12.2 Funktionsweise des P2P-Overlay-Netzes

Overlay-Netz als virtueller Server

Ein Overlay-Ringnetz ermöglicht die Kommunikation zwischen jeweils zwei Peers durch die Übermittlung der hierfür notwendigen Signalisierung (SIG) zum Aufbau einer virtuellen Verbindung zwischen zwei IP-Telefonen. Abbildung 12.2-1a veranschaulicht dies am Beispiel der Verbindung zwischen den IP-Telefonen X und Y. Bei der Übermittlung der Signalisierung zwischen diesen IP-Telefonen sind nur die Peers 3 und 6 als Knoten auf dem logischen Ringnetz beteiligt. Diese Peers können an verschiedenen Stellen im Internet installiert werden. Anhand des hier gezeigten Beispiels soll insbesondere zum Ausdruck gebracht werden, dass die Übermittlung der Signalisierung dank einer sog. Finger-Tabelle (siehe Abb. 12.2-4) nur zwischen einigen Knoten des logischen Overlay-Ringnetzes verläuft. Die Übermittlung von Sprache – als Benutzer-zu-Benutzer-Kommunikation – verläuft „außerhalb“ des Overlay-Ringnetzes.

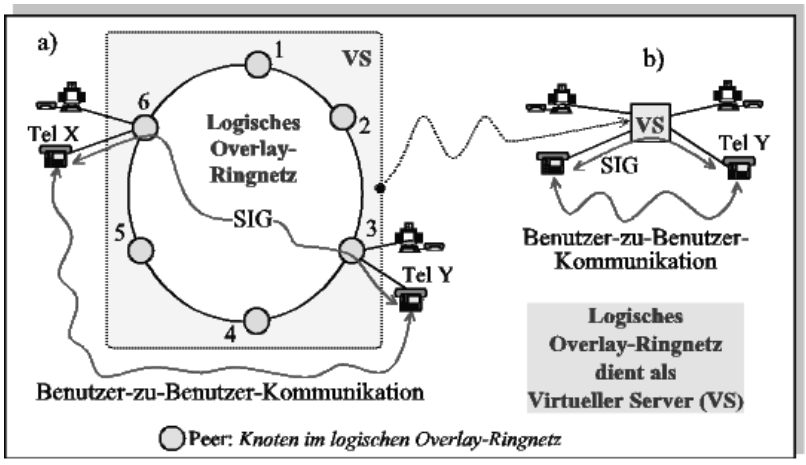


Abb. 12.2-1: Overlay-Ringnetz als virtueller Server für VoIP:
a) Logisches Overlay-Netzwerk als Ringnetz, b) Virtueller Server
SIG: Signalisierung, Tel: (IP-)Telefon

Overlay-Ringnetz als virtueller Server

Vergleicht man die traditionelle Client-Server-Architektur (hierfür siehe Abb. 12.1-1) mit der in Abbildung 12.2-1a gezeigten logischen Vernetzungsstruktur, so könnte das ganze Overlay-Ringnetz – der Funktion nach – quasi als verteilter Server angesehen werden. Demzufolge ist das Overlay-Ringnetz als *virtueller (scheinbarer) Server* zu verstehen. Abbildung 12.2-1b illustriert diese Sichtweise.

12.2.1 P2P-Overlay-Netz als Ringnetz

Overlay-Ringnetz als P2P-Chord

Im Weiteren wird nur auf Overlay-Netze mit einer Ringstruktur eingegangen. Diese Struktur ist sehr flexibel und alles deutet darauf hin, dass sie in allen zukünftigen Systemen eingesetzt werden wird. Die P2P-Netzarchitektur mit einem Overlay-Ringnetz ist dezentralisiert und so strukturiert, dass sie sich sehr gut zur Unterstützung der P2P-Kommunikation eignet. Die P2P-Netzarchitektur mit einem Overlay-Ringnetz wird oft als *P2P-Chord* bezeichnet. Abbildung 12.2-2 erläutert ihre Organisation.

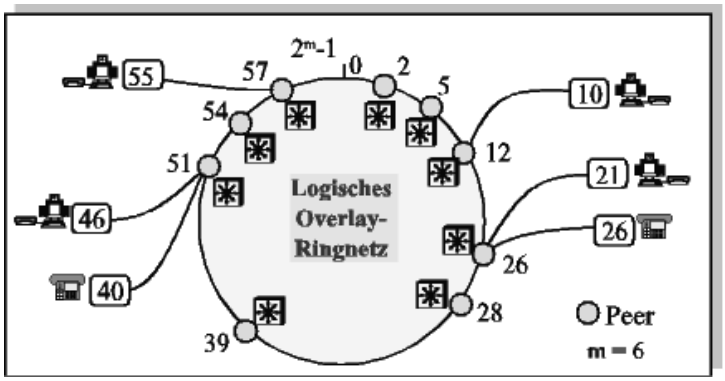


Abb. 12.2-2: Prinzip des Aufbaus eines Overlay-Ringnetzes mit maximal 2^m-1 (m = 5) Knoten

ID als Wert einer Hash-Funktion

Jeder Rechner – als Peer bzw. Client – muss über eine eindeutige Identifikation (ID) verfügen. Diese Identifikation wird als Wert einer Hash-Funktion F berechnet. *Als Argument dieser Funktion kann die IP-Adresse des Rechners genommen werden*. Die Identifikation (ID) wird daher wie folgt bestimmt:

ID = F(IP-Adresse)

Bei P2PP können u.a. folgende Hash-Funktionen genutzt werden:

- *Message Digest (MD):* MD4 oder MD5
- *Secure Hash Algorithm (SHA)*: SHA-1, SHA-256 oder SHA-512

Bemerkung: Jede Hash-Funktion hat folgende zwei Eigenschaften, die bei der P2P-Kommunikation von fundamentaler Bedeutung sind:

- *Sie ist kollisionsfrei*. Dies bedeutet, dass es keine zwei Argumente gibt, die zu dem gleichen Wert der Hash-Funktion führen. Nutzt man die IP-Adressen von Peers bzw. von Clients als Argumente einer Hash-Funktion, um ihre Identifikationen (IDs) zu bestimmen, kann man somit sicher sein, dass diese Identifikationen eindeutig sind.
- *Binäre Werte einer Hash-Funktion haben immer die gleiche Länge.* Sind die Argumente der Hash-Funktion – als binäre Zahlen – unterschiedlicher Länge, haben die binären Werte der Hash-Funktion aus diesen Argumenten immer die gleiche Länge – z.B. m. Daher können mit einem Overlay-Ring maximal 2^m-1 Peers logisch vernetzt werden.

Uhrmodell der Overlay-Ringnetze

Die Peers auf dem Overlay-Ring werden – *so wie Stunden auf dem Ziffernblatt einer Uhr* – fortlaufend nummeriert. Als Nummer jedes Peers auf dem Overlay-Ring wird seine ID (Identifikation) angenommen. Daher kann hier von einem *Uhrmodell* der Overlay-Ringnetze gesprochen werden. Da der Wert einer Hash-Funktion – als binäre Zahl – eine feste Länge hat, ist die Anzahl von Peers auf dem Ring begrenzt. Der Overlay-Ring in Abbildung 12.2-2 kann maximal 2^m-1 (`m = 6`) Peers als Knoten beinhalten.

Begriff: Successor eines Clients

Ein Client kann sich an ein Overlay-Netz – z.B. im in Abbildung 12.2-2 gezeigten Beispiel an einen Overlay-Ring – nach bestimmten Regeln an einen Peer anschließen. Dieser Peer wird als sein *Successor* bezeichnet. Als Successor des Clients mit `ID = k` dient der erste Peer auf dem Ring, dessen ID gleich nach `k` folgt bzw. gleich `k` ist. Wie Abbildung 12.2-2 zeigt, werden beispielsweise der Client mit `ID = 10` an den Peer mit `ID = 12` und die Clients mit `ID = 21` und `ID = 26` an den Peer mit `ID = 26` angeschlossen. Benutzt man den Begriff Successor, können die Anschluss-Peers von Clients mit IDs 10, 21 und 26 wie folgt spezifiziert werden:

Successor (10) = 12, Successor (21) = 26, Successor (26) = 26

Ermittlung: Woran ist der Client angeschlossen?

Das Prinzip, nach dem der Successor eines Clients bestimmt wird, ermöglicht eine eindeutige Ermittlung, an welchem Peer ein Client angebunden ist, vorausgesetzt, man kennt seine ID. Da diese ID aus der IP-Adresse des Clients durch eine Hash-Operation entsteht, ermöglicht dieses Prinzip eine eindeutige Ermittlung des Peers für den Anschluss des Clients, falls nur seine IP-Adresse bekannt ist. *Diese Tatsache ist beim P2P-Networking von zentraler Bedeutung.*

Peer muss seinen Successor kennen.

Ein Overlay-Ringnetz funktioniert nämlich nur dann korrekt, wenn jeder Peer auf dem Ring seinen Nachfolger – will heißen seinen Successor – kennt. Abbildung 12.2-3 bringt diese Voraussetzung für eine korrekte Funktionsweise eines Overlay-Ringnetzes näher zum Ausdruck.

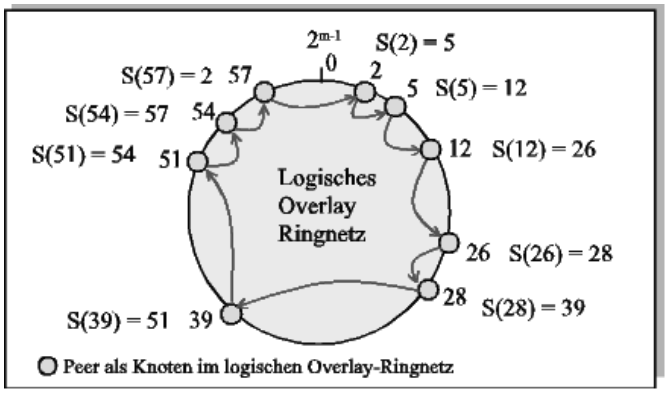


Abb. 12.2-3: In einem Overlay-Ring muss jeder Peer seinen Successor kennen.
S: Successor

Redundante Wege im Overlay-Ring

Dadurch, dass jeder Peer seinen Successor kennt, entsteht quasi eine geschlossene logische Schleife – das heißt ein Ring. Dies birgt jedoch ein Risiko in sich. Fällt ein Peer plötzlich aus, so gibt es auch keinen Ring mehr. Umgehen lässt sich dieses Problem mittels der Einführung *redundanter Wege im Overlay-Ringnetz*. Die Einführung und Erfassung der redundanten Wege erfolgen mit Hilfe sog. *Finger-Tabellen* und diese können als Kern des P2P-Networking auf der Basis von Overlay-Ringnetzen verstanden werden.

Einsatz von Finger-Tabellen

Zwecks der Möglichkeit einer Einführung redundanter Wege im Overlay-Ringnetz enthält jeder Peer also eine sog. *Finger-Tabelle*. Jedem Peer werden mehrere *Finger* zugeordnet. Abbildung 12.2-4 zeigt eine Finger-Tabelle und veranschaulicht dabei ihre Bedeutung. Wie hier zum Ausdruck gebracht wurde, *dient der Finger eines Peers als Verweis auf seinen Successor*. Dadurch können einem Peer mehrere Successors zugeordnet werden.

12.2.2 Bedeutung von Finger-Tabellen

Die Tatsache, dass ein Peer auf dem logischen Overlay-Ringnetz dank seiner Finger-Tabelle mehrere Successors hat, ist in P2P-Netzen von enormer Bedeutung. Aber die Finger-Tabelle hat noch einen weiteren positiven Aspekt, den man beim Routing im Overlay-Netz nutzt. Auf diesen wird in Abschnitt 12.2.4 (siehe Abb. 12.2-6) eingegangen.

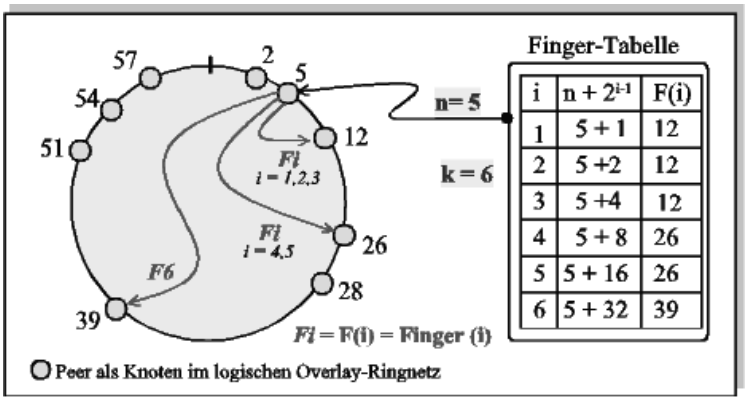


Abb. 12.2-4: Aufbau einer Finger-Tabelle und ihre Bedeutung

Im Ring mit 2^k-1 Peers können dem Peer mit ID = n mittels seiner Finger (F) folgende mögliche Successors auf dem Overlay-Ringnetz zugeordnet werden:

`Finger (i) = Successor(n+2`$^{i-1}$`)`, wobei `i = 1, 2, …, k`

Fällt z.B. der Peer mit `ID = 12` aus, so kann der Peer mit `ID = 5` den Peer mit `ID = 26` als seinen Successor nutzen. Folglich hat der Ausfall eines Peers

keine Auswirkung auf die weitere Funktionsweise des Overlay-Ringnetzes. Der Parameter `k` bestimmt die maximale Anzahl möglicher Successors auf dem Overlay-Ringnetz.

DHT als Beschreibung des Overlay-Ringnetzes

Die Vernetzungsstruktur innerhalb des Overlay-Ringnetzes und die Reihenfolge der Anbindung von Clients auf diesem können durch ihre, als Werte einer ausgewählten Hash-Funktion berechneten, Identifikationen in Form einer Tabelle beschrieben werden. Da die Inhalte dieser Tabelle verteilt und ihre Teile in einzelnen Peers und Clients abgespeichert sind, wird sie *Distributed Hash Table* (DHT) genannt. Die Abkürzung DHT ist bei P2P-Networking bereits üblich.

12.2.3 Beitritt eines Peers zum Overlay-Ringnetz

Das Overlay-Ringnetz kann als eine „offene Gruppe" von als Peers bezeichneten Rechnern betrachtet werden. Daher kann jederzeit ein neuer Peer hinzukommen. Man spricht in diesem Zusammenhang von einer *Join-* bzw. *Beitrittsphase.* Abbildung 12.2-5 illustriert ihren Verlauf.

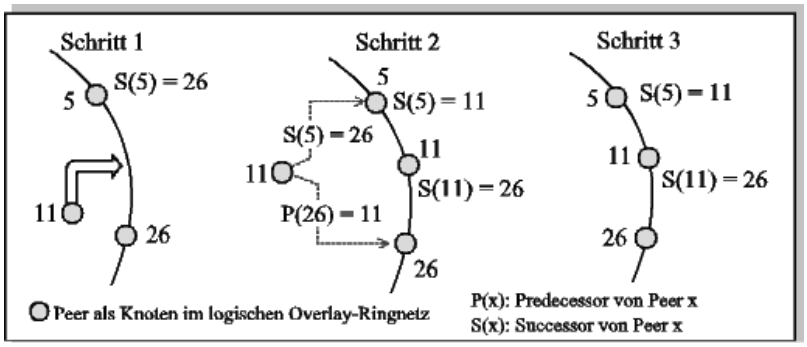


Abb. 12.2-5: Beispiel für den Verlauf einer Beitrittsphase

Während der Join-Phase sind folgende Schritte zu unterscheiden:

1. *Ermittlung des Successors*

   Ein Rechner – hier mit `ID = 11` – möchte sich als Peer einem Overlay-Ringnetz anschließen. Hierfür muss er seinen Successor finden. Der Successor des Peers mit `ID = 11` ist der Peer mit `ID = 26`. Für die Ermittlung des Successors sind spezielle Nachrichten erforderlich. Diese werden beispielsweise im Protokoll `RELOAD` (siehe RFC 6940) definiert.

2. *Benachrichtigung des Vorgängers auf dem Overlay-Ringnetz*

   Der neue Peer mit `ID = 11` muss dem Peer mit `ID = 5` – d.h. seinem Vorgänger (Predecessor) auf dem Overlay-Ringnetz – mitteilen, dass er

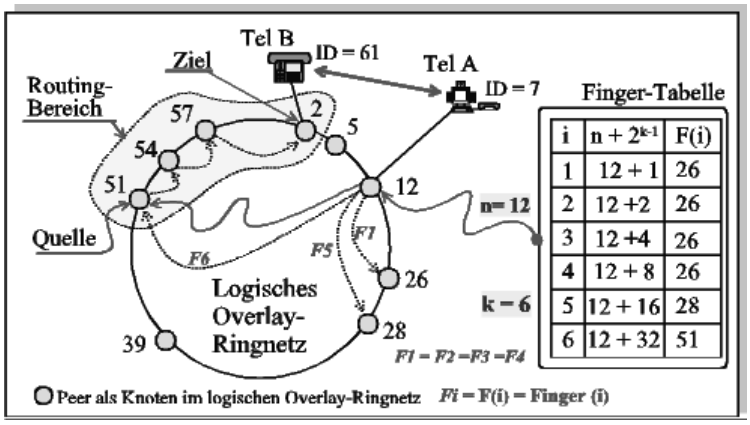
nun einen anderen Vorgänger (Successor) hat, denn der neue Peer ist jetzt sein Vorgänger.

3. *Neue Konfiguration auf dem Overlay-Ringnetz*

   Nachdem der neue Peer bei sich seinen Successor eingetragen und dem Vorgänger auf dem Ring mitgeteilt hat, dass dieser ab jetzt als sein Successor fungiert, ist eine neue stabile Konfiguration auf dem Overlay-Ringnetz gegeben.

12.2.4 Routing im Overlay-Ringnetz

Ein Overlay-Ringnetz enthält einerseits Knoten als Peers, d.h., jeder Peer kann mit jedem anderen Peer im gleichen Overlay-Ringnetz direkt kommunizieren. Andererseits kann das Overlay-Ringnetz der Funktion nach als verteilter – also als eine Art virtueller – Kommunikationsserver (siehe Abb. 12.2-1) verstanden werden. Dies wurde in Abbildung 12.2-1 verdeutlicht. Jeder Peer dient allen an ihn angeschlossenen Clients sozusagen als lokaler Kommunikationsserver. Abbildung 12.2-6 bringt dies zum Ausdruck.



i	$n + 2^{k-1}$	F(i)
1	12 + 1	26
2	12 + 2	26
3	12 + 4	26
4	12 + 8	26
5	12 + 16	28
6	12 + 32	51

Abb. 12.2-6: Prinzip der Realisierung von Routing in einem Overlay-Ringnetz

Zur Ermöglichung der Kommunikation zwischen zwei an verschiedenen Peers auf dem Overlay-Ringnetz angebundenen Clients müssen diese Peers entsprechende Informationen austauschen – also direkt miteinander kommunizieren. Dies führt jedoch zu einem Routing-Problem, welches im Folgenden mit Hilfe der Abbildung 12.2-6 erläutert wird.

In dieser Abbildung soll beispielsweise mit Hilfe des Signalisierungsprotokolls SIP eine Verbindung zwischen den IP-Telefonen A und B aufgebaut werden. Die Telefone sind als Clients an die Peers 12 und 2 angebunden. Die Verbin-

dung wird hier vom Telefon A mit `ID = 7` initiiert. Das Telefon A übermittelt an den Quell-Peer mit `ID = 12` die SIP-Nachricht `INVITE` mit der Angabe von SIP-URI des Telefons B. Aus SIP-URI ermittelt der Quell-Peer zuerst – mit Hilfe von DNS – die IP-Adresse von Telefon B. Dann berechnet er anhand der angewandten Hash-Funktion die Identifikation `ID = 61` des Telefons B.[2]

Routing-Notwendigkeit

Der Quell-Peer kennt nun den Ziel-Client. Genauer gesagt weiß er nur, dass dessen `ID = 61` ist und muss den Ziel-Peer ermitteln, an den der Ziel-Client (Tel B) angebunden ist und über den die Verbindung zum Ziel-Client aufgebaut werden kann. Um den Ziel-Peer – d.h. den Peer, an den der Ziel-Client angebunden ist – ermitteln zu können, muss ein Routing-Vorgang durchgeführt werden.

Quell-Peers als lokale Kommunikationsserver

Der Quell-Peer stellt aufgrund der ID des Ziel-Clients und seiner Finger-Tabelle fest, dass der Ziel-Peer ein Peer mit einer ID zwischen 51 und 5 sein muss (siehe Abb. 12.2-3). Daher ist der Ziel-Peer in diesem ID-Bereich zu ermitteln. Dies geschieht mit Hilfe eines Routing-Prozesses. Sobald der Ziel-Peer mit `ID = 2`, an den der Ziel-Client angeschlossen ist, ermittelt wurde, kann die Verbindung zwischen den Clients – hier den zwei IP-Telefonen – aufgebaut werden. Abbildung 12.2-7 illustriert diesen Vorgang näher.

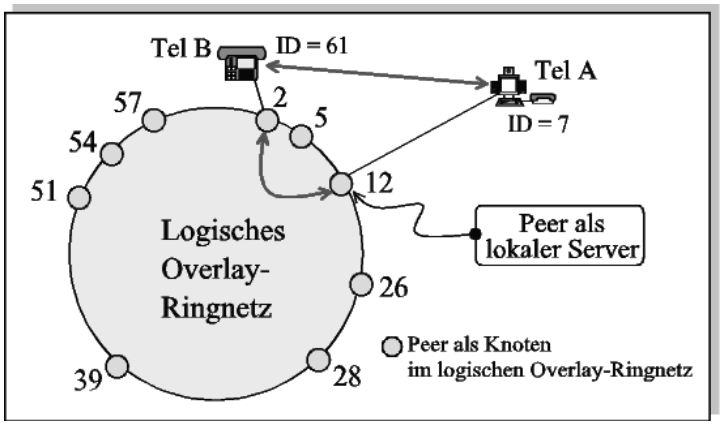


Abb. 12.2-7: Peers im Overlay-Ringnetz als lokale Kommunikationsserver

Aus dem hier dargestellten Beispiel geht hervor, dass das logische Overlay-Ringnetz als ein verteiltes System von Peers, einer Art Kommunikationsservern, dient. Mit zwei Peers wird die Kommunikation zwischen zwei Clients ermöglicht. Der Quell-Peer, der die Kommunikation initiiert, ist bekannt. Der

[2] Hierbei wurde die Nutzung von Hashwerten als Identifikationen stillschweigend vorausgesetzt. Diese Identifikationen von Peers und Clients werden aus ihren IP-Adressen als Hashwerte ermittelt.

Ziel-Peer, an den der andere Client angeschlossen ist, muss bestimmt werden, was zur Realisierung von *Routing im Overlay-Ringnetz* führt.

Routing-Arten

Im Allgemeinen ist zwischen den folgenden Arten von Routing zu unterscheiden:

- *rekursives Routing* und
- *iteratives Routing.*

Abbildung 12.2-8 zeigt die Unterschiede zwischen diesen Arten von Routing. Es sei angemerkt, dass das hier betrachtete Routing in einem *Overlay-Ringnetz* bei P2P-Kommunikation nur zur Ermittlung des Ziel-Peers führt.

Rekursives Routing

Wie aus der Abbildung 12.2-8a hervorgeht, leitet beim rekursiven Routing jeder Peer unterwegs zum Ziel-Peer die Nachricht des Quell-Peers – als *Request* (Req) – weiter. Hat dieser Request den Ziel-Peer erreicht, so sendet er eine Antwort – d.h. *Response* (Rsp) – an den Quell-Peer zurück. Diese wird an den Quell-Peer über die gleichen „Zwischen-Peers“ übermittelt.

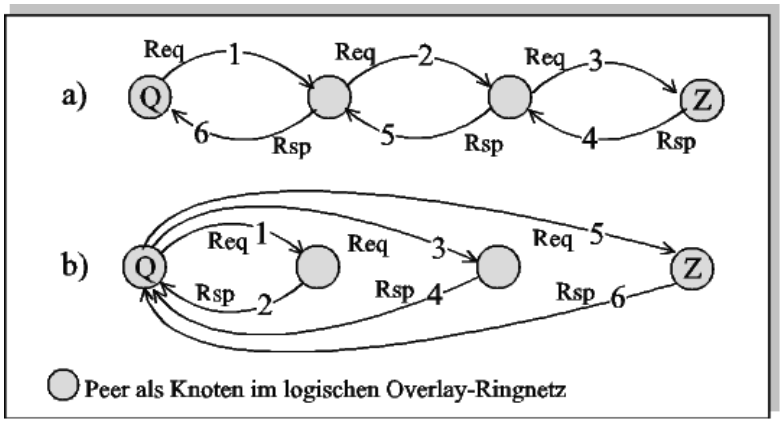


Abb. 12.2-8: Routing-Arten bei P2PP: a) rekursives Routing, b) iteratives Routing
Q: Quell-Peer, Z: Ziel-Peer, Req: Request, Rsp: Response

Iteratives Routing

Wie die Abbildung 12.2-8b zeigt, gibt ein Peer beim iterativen Routing in der Response, die er an den Quell-Peer auf dessen Request zurücksendet, seinen Successor an. Im nächsten Schritt sendet der Quell-Peer einen Request (Req) an diesen Successor – an den nächsten Peer – und erhält von diesem eine Response (Rsp).

Die in Abbildung 12.2-8 gezeigten Routing-Arten entsprechen weitgehend den Betriebsarten von SIP-Servern. Das heißt:

- das rekursive Routing entspricht dem Einsatz von *Proxy-Servern* (siehe Abschnitt 7.1.4) und

- das iterative Routing entspricht dem Einsatz von *Redirect-Servern* (siehe Abschnitt 7.2.2).

12.3 Ziele und Bedeutung des P2PSIP

Eine Auflistung der wichtigsten mittels des Protokolls P2PSIP (*Peer-to-Peer SIP*) umzusetzenden Ziele zeigt Abbildung 12.3-1.

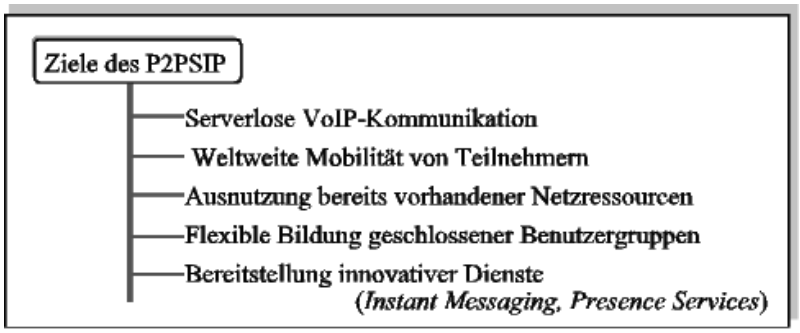


Abb. 12.3-1: Wichtigste Ziele des P2PSIP

Serverlose VoIP-Kommunikation

Ein wesentliches Ziel bei der Entwicklung von P2PSIP war die Schaffung einer Netzarchitektur, mit deren Hilfe eine Gestaltung der Kommunikation zwischen allen Rechnern ohne den Einsatz von Servern – wie z.B. SIP-Proxies – möglich ist. Bei der serverlosen VoIP-Kommunikation mit P2PSIP entfällt also sowohl die aufwendige Konfiguration als auch die Verwaltung verschiedener Server. In diesem Zusammenhang ist insbesondere die Bildung serverloser IP-TK-Anlagen – auch IP-PBX (*Private Branch eXchange)* genannt – hervorzuheben.

Weltweite Mobilität von Nutzern

Ein anderes wichtiges Ziel bei der Entwicklung von P2PSIP war die Gewährleistung einer weltweiten Benutzermobilität. Dies wird bei P2PSIP dadurch erreicht, dass das logische P2P-Overlay-Ringnetz ein selbstorganisierendes Netz ist – also ein sog. *Ad-hoc-Netz*, in dem jeder Rechner mit jedem anderen spontan kommunizieren kann. P2PSIP ermöglicht so die VoIP-Kommunikation in mobilen Ad-hoc-Netzen. Als Beispiel für ein mobiles Ad-hoc-Netz kann eine Vernetzung von Autos – also ein *Car-to-Car-Network* – dienen.

Ausnutzung vorhandener Netzressourcen

Für die Bildung eines P2P-Overlay-Ringnetzes zur Unterstützung der VoIP-Kommunikation – und somit auch zur Realisierung anderer Formen der Echtzeitkommunikation über IP-Netze – können bereits vorhandene Netzressourcen weitgehend weiterverwendet werden. Die Schaffung dieser Möglichkeit war ein wichtiges Anliegen bei der Konzeption von P2PSIP.

14 WebRTC – Konzept und Einsatz

Multimediale Echtzeit-kommuni-kation mit WebRTC

WebRTC (*Web Real-Time Communication*) ist eine richtungsweisende Idee, die eine Realisierung multimedialer Echtzeitkommunikation mit Hilfe von Webbrowsern – also eine *Web-Echtzeitkommunikation* – ermöglicht, für die keine besonders komplexen zusätzlichen Software-Module installiert werden müssen. Zur Unterstützung von WebRTC seitens eines Webbrowsers können bei Bedarf von einem Webserver verschiedene RTC-spezifische Funktionsmodule heruntergeladen und in den Webbrowser direkt, quasi automatisch, „eingebaut" werden. Welche Module hierfür nötig sind, wird später näher erläutert.

WebRTC als Extended VoIP over Web

Betrachtet man WebRTC rein vom technischen Standpunkt her, so stellt man fest, dass weitgehend das Konzept von VoIP übernommen und um zusätzliche Funktionen erweitert wurde. Daher könnte man einige Systemlösungen für WebRTC auch als *Extended VoIP over Web* bezeichnen. Dies wird im Weiteren zum Ausdruck gebracht und dabei gezeigt, dass WebRTC auch als eine besondere Weiterentwicklung von VoIP angesehen werden kann. Demzufolge wird die Integration von WebRTC in bestehende VoIP-Systeme, vor allem im Hinblick auf eine breite Verwendung VoIP-fähiger Smartphones und deren Einsatz zur Nutzung webbasierter Internet-Dienste, von großer Bedeutung sein.

Überblick über das Kapitel

Das Ziel dieses Kapitels ist es, eine Übersicht über die allgemeinen Ideen, Konzepte und Anwendungsmöglichkeiten von WebRTC zu geben. Nach der Darstellung funktionaler Komponenten in Abschnitt 14.1 wird in Abschnitt 14.2 ein Kommunikationsmodell präsentiert. Welche Schritte vor und nach der WebRTC-Nutzung nötig sind und wie Sessions zwischen Clients verlaufen, zeigen die Abschnitte 14.3 und 14.4. Die Bedeutung von ENUM ((*tElephone Number URI Mapping*) zeigt Abschnitt 14.5. Der Einsatz von SIP und die Kopplung von WebRTC mit VoIP-Systemen wird in den Abschnitten 14.6 und 14.7 präsentiert. Auf Sicherheitsaspekte und die Standardisierung von WebRTC gehen die Abschnitte 14.8 und 14.9 ein.

Ziel dieses Kapitels

Dieses Kapitel geht u.a. auf folgende Fragestellungen ein:

- Welche Kommunikationsmöglichkeiten sind bei WebRTC denkbar?
- Welche Ideen liegen WebRTC-basierten Homeoffices zugrunde?
- Wie können Kommunikationssysteme mit WebRTC aufgebaut werden?
- Welche Schritte sind beim Verlauf von WebRTC-Anwendungen nötig?
- Welche Bedeutung hat das VoIP-Protokoll SIP bei WebRTC?
- Wie kann WebRTC in VoIP-Systeme integriert werden?
- Welche Sicherheitsprobleme sind bei WebRTC relevant?

14.1 Funktionale Komponenten von WebRTC

WebRTC als Basis für Homeoffices

Mit WebRTC wird eine Idee zur Integration multimedialer Kommunikation, insbesondere von VoIP, mit Webanwendungen realisiert. Diese Integration kann als die wichtigste Besonderheit von WebRTC angesehen werden. Die WebRTC-Idee liegt allen Systemlösungen zugrunde, die als technische Basis für die Einrichtung von Homeoffices anzusehen sind. Von welcher, fast existenzieller, Bedeutung Homeoffices sein können, hat sich während der COVID-19-Pandemie gezeigt.

14.1.1 Webbrowser mit WebRTC-Unterstützung

RTC-fähiger Webbrowser

Bei WebRTC kann ein Webbrowser – auch kurz *Browser* genannt – nicht nur als Software-Telefon (Softphone), sondern auch als multimediale Kommunikationsinstanz zur Unterstützung von Sprach-, Video- und Datenkommunikation zwischen über das Internet kommunizierenden Personen dienen – d.h. als eine Art *Web-Videotelefon* (Web Video Phone) genutzt werden. Damit ein Webbrowser diese Funktionen bereitstellen kann, muss er um eine RTC-spezifische Software erweitert werden, also RTC-fähig sein. Abbildung 14.1-1 illustriert dies näher und zeigt, welche Komponenten nötig sind, um webspezifische Internet-Anwendungen mit *Multimedia over IP* (MoIP) integrieren zu können.

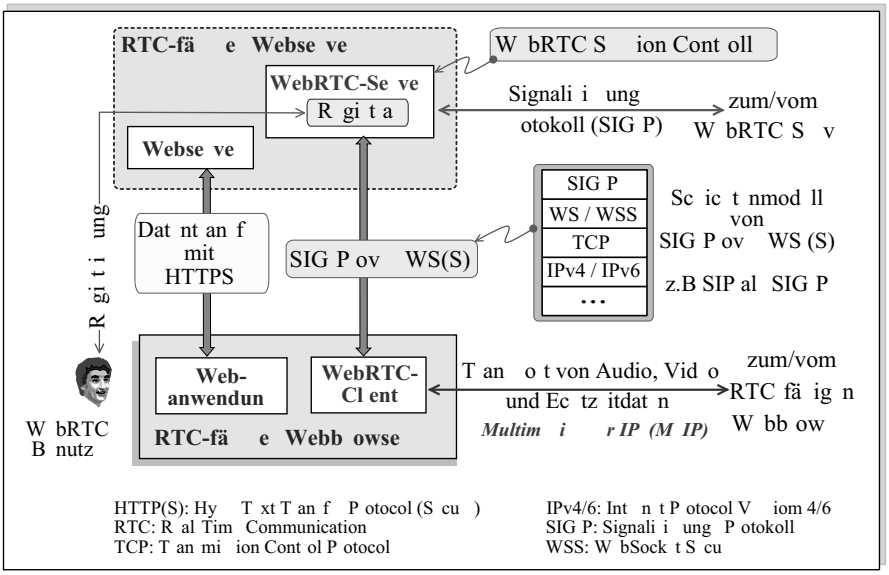


Abb. 14.1-1: Vereinfachtes Modell der Kommunikation zwischen WebRTC-Client und -Server

Sollte der Webbrowser in einem Rechner noch nicht RTC-fähig sein, so wird der Internetnutzer in die Lage versetzt, unter Verwendung eines Links auf dem Webbrowser zu veranlassen, dass eine RTC-spezifische Software mit Hilfe des Protokolls HTTPS (*HyperText Transfer Protocol Secure*) von einem Webserver heruntergeladen und in den Webbrowser automatisch integriert wird. Auf diese einfache Weise kann der Webbrowser RTC-fähig gemacht werden und als eine Art webfähiges Videotelefon – kurz Web-Videotelefon – dienen.

WebRTC-Server als Vermittlungsknoten

In allen privaten Systemen, welche Telefonie ermöglichen, werden für die Dauer der Kommunikation (des Telefonats) Telefonapparate als Endgeräte miteinander unter Mitwirkung zentraler Vermittlungsknoten verbunden. Als Vermittlungsknoten dient bei VoIP mit dem Protokoll SIP (*Session Initiation Protocol*) in einer Organisation (Unternehmen, Institution ...) ein spezieller SIP-Proxy-Server (s. Abb.7.2-1). Bei WebRTC wird ein funktionell ähnlicher Server, im Weiteren *WebRTC-Server* genannt, benötigt, der als Vermittlungsknoten zwischen Web-Videotelefonen fungiert. Eine RTC-spezifische, die Funktion eines Web-Videotelefons erbringende Software-Instanz im Webbrowser kann somit als Client des WebRTC-Servers angesehen werden. Aus diesem Grund wird diese in den Webbrowser integrierte Software-Instanz hier als *WebRTC-Client* bezeichnet.

Idee von WebRTC

Die grundlegende Idee von WebRTC (s. auch die Abbildungen 14.1-1, 14.2-2, 14.3-1und 14.4-1) besteht darin, dass ein WebRTC-Server quasi als Manager multimedialer Verbindungen (Sessions) zwischen WebRTC-Clients fungiert. Über einen WebRTC-Server können zahlreiche, als Web-Videotelefone dienende WebRTC-Clients logisch/virtuell so miteinander verbunden werden, dass sie untereinander über das Internet in IP-Paketen Sprache, Video und Daten übermitteln können. Jeder WebRTC-Server dient somit als WebRTC Session Controller.

Registrar-Funktion im WebRTC-Server

Ein WebRTC-Server kann die Kommunikation zwischen Benutzern nur dann ermöglichen, wenn sie zur Initiierung der Kommunikation berechtigt und dem WebRTC-Server ihre WebRTC-spezifischen Adressen bekannt sind. Hierfür müssen die Benutzer beim WebRTC-Server entsprechend registriert werden. Folglich muss im WebRTC-Server eine spezielle – oft als *Registrar* bezeichnete – Funktion enthalten sein bzw. eine solche Funktion an ihn angebunden werden.

Bedeutung von WebSocket Protocol

Um welche RTC-Funktionen Webbrowser und Webserver erweitert werden müssen, wird aus den im Weiteren gezeigten Systemlösungen ersichtlich. Eine Grundlage für WebRTC bildet z.B. das im RFC 6455 spezifizierte *WebSocket Protocol* (WS-Protokoll bzw. kurz WSP oder WS). Dieses wird im Weiteren detaillierter erläutert.

14.1.2 WebRTC-Server und WebSocket-Protokoll

Wo wird die Funktion vom WebRTC-Server erbracht?

Da das WS-Protokoll normalerweise zwischen Webbrowsern und Webserver zum Einsatz kommt, wird in diversen Publikationen zum Thema WebRTC erläutert, dass die Funktion des WebRTC-Servers von einem Webserver erbracht wird. Aus der Sicht eines RTC-fähigen Webbrowsers kann der WebRTC-Server als eine funktionelle Erweiterung eines Webservers angesehen werden. Für einen RTC-fähigen Webbrowser bildet ein aus einem Webserver und einem WebRTC-Server bestehendes Paar quasi einen RTC-fähigen Webserver. In diesem Kapitel wird aber keine Festlegung getroffen, wo und wie die Funktion des WebRTC-Servers erbracht wird – also ob in einem physischen Webserver oder in einem anderen dedizierten Server, der das WebSocket-Protokoll unterstützt.

Bedeutung des WebSocket-Protokolls

Klassischen Webanwendungen liegt eine request/response-spezifische Kommunikation zwischen Webbrowser und Webserver zugrunde. Diese besteht darin, dass der Webbrowser – immer zuerst – einen Request an den Webserver schickt und dieser dem Webbrowser darauf mit einer Response antwortet. Dies bedeutet also, dass in klassischen Webanwendungen die Kommunikation zum Webserver immer seitens des Webbrowsers initiiert wird. Der Webserver bei klassischen Webanwendungen war nicht dazu befähigt, seinerseits den Aufbau von Verbindungen zu Webbrowsern zu veranlassen.

WebRTC-Server im Webserver

WebRTC ist aber keine klassische Webanwendung mehr. Bei WebRTC muss jeder Webserver, der zusätzlich u.a. die Funktion eines WebRTC-Servers erbringt, in der Lage sein, eine Verbindung zum Webbrowser zu initiieren. Genau betrachtet, muss jeder WebRTC-Server fähig sein, eine Verbindung zum WebRTC-Client im Webbrowser zu initiieren (vgl. Abb. 14.1-1). Ein wichtiges Ziel der Entwicklung des WS-Protokolls war es, die Webserver dazu zu befähigen. Dank der Nutzung des WS-Protokolls können Webserver daher diese – zusätzliche – Funktion von WebRTC-Servern erbringen und dadurch zur Unterstützung von WebRTC verwendet werden, als Verbindungsmanager dienen und u.a. den Aufbau multimedialer Verbindungen zu Webbrowsern initiieren.

> **Bemerkung:** Theoretisch gesehen kann jeder Webserver so um RTC-Funktionen erweitert werden, dass er RTC-fähig wird und als Verbindungsmanager dienen kann. Soll ein „normaler“ Webserver aber eine große Anzahl von WebRTC-Benutzern unterstützen, also einer großen Anzahl von Webbrowsern ermöglichen, untereinander zu kommunizieren, dann ist es sinnvoller, zu diesem Zweck einen separaten Server mit Unterstützung des WS-Protokolls als dedizierten WebRTC-Server zu verwenden.

14.1.3 Signalisierungsprotokoll bei WebRTC

SIP als Signalisierungsprotokoll bei WebRTC

Bei WebRTC handelt es sich um eine multimediale Kommunikation – d.h. um eine Art Videotelefonie. Daher bedarf es eines Protokolls, welches nicht nur dazu dient, virtuelle Verbindungen zwischen WebRTC-Clients in Webbrowsern auf- und abzubauen, sondern Benutzern ankommende Videotelefonieanrufe auch optisch anzuzeigen und bei Bedarf auch akustisch zu signalisieren; hierfür wird also ein Signalisierungsprotokoll benötigt. Als solches eignet sich besonders gut das Protokoll SIP. Es wurde hierfür bereits *SIP in JavaScript* (SIP-JS) implementiert – siehe `https://sipjs.com` für Näheres darüber.

SIP over WS

Wie die Abbildungen 14.1-1 und 14.4-1 zeigen, wird SIP zwischen einem WebRTC-Client und einem -Server verwendet. Damit ein im Webserver zusätzlich installierter WebRTC-Server die zu einem WebRTC-Client im Webbrowser führenden Verbindungen initiieren kann, muss zwischen Webserver und Webbrowser das *WS-Protokoll* (WS: WebSocket) verwendet werden. Nach dem Aufbau einer WS-Verbindung zwischen Webserver und Webbrowser (s. Abb. 14.3-1) werden über diese WS-Verbindung die SIP-Nachrichten in WS-Frames übermittelt. Dies kann als Realisierung von *SIP over WS* verstanden werden – siehe hierzu die Abbildung 14.4-1.

SIP over WSS

Das WS-Protokoll nutzt das verbindungsorientierte Transportprotokoll TCP und kann zur Absicherung der Kommunikation zwischen WebRTC-Client und -Server auch das Sicherheitsprotokoll TLS (*Transport Layer Security*) nutzen. Ist dies der Fall, spricht man vom *Secure WebSocket Protocol* bzw. vom *WebSocket Security Protocol* (kurz *WSS Protocol*). Um Sicherheit bei der Übermittlung von SIP-Nachrichten zwischen WebRTC-Client und -Server zu garantieren, kann außerdem *SIP over WSS* eingesetzt werden.

NAT und ICE bei WebRTC

Um die Realisierung WebRTC-basierter Systemlösungen in Netzwerken mit privaten IPv4-Adressen zu ermöglichen, wird das Protokoll ICE (*Interactive Connectivity Establishment*) verwendet. Wie bereits in Abschnitt 10.6. gezeigt, können mit dem Einsatz des Protokolls SIP bei der Nutzung privater IPv4-Adressen entstehende Probleme mit ICE-Hilfe gelöst werden.

14.1.4 Arten der Kommunikation bei WebRTC

Es sei hervorgehoben, dass bei WebRTC alle Formen der Kommunikation zwischen Webbrowsern realisiert werden können (s. Abb. 14.1-1) – und zwar:

- *Videotelefonie* – d.h. die gleichzeitige Sprach- und Videokommunikation über eine multimediale Session. Hier kommen die Echtzeittransportprotokolle RTP (*Real-time Transport Protocol*) und RTCP (*RTP Control Protocol*) zum Einsatz (s. Abschnitte 5.3 und 5.5). Für alle sicherheitsrelevanten Applikationen ist aber die Nutzung von SRTP (*Secure RTP*) erforderlich (s.

Videotelefonie mit Hilfe von SRTP

Abschnitt 5.7). Mit SRTP können die drei wichtigen Sicherheitsziele – Vertraulichkeit der Kommunikation, Authentifizierung von Nachrichten und Anti-Replay-Schutz – erreicht werden. Für weitere Anforderungen an die Echtzeitkommunikation bei WebRTC sei auf den Internetstandard RFC 8834 (*Media Transport and Use of RTP in WebRTC*) verwiesen.

Datenkommunikation mit Hilfe von SCTP

- *Datenkommunikation* über einen speziellen Datenkanal – als *Data Channel* bezeichnet. Über diesen Datenkanal können mit Hilfe des Protokolls SCTP (*Stream Control Transport Protocol*) mehrere Datenströme in beide Richtungen übermittelt werden. Um die Sicherheit des Datentransports zu gewährleisten, soll hier das Sicherheitsprotokoll DTLS (*Datagram Transport Layer Security*) zum Einsatz kommen; also *STCP over DTLS* realisiert werden. Für weitere Details darüber sei verwiesen auf die Internetstandards RFC 8831, RFC 8832 und RFC 8835.

Damit Kommunikation zwischen Benutzern, die bei verschiedenen WebRTC-Servern registriert sind, stattfinden kann, müssen diese Server untereinander kommunizieren und dabei entsprechende Signalisierungsnachrichten übermitteln. Als Signalisierungsprotokoll zwischen ihnen wird mit Sicherheit SIP oder SIPS (*SIP Secure*) zum Einsatz kommen. Bei einigen Anwendungen kann auch XMPP (*Extensible Messaging and Presence Protocol*) oder die als *Jingle* bezeichnete XMPP Extension verwendet werden.[1]

14.2 Modell der Kommunikation bei WebRTC

Zur Ermöglichung der WebRTC-spezifischen multimedialen Kommunikation im Internet zwischen RTC-fähigen Webbrowsern (s. Abb. 14.1-1) müssen zwischen ihnen besondere virtuelle Verbindungen eingerichtet werden. Hierfür müssen zwischen den Webbrowsern spezielle, als *WebRTC-Signalisierung* bezeichnete Steuerungsvorgänge stattfinden. In folgendem Abschnitt wird die WebRTC-Signalisierung mittels eines Dreiecksmodells veranschaulicht.

14.2.1 Dreiecksmodell von VoIP mit SIP

Ähnlichkeit von Ideen VoIP mit SIP und WebRTC

Mit dem Ziel einer fundierten Erläuterung der Idee von WebRTC zeigt Abbildung 14.2-1 eine entsprechend an die WebRTC-Idee angepasste Darstellung des allgemeinen Konzeptes von VoIP mit SIP innerhalb der als Beispiel dienenden DNS-Domain (*Domain Name System*) `abc.de`. Für eine detaillierte Erläuterung des Konzeptes von VoIP mit SIP sei auf das Kapitel 7 verwiesen. Ba-

[1] Für detaillierte Informationen darüber siehe die Spezifikation „XEP-0166: Jingle“ unter der Adresse: `https://xmpp.org/extensions/xep-0166.html`

sierend auf der in Abbildung 14.2-1 dargestellten Idee von VoIP mit SIP wird anschließend in Abbildung 14.2-2 die Idee von WebRTC erklärt und dabei zum Ausdruck gebracht, dass es sich bei WebRTC de facto um ein dem VoIP mit SIP ähnelndes Konzept handelt.

Dreiecksmodell für den Verlauf der Signalisierung

Wie aus der Abbildung 14.1-1 ersichtlich ist, wird bei VoIP mit SIP als Vermittlungsknoten zwischen als VoIP-Clients bezeichneten Telefonen ein spezieller VoIP-Server, oft auch VoIP-Proxy genannt, eingesetzt. Demzufolge verläuft das Protokoll SIP beim Aufbau einer Session – für Transport von Voice und Video nach den Protokollen RTP/RTCP (*Real-time Transport Protocol*/ *RTP Control Protocol*) – zwischen zwei VoIP-Clients entlang eines Dreiecks. Deshalb kann in diesem Zusammenhang auch vom *Signalisierungsdreieck* bzw. *SIP-Dreieck* gesprochen werden.

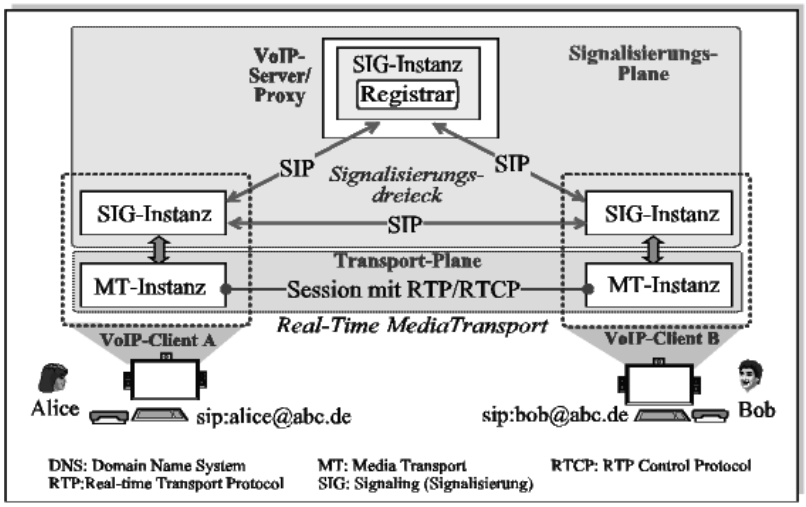


Abb. 14.2-1: Dreiecksmodell für den Verlauf der Signalisierung bei VoIP mit SIP – innerhalb einer DNS-Domain

Im hier gezeigten Dreiecksmodell sind insbesondere folgende zwei Planes hervorzuheben:

- *Signalisierungs-Plane*, in der das Signalisierungsprotokoll SIP verläuft, um zwischen den MT-Instanzen in beiden VoIP-Clients eine Session mit den Protokollen RTP/RTCP für den Transport von Voice und Video einzurichten.
- *Transport-Plane*, innerhalb der die Echtzeitmedien Voice und Video über die zwischen den MT-Instanzen eingerichtete Session transportiert werden.

SIP-Server als zentrale Signalisierungsinstanz

Eine neue, als Session bezeichnete virtuelle Verbindung wird bei VoIP mit SIP mit der SIP-Nachricht `INVITE` initiiert, in der die als SIP-URI bezeichnete Ziel-VoIP-Adresse `sip:bob@xyz.de` enthalten ist. In Abbildung 14.2-1 ist der als VoIP-Client A bezeichnete Rechner von Alice der Initiator einer Session zu Bob. Beim Initiieren einer neuen Session wird `INVITE` an den SIP-Server übergeben. Dieser muss `INVITE` an das Ziel – hier zum VoIP-Client B – weiterleiten.

Hat `INVITE` den Rechner von Bob erreicht und die gewünschte Session kann zustande kommen, wird dies Bob akustisch durch Klingeln (Ringing) signalisiert und auch dem Rechner von Alice mittels der SIP-Nachricht `180 Ringing` mitgeteilt. Hat Bob den Anruf entgegengenommen, wird dies dem Rechner von Alice mittels der SIP-Nachricht `200 OK` angezeigt. Die beiden Nachrichten `180 Ringing` und `200 OK` werden über den SIP-Server übermittelt. Nach dem Empfang dieser Nachrichten kennt der Rechner von Alice bereits die IP-Adresse von Bobs Rechner und sendet daher die SIP-Nachricht `ACK` als Bestätigung des Empfangs von `200 OK` direkt an diesen – ohne den VoIP-Server nutzen zu müssen. Dadurch entsteht ein SIP-Dreieck innerhalb der Signalisierungs-Plane.

Nach dem Eintreffen von `200 OK` bei Bobs Rechner wird der Aufbau der Session beendet, und die beiden Medien Voice und Video können mit Hilfe von RTP/RTCP in IP-Paketen zwischen MT-Instanzen über diese Session übermittelt werden.

14.2.2 WebRTC-Dreiecksmodell – ohne Transcoder-Einsatz

Ähnlichkeit zwischen Dreiecksmodellen von VoIP und von WebRTC

Auf der Grundlage des in Abbildung 14.1-2 gezeigten Dreiecksmodells der Signalisierung bei VoIP mit SIP kann die grundlegende Idee von WebRTC anhand eines ähnlichen Dreiecksmodells dargestellt werden. Abbildung 14.2-2 zeigt ein solches Dreiecksmodell von WebRTC und verdeutlicht dabei, dass die beiden Konzepte VoIP mit SIP und WebRTC sehr ähnlich sind. Es gibt aber auch einige Unterschiede, auf die im Folgenden näher eingegangen wird.

Bei WebRTC soll in jedem Rechner, beispielsweise durch Anklicken auf einen Link bzw. auf ein Icon, die Möglichkeit bestehen, mittels des Protokolls HTTPS (*HTTP Secure*) aus einem Webserver einen WebRTC-Client herunterzuladen und diesen in den Webbrowser quasi automatisch zu integrieren.

Damit jeder Webserver auch als Vermittlungsknoten bei WebRTC, genauer gesagt als WebRTC-Server, dienen kann, wurde das verbindungsorientierte *WebSocket Protocol* (WS-Protokoll) zur Übermittlung von Signalisierungsnachrichten zwischen WebRTC-Client und -Server entwickelt. Das WS-Protokoll ist ein verbindungsorientiertes Protokoll oberhalb des Transportprotokolls TCP. Bei

WS kann auch das Protokoll TLS[2] (*Transport Layer Security*) verwendet werden; ist dies der Fall, so spricht man von *Secure WS* (WSS).

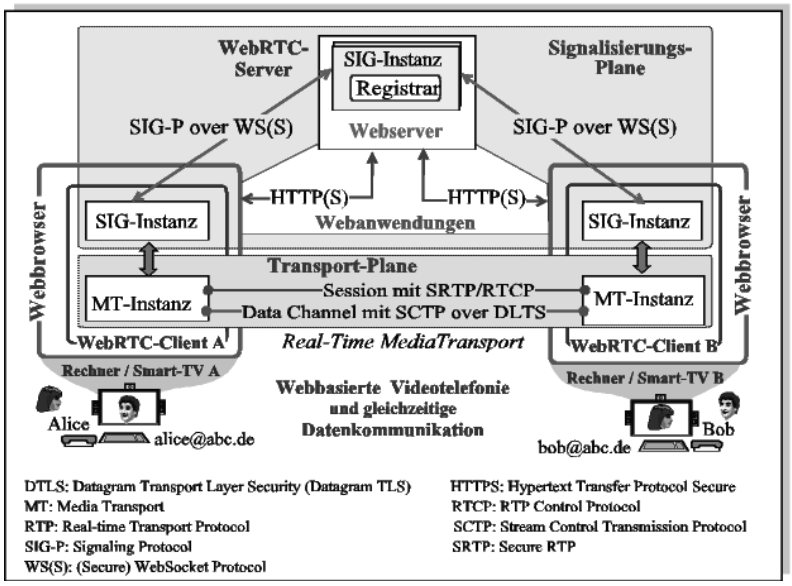


Abb. 14.2-2: Modell für den Verlauf der Signalisierung bei WebRTC – vergleiche dieses mit dem in Abb. 14.2-1 gezeigten Modell für VoIP mit SIP.

SIG-P over WS bzw. over WSS

Mit Hilfe des WS-Protokolls ist der WebRTC-Server, welcher auf einem „normalen" Webserver oder auf einem speziellen Server mit Unterstützung des WS-Protokolls installiert ist, in der Lage, eine Verbindung zum WebRTC-Client zu initiieren. Bei WebRTC verläuft das Signalisierungsprotokoll SIG-P (wie z.B. SIP) somit über WS oder WSS. Dies wird kurz als *SIG-P over WS(S)* bezeichnet. Die Nachrichten von SIG-P werden zwischen WebRTC-Client und -Server in WS-Frames (s. RFC 6455 – *The WebSocket Protocol)* transportiert:

- *bei SIG-P over WS* – über eine normale, also ungesicherte TCP-Verbindung,
- *bei SIG-P over WSS* – über eine mit Hilfe des Sicherheitsprotokolls TLS gesicherte TCP-Verbindung.

Kein Signalisierungs-dreieck bei WebRTC

Da das WS-Protokoll nur zwischen WebRTC-Client und -Server verwendet wird, können die Nachrichten eines Signalisierungsprotokolls bei WebRTC nur über WebRTC-Server übermittelt werden und nicht, so wie es bei VoIP mit SIP der Fall war, auch direkt zwischen SIP-Instanzen in Webbrowsern. Demzufol-

[2] Für allgemeine Informationen über TLS sei verwiesen auf die Adresse:
`https://www.researchgate.net/publication/292995131`

ge entsteht bei WebRTC, anders als bei VoIP, kein *Signalisierungsdreieck* – vergleiche hierfür die Abbildungen 14.2-1 und 14.2-2.

Datenkanal bei WebRTC mit SCTP over DTLS

Im Gegensatz zu VoIP mit SIP sind bei WebRTC die MT-Instanzen (*Media Transport*) in der Lage, untereinander einen gesicherten Datenkanal (*Data Channel*) zum Datentransport einzurichten. Um verteilte Datenanwendungen und verschiedene Arten multimedialer Kommunikation zu ermöglichen, kommt hierfür das Protokoll SCTP over DTLS zum Einsatz.

Gemeinsames Socket für RTP und RTCP bei WebRTC

Es sei hervorgehoben, dass es einen wesentlichen Unterschied zwischen einer multimedialen Session bei VoIP und einer multimedialen Session bei WebRTC gibt. Zwar werden die Protokolle RTP (*Real-time Transport Protocol*) und RTCP[3] (*RTP Control Protocol*) bei beiden Arten von Sessions eingesetzt, aber bei WebRTC ist es anders als bei VoIP. Im Unterschied zu VoIP wird bei WebRTC ein gemeinsames Socket für RTP und RTCP verwendet. Diese Lösung verfolgt das Ziel, die Nutzung privater IPv4-Adressen bei WebRTC zu erleichtern. Um private IPv4-Adressen bei WebRTC nutzen zu können, soll das Protokoll ICE zur Lösung des Problems NAT (*Network Address Translation*) zum Einsatz kommen (s. Abschnitt 10.6.7).

14.2.3 WebRTC-Dreiecksmodell – mit Transcoder-Einsatz

Mehrere Audio- und Video-Codecs

Die Verwirklichung der WebRTC-Idee soll Benutzern während einer Echtzeitkommunikation die Übermittlung aller Informationsformen – also von Audio und Video sowie Daten – ermöglichen. Aber damit sie erfolgreich kommunizieren und die gemeinsam genutzten Medien auch präsentieren können, müssen sich die kommunizierenden Webbrowser auf einzusetzende Codecs, insbesondere Audio- und Video-Codecs, verständigen.

Bei WebRTC soll die Anzahl der Audio- und Video-Codecs sehr eingeschränkt sein. Es werden die folgenden Codecs verbindlich vorgesehen:[4]

- für Audio der Codec nach dem ITU-T-Standard G.711 und der im RFC 6716 spezifizierte *Opus* Codec,
- für Video der Codec nach dem ITU-T-Standard H.264 und der lizenzgebührenfreie Codec VP8. Im Hinblick darauf sei verwiesen auf RFC 7741 mit der Spezifikation von „RTP Payload Format for VP8 Video“.

[3] Das Protokoll RTCP wird auch als *Real Time Control Protocol* bezeichnet.

[4] Für weitere Informationen über „Codecs used by WebRTC“ sei verwiesen auf die Adresse: `https://developer.mozilla.org/en-US/docs/Web/Media/Formats/WebRTC_codecs`