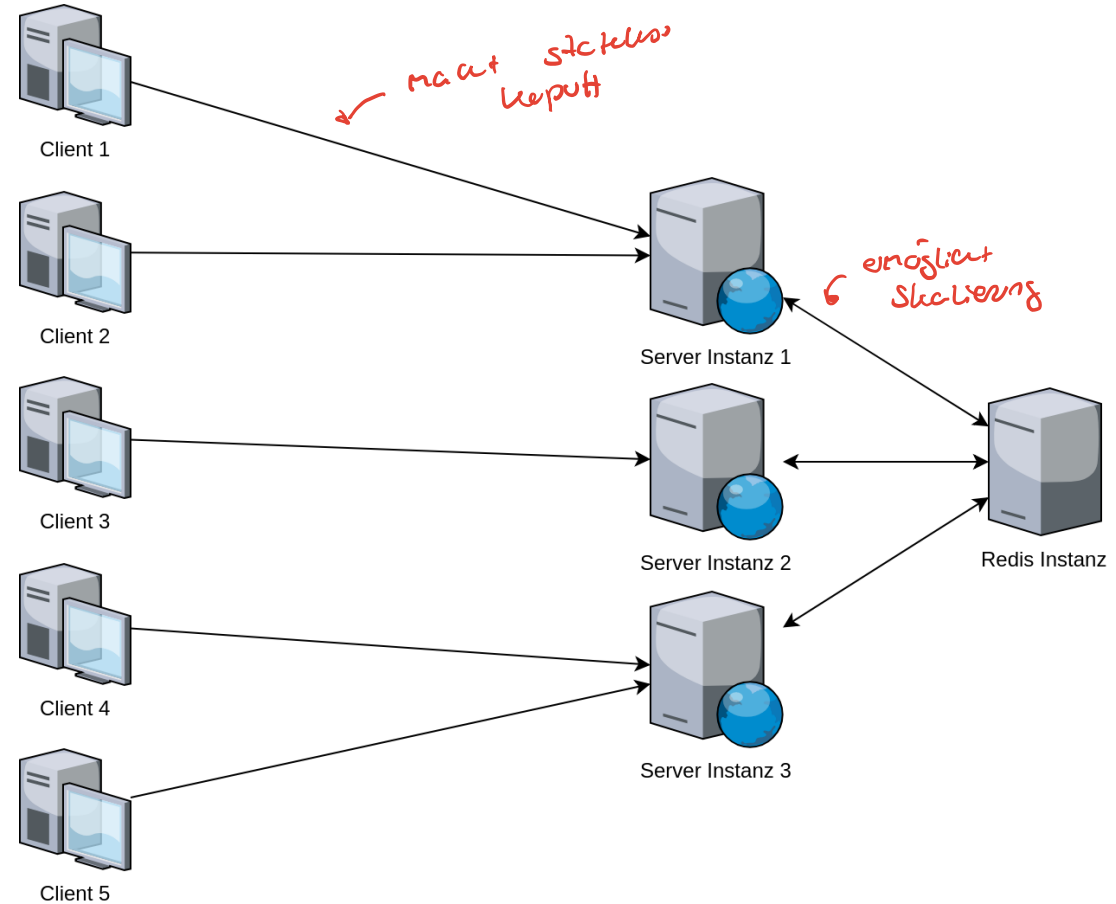
Pseudo-Lösung: Mesh-Netzwerk

- „Theoretische“ Lösung
- Anzahl der Verbindungen sprengt schnell die technischen Möglichkeiten
- Anzahl Connections = Anzahl Clients * Anzahl Server
- Routing innerhalb dieser Connections nicht trivial



Lösung: Architektur Probleme mit WebSockets

- Alle Server-Instanzen werden mithilfe einer Redis-Instanz verbunden
- Redis ist ein In-Memory Key-Value Store mit Publish-Subscribe Schnittstelle *→ Notifikationen wenn sich etwas für bestimmten Client ändert*
- Neben Redis existieren noch andere Lösungen z.B. bietet auch PostgreSQL PubSub oder RabbitMQ



WebSocket Anwendungsgebiete

- Überall wo real-time Daten notwendig sind:
 - Livechats
 - Online-Spiele (z.B. surviv.io)
 - Notifications (z.B. Facebook, Twitter)
 - Börsenkurse (z.B. cryptowat.ch)

Live Demo

- Beispielanwendung:
Das Gästebuch mit WebSocket-Kommunikation
- Zu finden im Repository unter:
`./Beispiele/Node.js/GuestbookWebsocket`