

Stream-Up your life!

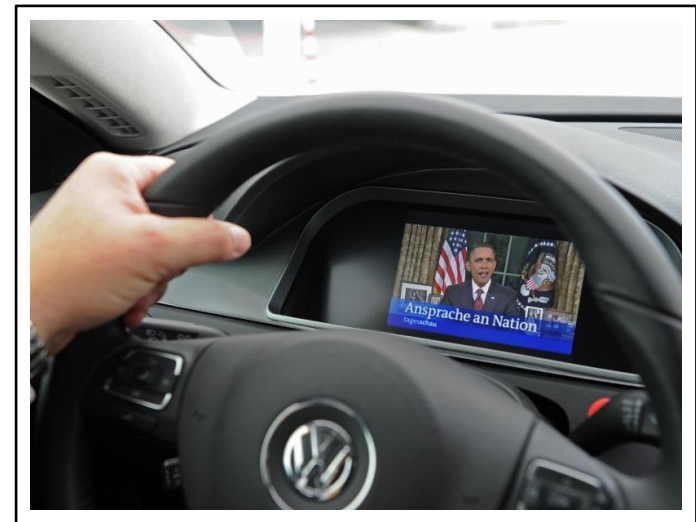
- Live-Video-Streaming auf Basis von Web-Standards -



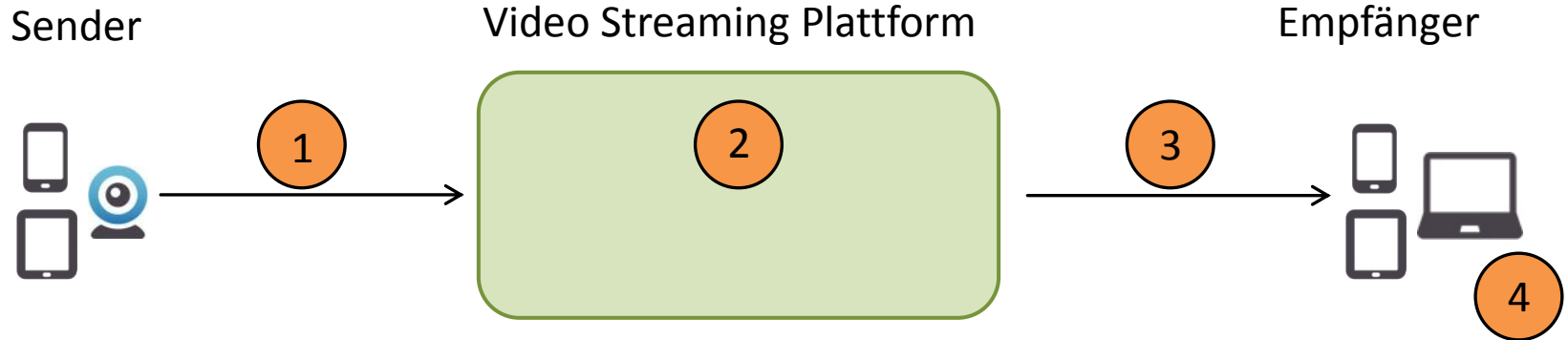
Guido Oelmann

JavaLand 2016

Video Streaming Plattformen



Ziel

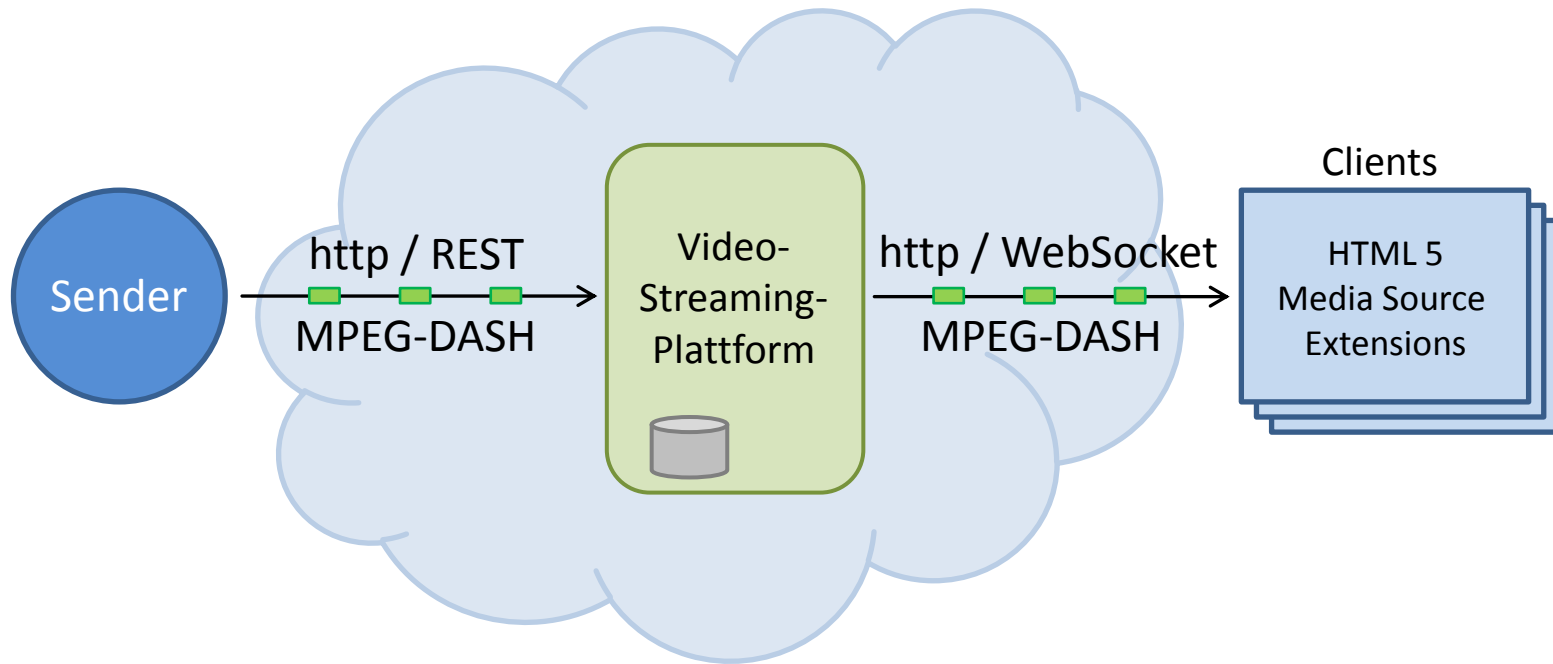


- Wie verpacken?
- Wie übertragen?

- Wie entgegennehmen?
- Wie verteilen?

- Wie übertragen?
- Wie anzeigen?

Lösung



mpeg-DASH

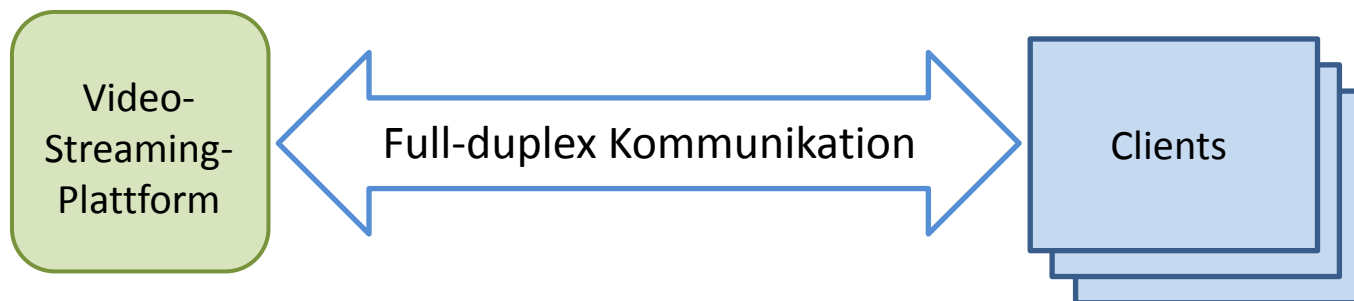


REST und WebSocket

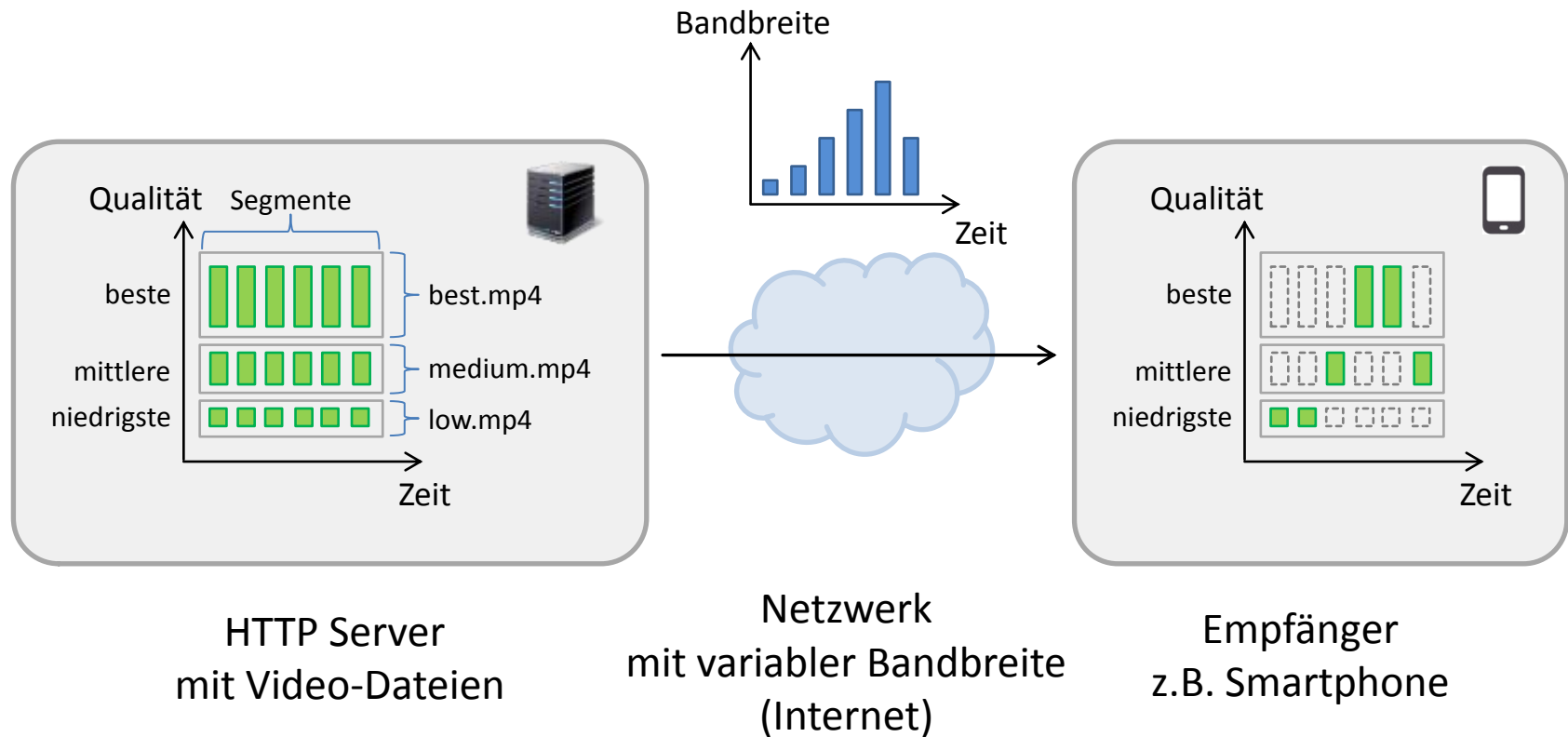
Java API for RESTful Web Services (JAX-RS)



Java API for WebSocket



Dynamic Adaptive Streaming over HTTP (DASH)



HTML 5 Video-Element

- `<video></video>`
- für Adaptive Streaming wird **Media Source Extensions (MSE)** benötigt
- erlaubt JavaScript das Senden von byte streams an das Video Element
- direkte Anzeige des MPEG-DASH Streams möglich

Implementierung



Stream empfangen

```
@Path("/video")
public class VideoResource {

    ...

    @Inject
    @Chunk
    private Event<ByteBuffer> chunkEvent;

    @POST
    @Consumes(MediaType.APPLICATION_OCTET_STREAM)
    public Response postChunk(InputStream chunk) {
        try {

            byte[] bytes = ByteStreams.toByteArray(chunk);
            ByteBuffer buf = ByteBuffer.wrap(bytes);

            Files.copy(chunk, Paths.get(PATH_STORE_VIDEOS + filename));

            chunkEvent.fire(buf);

        } catch (Exception e) {
            Log.error(e.getMessage());
        }
        return Response.ok().build();
    }
}
```

```
{
  "Local-Start-Time" : "2016-02-23T09:51:30.031",
  "Category" : "Konferenz",
  "Tags" : ["javaland","stream"],
  "Location" : {
    "Latitude" : 50.7984,
    "Longitude" : 6.8812,
    "Altitude" : 129m }
}
```

Stream verteilen

```
@ServerEndpoint(value = "/stream", encoders = { MessageEncoder.class }, decoders = { MessageDecoder.class })
public class StreamingEndpoint {

    private static final Set<Session> sessions = Collections.synchronizedSet(new HashSet<Session>());

    @OnOpen
    public void onOpen(Session session) {
        sessions.add(session);
    }

    public void onChunk(@Observes @Chunk ByteBuffer buffer) {
        for (Session session : sessions) {
            try {
                session.getBasicRemote().sendBinary(buffer);
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
    }

    @OnMessage
    public void onMessage(Message message, Session session) {
        for (Session s : sessions) {
            try {
                s.getBasicRemote().sendObject(message);
            } catch (IOException | EncodeException ex) {
                ex.printStackTrace();
            }
        }
    }
}
```

Videoausgabe vorbereiten

```
<video id="video" autoplay="true"/>
```

```
var video = document.querySelector('video');
```

```
var queue = [];  
var buffer;
```

```
var mediaSource = new MediaSource();  
mediaSource.addEventListener('sourceopen', function(e) {  
    video.play();  
  
    buffer = mediaSource.addSourceBuffer('video/mp4;codecs="avc1.64001f,mp4a.40.2"');  
    buffer.addEventListener('update', function() {  
        if (queue.length > 0 && !buffer.updating) {  
            buffer.appendBuffer(queue.shift());  
        }  
    });  
}, false);
```

```
video.src = window.URL.createObjectURL(mediaSource);
```

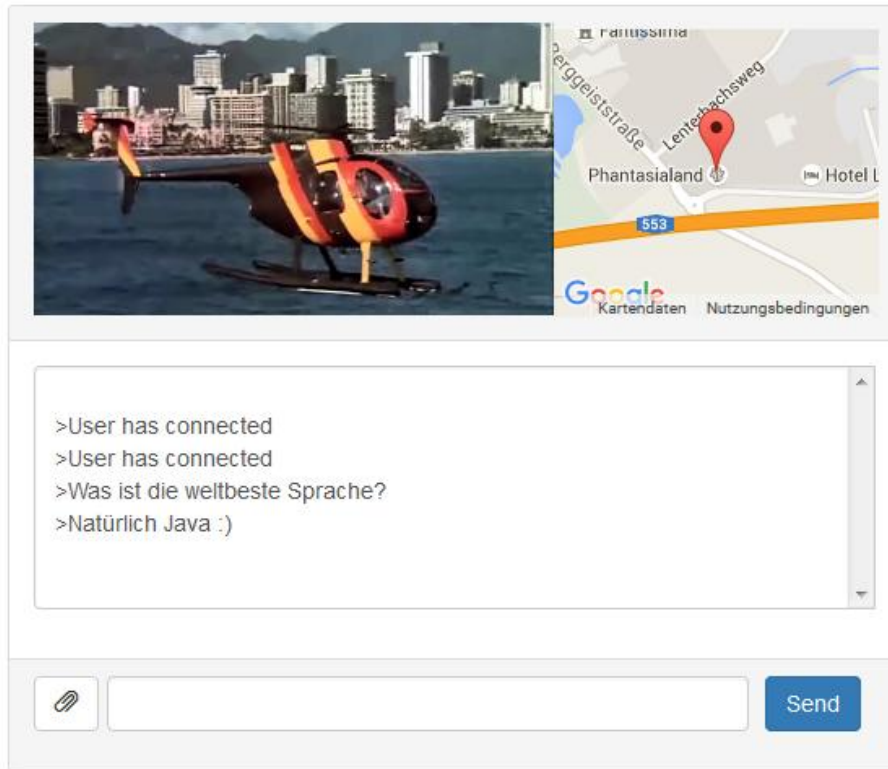
Video Stream anzeigen

```
var websocket = new WebSocket('ws://localhost:8080/streamup/stream');
websocket.binaryType = 'arraybuffer';

websocket.onmessage = function(event){
  if (event.data instanceof ArrayBuffer) {
    console.log('video chunk arrived');
    try {
      if (buffer.updating || queue.length > 0) {
        queue.push(event.data);
      } else {
        buffer.appendBuffer(event.data);
      }
    } catch (e) {
      console.log(e);
    }
  } else {
    writeResponse(event.data);
  }
};
```

Ergebnis

Prototyp



Video Live Streaming
=
MPEG-DASH
+
WebSockets
+
HTML5 Video Tag
+
Media Source Extensions
+
(REST)

Prototyp unter:

<http://JavaAkademie.de/javaland2016>

Guido.Oelmann@JavaAkademie.de

