23. Dezember 2016 Studie · Web & Cloud · 9 Min. Lesezeit

Flexible Kommunikation mit Cross-Platform und Web-Technologien

Archiviert. Dieser Inhalt wird nicht mehr aktualisiert. Er bleibt zur Information verfügbar.

Quelle https://technology2.a-sit.at/veroeffentlichungen/flexible-kommunikation-mit-cross-platform-und-web-technologien/

Obwohl Webtechnologien, wie sie in Webapplikationen und Cross-Platform Applikationen verwendet werden können, mittlerweile die notwendigen Werkzeuge bieten, um beliebige Applikationen zu entwickeln, fehlt es an direkten Kommunikationsmöglichkeiten zwischen Applikationen. In diesem Projekt wurden verschiedene Ansätze untersucht, um dieses Problem zu lösen. Für den Super-Node basierten Ansatz, der als am flexibelsten gilt, wurde eine Umsetzung entwickelt und wird hier zum Download bereitgestellt.

1. Einleitung

Ursprünglich dienten Webseiten rein der Anzeige von Informationen, Interaktionen waren nur sehr beschränkt möglich. Dies hat sich bereits seit einiger Zeit geändert, trotzdem blieben Webapplikationen einige Funktionalitäten verwehrt. Webapplikationen war es einzig und alleine erlaubt mit dem Server zu kommunizieren, über den die Webseite ausgeliefert wurde (mit einigen Ausnahmen, wie Bilder bzw. Content von anderen Servern). Außerdem musste eine Verbindung immer vom Browser initiiert werden. Server hatten keine Möglichkeit um Browser über geänderte Daten zu informieren. Erst seit der Einführung von HTML5 und den entsprechenden Javascript APIs, können Web Applikationen direkt mit anderen Webapplikationen kommunizieren, und können Zweiwegeverbindungen zu Servern aufbauen um über auftretende Ereignisse vom Server informiert zu werden ohne aktives Polling zu betreiben. Außerdem wurden Frameworks entwickelt die es ermöglichen mit denselben Technologien Applikationen für mobile Geräte zu entwickeln, die automatisch auf einer Vielzahl von unterschiedlichen mobilen Betriebssystemen lauffähig sind. Diese Basistechnologien werden in diesem Projekt herangezogen um ein Kommunikationsnetzwerk zu realisieren, das es jeder Applikation ermöglicht, mit anderen Instanzen direkt zu kommunizieren, ohne dass dabei eine zentrale Serverinstanz notwendig ist.

2. Technologien

Prinzipiell kommen für diese Art der Kommunikation zwei Basistechnologien neben Client-Server basierten HTTP-Verbindungen in Frage, die in diesem Kapitel besprochen werden.

2.1. WebRTC – Web-based Real Time Communication

Bei Web Real Time Communication (WebRTC) handelt es sich um eine Technologie, die eine direkte Client zu Client Verbindung ermöglicht. Dabei werden sowohl Webapplikationen, als auch native mobile Applikationen unterstützt. Eine Übersicht über WebRTC ist in Abbildung 1 ersichtlich.

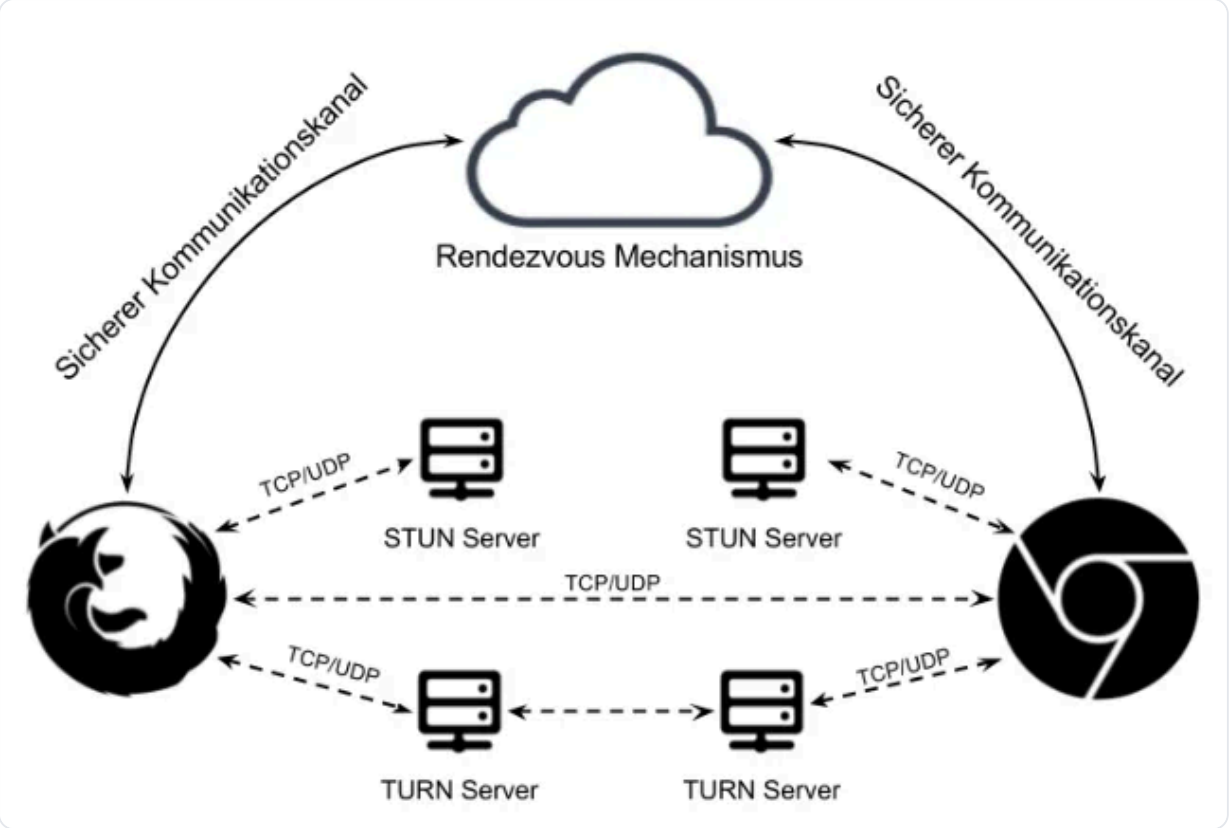


Abbildung 1: - WebRTC Übersicht Um eine Verbindung zwischen zwei Endpunkten mittels WebRTC aufzubauen, wird ein externer Rendezvousmechanismus benötigt der es ermöglicht, Verbindungsinformationen auszutauschen. Bei Webapplikationen wird dafür oft der Web-Server verwendet. Über diesen Rendezvousmechanismus versuchen die beiden Endgeräte einen Verbindungspfad zu finden. Beide tauschen dabei IP-Adress- Portpaare aus, zu denen versucht wird, gegenseitig zu verbinden. Sollten Geräte hinter Firewalls oder Netork Address Translation (NAT) Gateways stehen, so können Session Traversal Utilities for NAT (STUN) Server verwendet werden, um die tatsächliche öffentliche IP Adresse zu ermitteln, oder um Techniken wie NAT Hole

Punching anzuwenden. Kann kein Verbindungspfad gefunden werden, besteht als letzte Möglichkeit die Verbindung über externe *Relay Server* (TURN Server) herzustellen. Der benötigte externe Rendezvousmechanismus kann gleichzeitig als eine der Problemstellen von WebRTC betrachtet werden. Durch diesen Mechanismus ist es nicht möglich, auf den externen Server komplett zu verzichten, sondern zumindest zum Verbindungsaufbau ist er immer erforderlich.

2.2. WebSockets

Bei WebSockets handelt es sich gewissermaßen um eine Erweiterung von Standard HTTP Verbindungen. Dabei wird eine ständige Verbindung zum Server aufgebaut, wobei das Senden von Daten beidseitig initiiert werden kann, also vom Client, aber auch vom Server. Im Browser dürfen (ohne weitere Maßnahmen) von Javascript keine Anforderungen an andere Hosts als den Ursprungshost abgesetzt werden (sogenannte „Same Origin Policy“). Bei WebSockets trifft diese Einschränkung nicht zu und es können Verbindungen zu beliebigen Hosts aufgebaut werden. Nun ist es für Webapplikationen nicht möglich WebSocket Verbindungen anzunehmen, diese Möglichkeit besteht aber durchaus für mobile Applikationen die mit Cross-Platform Frameworks erstellt werden, oder anderen nativen Applikationen. Somit werden diese hier angeführt. WebSockets können also als Ergänzung zu WebRTC gesehen werden, keine der beiden Technologien kann für sich aber die gewünschte Funktionalität bereitstellen.

3. Umsetzungen zur Flexiblen Kommunikation

Ziel dieses Projekts ist es nun, dass jede Applikation mit jeder kommunizieren kann, ohne dass dafür ein zentraler Server benötigt wird, unter Verwendung der oben beschriebenen Technologien, da diese auf jedem Gerät und in der ein oder anderen Form für viele Umgebungen verfügbar sind.

3.1. Peer-to-Peer Netzwerke basierend auf WebRTC

Bei einem Peer-to-Peer Netzwerk handelt es sich prinzipiell um einen Verbund von Geräten, die miteinander kommunizieren können, ohne dass dabei zentrale Serverkomponenten notwendig sind. Es gibt dafür bereits einige existierende Modelle, die an dieser Stelle aber nicht diskutiert werden. Konkret wurde hier basierend auf der WebRTC Technologie versucht, ein Peer-to-Peer Netzwerk aus Webapplikationen aufzubauen. Hierzu gibt es bereits vorhandene proof-ofconcept Implementierungen, wie beispielsweise *WebRTC-Explorer* [1] . Hierbei wird ein Overlay Netzwerk, basierend auf WebRTC bereitgestellt, in dem jeder verbundene Knoten mit jedem kommunizieren kann. In diesem Ansatz können mehrere Probleme identifiziert werden, die jedoch mit den aktuell verfügbaren Standards nicht gelöst werden können:

```
- Um eine Verbindung zum Overlay-Netzwerk aufzubauen, ist ein zentraler Server
notwendig, um die initiale WebRTC Verbindung zum Einsprungspunkt aufzubauen. Zu
diesem Server müssen beide Endpunkte verbunden sein. Geht die Verbindung zum
Endpunkt verloren, ist zum erneuten Aufnehmen der Verbindung wieder eine dritte
Instanz notwendig, über die Verbindungsdaten ausgetauscht werden können.

- Ein Overlay-Netzwerk bestehend aus Webanwendungen und evtl. anderen mobilen
Applikationen kann sich als problematisch herausstellen. Einerseits sind Knoten
durch die Teilnahme am Overlay Netzwerk dazu verpflichtet, auch am Routing von
Paketen im Netzwerk teilzunehmen, andererseits können mobile Endgeräte auch über
nur sehr instabile Verbindungen verfügen. Instabile und langsame Verbindungen
können somit das gesamte Netzwerk beeinflussen.
```

1 https://github.com/diasdavid/webrtc-explorer Ein ähnliches System, allerdings mit anderen Zielen, wird mit *WebTorrent* [2] entwickelt. Hierbei handelt es sich um BitTorrent, das als Webapplikation mittels WebRTC implementiert wurde. BitTorrent verfügt bereits über eine Tracker-Struktur, die als zentraler

Anlaufpunkt zur Identifizierung von anderen Nodes verwendet wird. Für diesen Fall scheint die WebRTC Implementierung also keine Nachteile zu bringen. Auf den allgemeinen Fall, wie oben im WebRTC-Explorer Beispiel beschrieben, trifft dies nicht zu, da nach wie vor ein zentraler Server benötigt wird.

3.2. WebSocket basiertes Peer-to-Peer Netzwerk

Mittels WebSockets können die beschriebenen Probleme teilweise umgangen werden. WebSocket Server sind HTTP Webserver, deren Verbindung zu einem späteren Zeitpunkt zu einer WebSocket Verbindung hochgestuft wird. Da die Same Origin Policy für WebSockets nicht gilt, können Verbindungen von beliebigen Domains zu beliebigen Domains aufgebaut werden. Zum Verbindungsaufbau ist kein zusätzlicher Kanal notwendig, sondern kann direkt, ohne Einbeziehung Dritter erfolgen. Ein Problem ist, dass WebSockets in Webapplikationen nur als Client Sockets erstellt werden können, also aktiv keine Verbindungen von außen akzeptiert werden können. Dies würde die selben NAT-Umgehungsstrategien benötigen, wie WebRTC. WebSockets zielen aber eher auf Server basierte Anwendungsfälle ab.

3.3. Super-Node basiertes Peer-to-Peer Netzwerk

Um die oben beschriebenen Probleme zu umgehen bzw. zu vermeiden, wurde eine Lösung entwickelt die Eigenschaften aus beiden Ansätzen kombiniert:

```
- Applikationen auf Endgeräten sollen direkt miteinander kommunizieren, dabei soll
es unerheblich sein, ob es sich um eine Serverapplikation oder um
Clientapplikationen mit sehr eingeschränkten Berechtigungen handelt.

- Die benötigte Infrastruktur soll keinen einzelnen Schwachpunkt aufweisen, sondern
soll verteilt auf mehreren Rechnern laufen. Die angestrebte Lösung ist in Abbildung
2 skizziert. Es wird ein verteiltes Peer-to-Peer Netzwerk aus Super-Nodes
aufgesetzt, das zum Verwalten der Verbindungsinformationen dient. Der Begriff
_Super-Node_ soll darauf hinweisen, dass die Knoten über eine stabile
Internetanbindung verfügen und längerfristig online sind. Das System soll so
konzipiert sein, dass Ausfälle einzelner Super-Nodes keine Auswirkung auf den
Betrieb haben. Die eingesetzten Super-Nodes sind in einem Peer-to-Peer Netzwerk
organisiert mit folgenden Eigenschaften:

- Jeder Knoten im Netzwerk verfügt über einen eindeutigen Identifier.

- Jeder Knoten kann jedem anderen Knoten über seinen eindeutigen Identifier
Nachrichten senden.

- Das Peer-to-Peer Netzwerk fungiert zusätzlich als Distributed-Hash-Table, in der
Name-Wert Paare auf einem Knoten eingefügt werden können und dann auf einem
beliebigen anderen Knoten abgerufen werden können.
```

2 https://github.com/feross/webtorrent

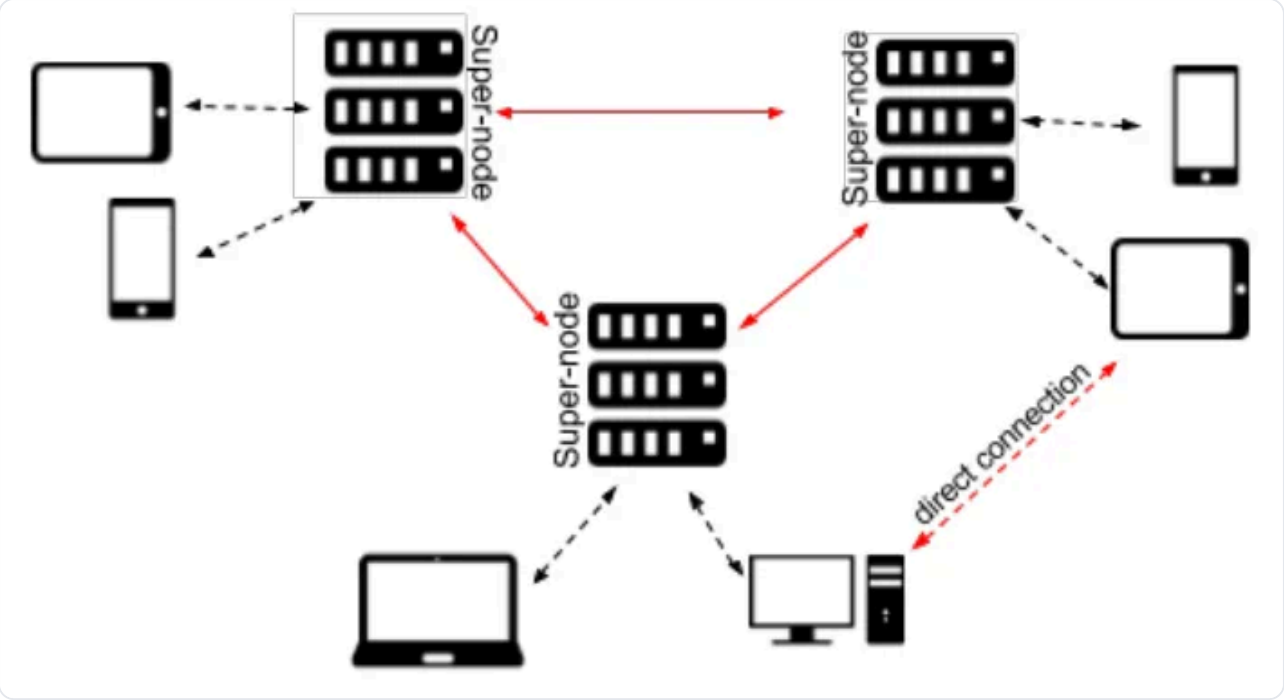


Abbildung 2: - Super-Node basiertes Peer-to-Peer Netzwerk Neben der Peer-to-Peer Funktionalität, stellen die Super-Nodes Zugriffspunkte für Endgeräte dar. Über diese Zugriffspunkte können sich Applikationen auf mobilen und stationären Endgeräten im Netzwerk registrieren. Im Zuge der Registrierung wird erfasst, wie der Applikationsendpunkt für andere Geräte erreichbar ist, nach dem Schema eines Verzeichnisdienstes. Folgende Endpunkttypen wurden berücksichtigt bzw. umgesetzt:

```
- Sollte eine Clientapplikation die Möglichkeit haben, einen WebSocket Endpunkt zu
öffnen, so wird die URL des Endpunktes im Netzwerk registriert.

- Für WebRTC Endpunkte, dient das Super-Node basierte Peer-to-Peer Netzwerk auch
als Rendezvousmechanismus um die notwendigen Verbindungsinformationen
auszutauschen. Ziel des Netzwerkes ist es nicht, dass sämtliche Kommunikation über
das Peer-to-Peer Netzwerk abläuft, sondern es nimmt nur eine unterstützende Rolle
ein um eine direkte Verbindung zwischen den Applikationen/Endpunkten aufzubauen.
Entweder in der Form, dass die entsprechende URL ausgetauscht wird, oder dass die
benötigten Verbindungsinformationen zum Herstellen einer WebRTC Verbindung
ausgetauscht werden können. Endergebnis soll immer sein, dass eine direkte
Verbindung besteht. Für zukünftige Erweiterungen kann angedacht werden, auch eine
Kommunikation über das Peer-to-Peer Netzwerk mit anderen Endpunkten zu ermöglichen,
sollte keine andere Verbindung möglich sein.
```

4. Implementierung

Für den in Kapitel 3.3 beschriebenen Ansatz wurde eine Umsetzung der zentralen Super-Node Komponente entwickelt. Sie basiert auf Java und verwendet TomP2P [3] als Basis Peerto-Peer Framework. Dieses stellt bereits die erwünschten Basisfunktionalitäten bereit, kann aber prinzipiell durch jede beliebige Implementierung ausgetauscht werden. Für die Verbindung der Endgeräte/Applikationen zu den Super-

Nodes werden WebSockets verwendet, um auch eine Verbindung von Webapplikationen zu ermöglichen. Über diese WebSockets können folgende Funktionalitäten unter Verwendung des JSON-RPC2 [4] Protokolls aufgerufen werden.

```
- _register:_ Ermöglicht es Endpunkte im Netzwerk zu registrieren. Dabei wird ein
eindeutiger Client-Identifier übergeben, oder generiert. Außerdem wird der
Verbindungsendpunkt übergeben. Im Peer-to-Peer Netzwerk werden dann neben den
übergebenen Daten auch der entsprechende Super-Node Identifier registriert.

- _connectionInfo:_ Ermöglicht es für einen gegebenen Client Identifier die
Verbindungsinformationen abzurufen, um eine direkte Verbindung aufzubauen.
```

3 https://tomp2p.net/ 4 http://www.jsonrpc.org/specification

```
- _discover:_ Ermöglicht es per Peer-to-Peer Broadcast, andere Endpunkte ausfindig
zu machen.

- Derzeit ist nur die Unterstützung für WebSocket Endpunkte implementiert. Die
Unterstützung von WebRTC Endpunkten ist aber vorgesehen durch _connectionData._
Diese Methode sieht vor, dass beliebige Daten zwischen zwei Endpunkten ausgetauscht
werden können, um beispielsweise das Verbindungsaufbauprotokoll von WebRTC
abzubilden.
```

Die bereitgestellte Umsetzung verfügt über eine Vielzahl von Kommandozeilen Parametern die wie folgt genutzt werden können. Der erste Knoten, der gestartet wird, muss mit `--createNew` initialisiert werden. Hiermit wird ein neues Peer-to-Peer Netzwerk initialisiert. Damit andere Super-Nodes diesem Netzwerk beitreten können, muss ihnen beim Start mittels `--bootstrapfile` eine Liste mit anderen Knoten des Peer-to-Peer Netzwerkes übergeben werden. Im Testdeployment wurde diese Auflistung über einen Webserver für alle anderen Nodes zugänglich gemacht. Derzeit wird keine automatische Erkennung der öffentlichen IP Adresse unterstützt. Diese kann mittels `--overridePublicIp` manuell gesetzt werden. Für die verwendeten Ports, einerseits für die lokalen Peer-to-Peer Netzwerk Endpunkte, und andererseits zur Bereitstellung der WebSocket Konnektivität, können die Parameter `--p2pport` und `-websocketport` verwendet werden.

5. Offene Punkte

Im Zuge dieses Projekts wurde eine Basisumsetzung durchgeführt. Die folgenden Punkte könnten beispielsweise in Folgeprojekten aufgegriffen werden:

- Die Komponenten können mit überschaubarem Aufwand erweitert werden, um auch die Verbindungserstellung für WebRTC zu unterstützen.

- Sollte keine direkte Verbindung (unabhängig von der verwendeten Technologie) möglich sein, so sollte die Möglichkeit bestehen das Super-Node Netzwerk zur Kommunikation zu verwenden.

- Derzeit wird weder für angeschlossene Geräte/Anwendungen noch für Knoten des Super-Node Netzwerkes die automatische Erkennung der öffentlichen IP-Adresse unterstützt. Für Knoten im Super-Node Netzwerk ist dies bereits vom verwendeten Peer-to-Peer Netzwerk Layer vorgesehen. Für angeschlossene Geräte/Anwendungen kann diese Erkennung von den einzelnen Knoten durchgeführt werden.

- Das Protokoll, mit dem Geräte und Anwendungen direkt miteinander kommunizieren, ist prinzipiell unabhängig von den hier beschriebenen Mechanismen. Trotzdem wird es in vielen Anwendungsfällen notwendig sein, eine gemeinsame Vertrauensbasis mit der Gegenstelle zu etablieren. Hierfür könnte ebenso das Super-Node basierte Netzwerk verwendet werden. Zusätzlich zu den Verbindungsinformationen können beispielsweise Zertifikatsdaten gespeichert werden, über die dann eine Authentifizierung des jeweiligen Knotens erfolgt.