P2P Content Sharing mit WebRTC

Christian Vogt - christian.vogt@haw-hamburg.de
Max Jonas Werner - maxjonas.werner@haw-hamburg.de

Agenda

Agenda

- Web Plattform
- WebRTC Technology
- WebRTC Demos
- BOPlish
- BOPlish Demo
- Ausblick

Web Plattform

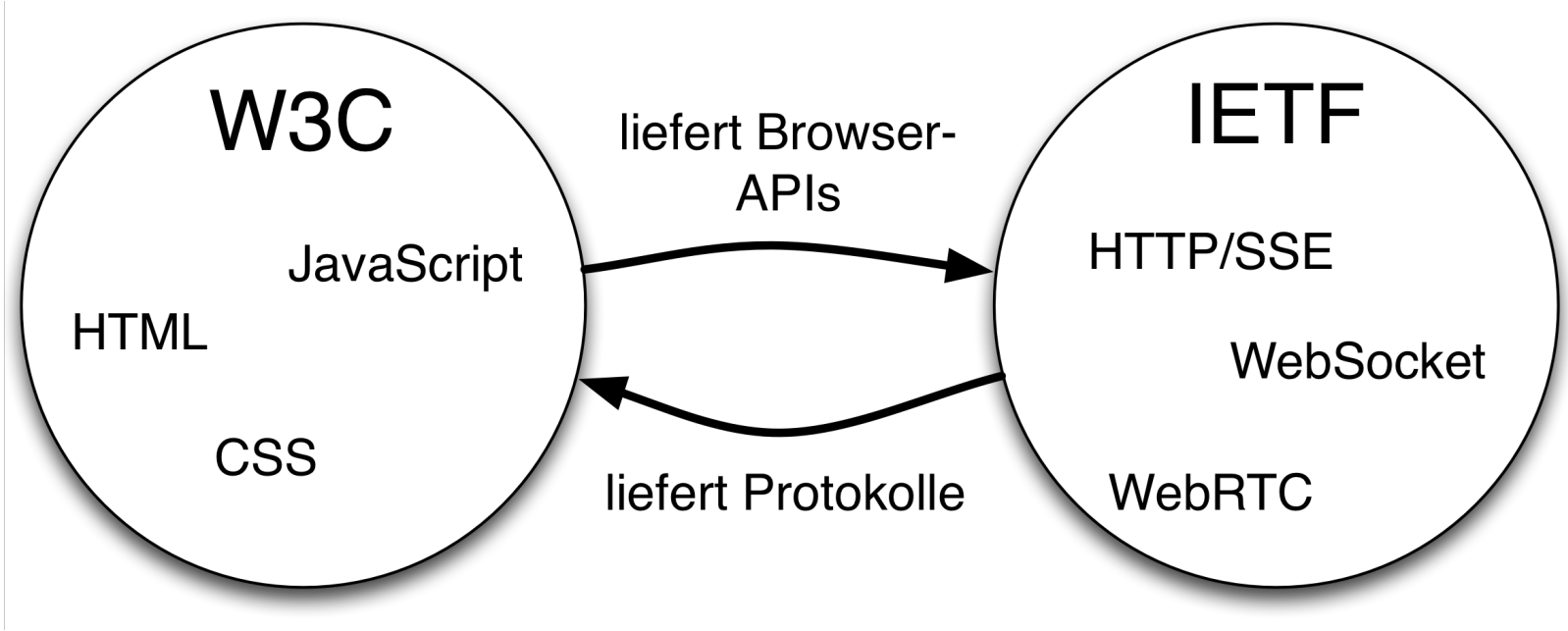
Web Plattform

Technologieüberblick

- Interface - Browser
- Struktur - HTML
- Aussehen - CSS
- Logik - JavaScript
- Datenübertragung - HTTP,WebSocket,WebRTC

Web Plattform

Technologieüberblick



Web Plattform

Vergleich der Protokolle zur Datenübertragung

	Async	Architektur	Kommunikation
HTTP	nein	Client/Server	unidirektional (Client iniitiert Verb.)
Server-Sent Events	ja	Client/Server	unidirektional (Server initiiert Verb.)
WebSocket	ja	Client/Server	bidirektional
WebRTC	ja	Peer-To-Peer	bidirektional

Web Plattform

Vergleich der Protokolle zur Datenübertragung

	Async	Architektur	Kommunikation
HTTP	nein	Client/Server	unidirektional (Client iniitiert Verb.)
Server-Sent Events	ja	Client/Server	unidirektional (Server initiiert Verb.)
WebSocket	ja	Client/Server	bidirektional
WebRTC	ja	Peer-To-Peer	bidirektional

WebRTC - **Web** **R**eal **T**ime **C**ommunication

WebRTC

Überblick (1)

Features

- Multi-Plattform (Browser, VoIP Equipment, ...)
- Peer-To-Peer Verbindung
- JavaScript API ohne Plugins

WebRTC

Überblick (1)

Features

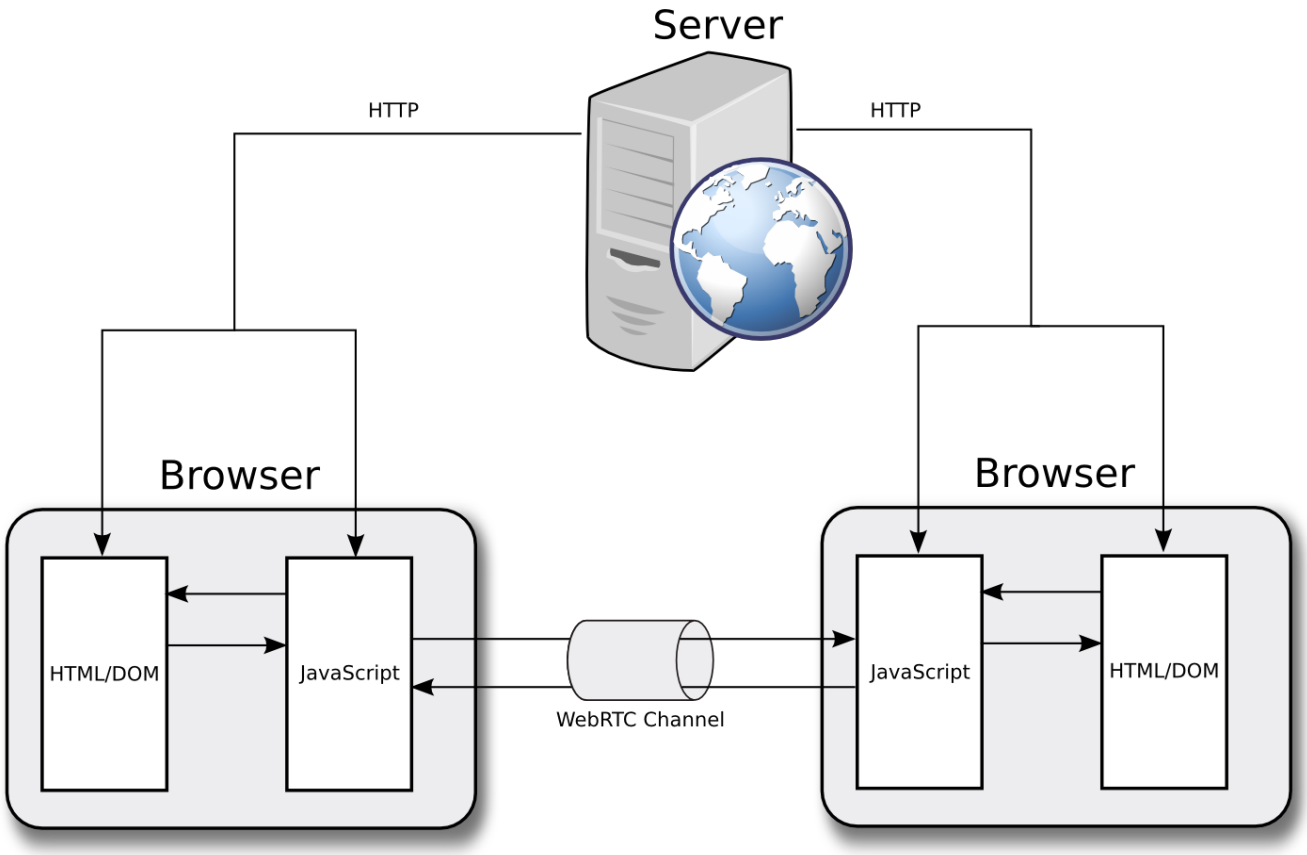
- Multi-Plattform (Browser, VoIP Equipment, ...)
- Peer-To-Peer Verbindung
- JavaScript API ohne Plugins

Funktionen

- Verbindung zu anderen Peers herstellen
- Audio/Video-Streams (Zugriff und Übertragung)
- Datenübertragung

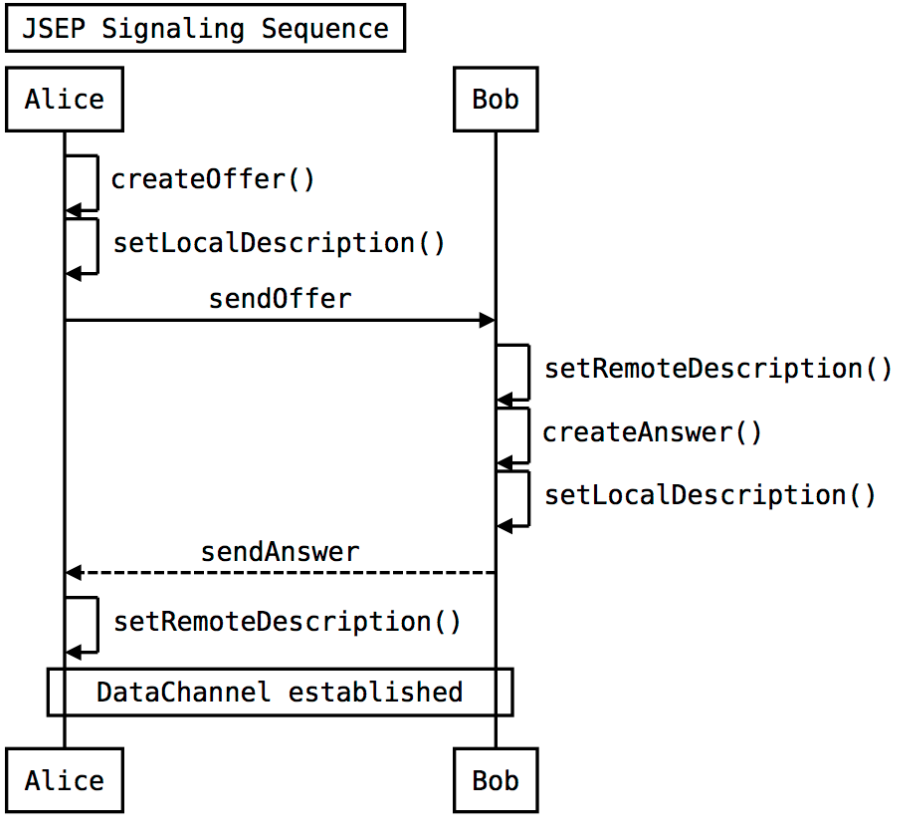
WebRTC

Überblick (2)



WebRTC

Signalisierung/Verbindungsaufbau



WebRTC

Signalisierung/Verbindungsaufbau

JAVASCRIPT

```
// Alice
var pc = new window.RTCPeerConnection();
channel.onmessage = function(msg){
  pc.setRemoteDescription(msg);
};
function createOfferCallback(offer) {
  pc.setLocalDescription(offer);
  channel.send(offer);
}
pc.createOffer(createOfferCallback);
```

JAVASCRIPT

```
// Bob
var pc = new window.RTCPeerConnection();
function createAnswerCallback(answer) {
  channel.send(answer);
}
channel.onmessage = function(msg){
  pc.setRemoteDescription(msg);
  pc.createAnswer(createAnswerCallback);
};
```

WebRTC

Mediastream

JAVASCRIPT

```
// Alice
navigator.getUserMedia({ "audio": true, "video": true }, function (stream) {
  pc.addStream(stream);
});
```

JAVASCRIPT

```
// Bob
pc.onaddstream = function(evt) {
  aliceView.src = URL.createObjectURL(evt.stream);
};
```

WebRTC

Data Channel

JAVASCRIPT

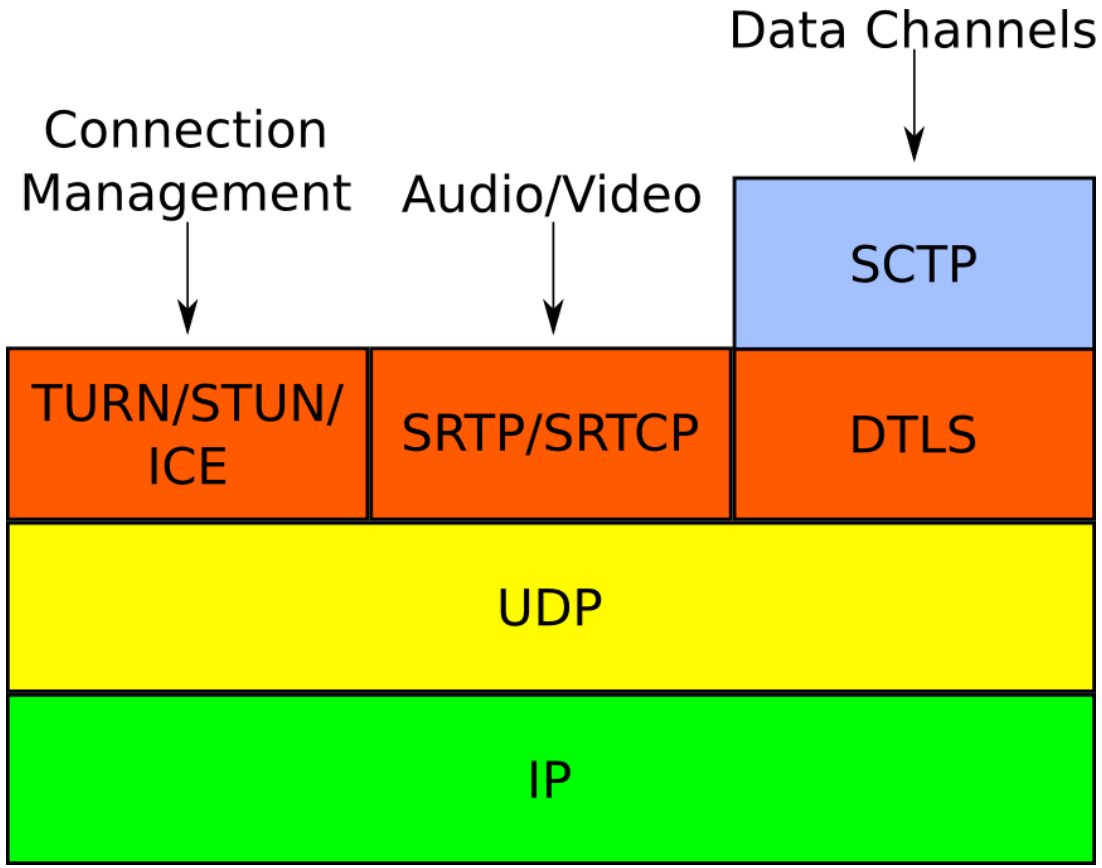
```
// Alice
var dc = pc.createDataChannel();
dc.onopen = function() {
  dc.send('hi bob');
};
```

JAVASCRIPT

```
// Bob
pc.ondatachannel = function(evt) {
  var dc = evt.target;
  dc.onmessage = function(msg) {
    alert(msg); // prints 'hi bob'
  };
};
```

WebRTC

Protokollstack



WebRTC

Protokollübersicht

- Verbindungsaufbau
  - Signalisierung - JSEP/SDP
  - NAT Traversal - TURN/STUN/ICE
- Audio/Videoübertragung - SRTP/SRTCP
- Datenübertragung - SCTP

WebRTC

Zusammenfassung

- Vorteile
  - Diverse neue Use Cases
  - Massives Deployment
  - Lösung für NAT-Traversal integriert
- Beschränkungen
  - Signalisierungs-Server wird benötigt
  - Generelle Punkt-zu-Punkt Verbindungen, kein P2P System

WebRTC - Demos

Demos

- Audio/Video-Konferenzen
- Filesharing
- ASCII Camera

```
window.RTCPeerConnection
```

```
navigator.getUserMedia
```

BOPlish - Browser-based Open Publishing

Content Sharing

Benutzer möchten...

- aus dem Browser heraus Content verteilen,
- mit möglichst wenig Abhängigkeit zu fremder Infrastruktur
- und der Content soll auch verfügbar sein, wenn sie selbst offline sind

Content Sharing

Was nutzt man jetzt?

- Dropbox/Google Drive
- Flickr
- Facebook

Content Sharing

Was nutzt man jetzt?

- Dropbox/Google Drive
- Flickr
- Facebook
- Eigener Server

Zentralisierte Services, keine Kontrolle über Content und Zugriff

Was ist also nötig?

Kontrolle über Inhalte und Infrastruktur in Nutzerhand geben

Was ist also nötig?

Kontrolle über Inhalte und Infrastruktur in Nutzerhand geben

Wie macht man das?

Aufbau einer User-zentrischen Infrastruktur, die Inhalte von Anbietern entkoppelt

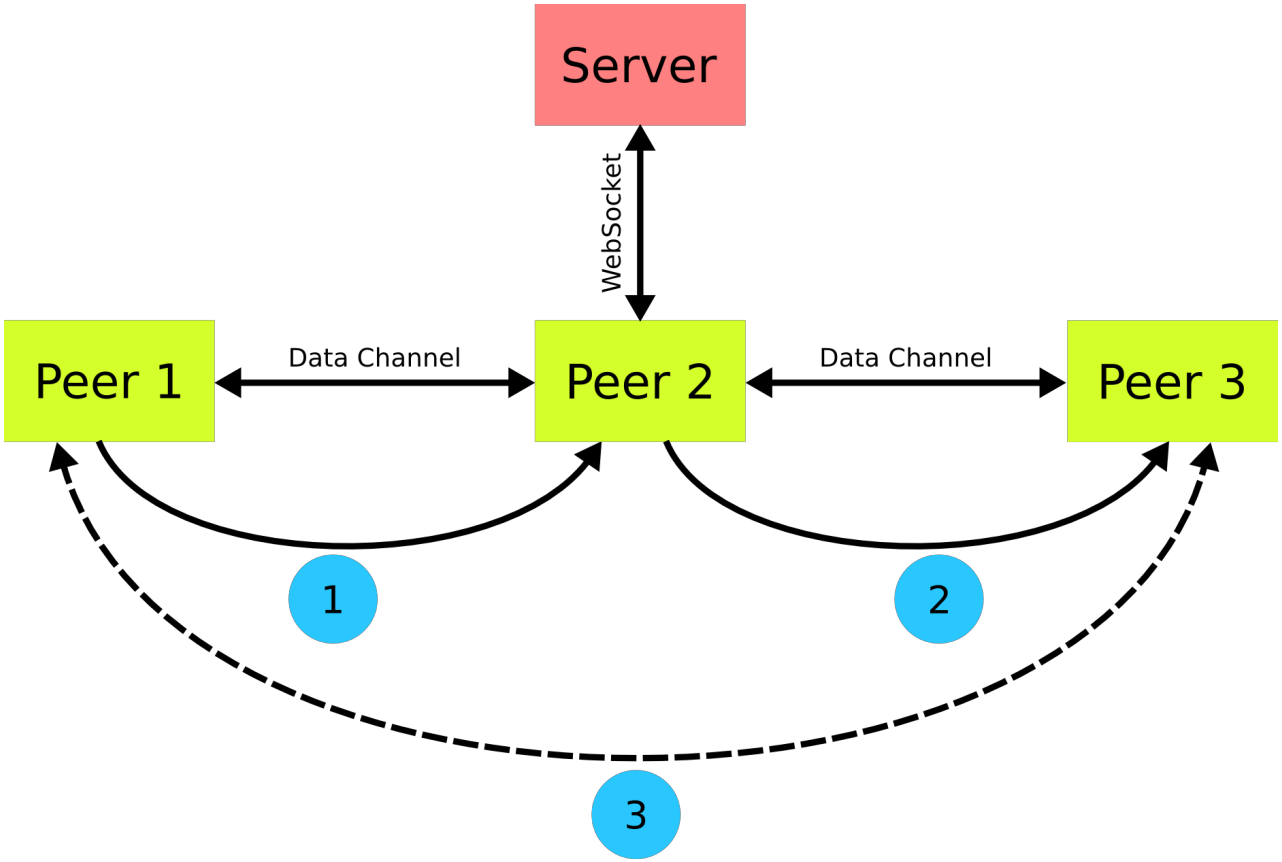
BOPlish

Virtuelle Peer-to-peer Infrastruktur aufbauend auf WebRTC

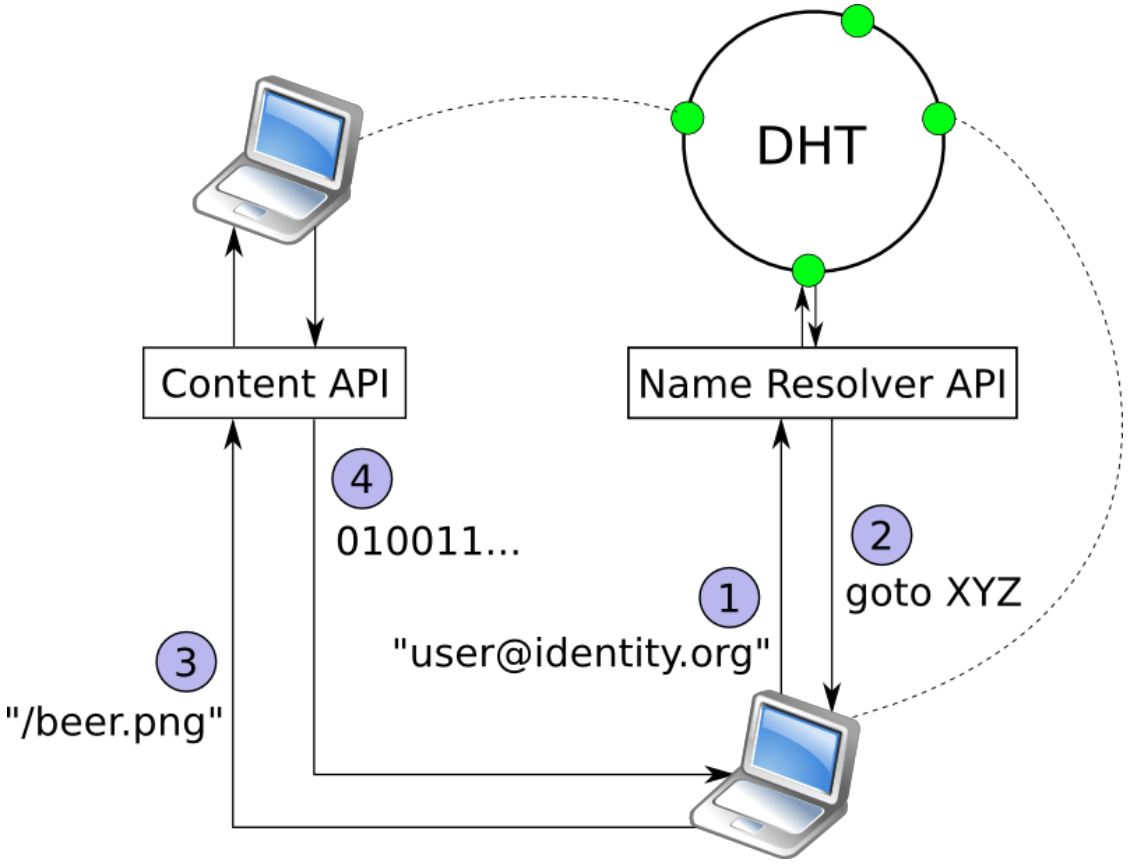
Einfaches Protokoll für Bootstrap-Signalisierung

Drop-in JavaScript Bibliothek

1. P2P-Netzaufbau



2. BOPlish Architektur



3. Content wird mit URIs adressiert

`bop://<user-identity>/<path>?<sec-credentials>`

Beispiele:

`bop://alice@example.org/public/*.ogg`

`bop://bob@example.com/secret.doc?hash=urn:sha1:<hash>`

3. Content wird mit URIs adressiert

`bop://<user-identity>/<path>?<sec-credentials>`

- Inspiriert von der Common Multicast API
- Geräte/Verbindungs-unabhängig (keine IP-Adresse)
- Infrastruktur-unabhängig (keine direkte DNS-Auflösung)

BOPlish

- Infrastruktur für Naming und Content Sharing zwischen Browsern
- Namensauflösung mittels P2P-Layer
- Content API für Applikations-spezifische Protokolle

➔ Server-less Web

BOPlish - Demos

BOPlish - Ausblick/Diskussion

BOPlish – Ausblick

- Sicherheit: Identifier, Authentifizierung, Autorisierung
- Performance-Messungen (Emulator)
- Erweiterungen für andere Use Cases (Offloading, Publish/Subscribe, …)
- Publish/Subscribe
- Bachelor-PO #RTCDime
- https://github.com/boplish

BOPlish – Ausblick

- Sicherheit: Identifier, Authentifizierung, Autorisierung
- Performance-Messungen (Emulator)
- Erweiterungen für andere Use Cases (Offloading, Publish/Subscribe, …)
- Publish/Subscribe
- Bachelor-PO #RTCDime
- https://github.com/boplish

Fragen?

Vielen Dank +
Frohe Weihnachten!

Christian Vogt - christian.vogt@haw-hamburg.de
Max Jonas Werner - maxjonas.werner@haw-hamburg.de