WebRTC in Second Life: Ein Entwicklerblick
Dieses Projekt beschreibt die Migration der Sprachkommunikation in Second Life auf **WebRTC (Web Echtzeitkommunikation)**.

Hinweis: Dies ist ein lebendiges Dokument und wird bei auftretenden Fragen aktualisiert.

Hintergrund

WebRTC (Web Real-Time Communications) ist das vorherrschende Telekommunikationsprotokoll, das von webbasierten Anwendungen wie Google Meet verwendet wird. Es ist in Chrome, Safari, Firefox und vielen anderen Webbrowsern integriert und ermöglicht sowohl Audio- als auch Daten- und Videokommunikation. All diese Browser unterstützen eine offene API, die über JavaScript auf WebRTC-Funktionen zugreifen lässt. Die Kernimplementierung von WebRTC ist ein Open-Source-C+±Paket, das aktiv von Google und vielen anderen gepflegt wird (https://webrtc.org/).

Die von der webrtc.org Implementierung bereitgestellten Funktionen umfassen:

- NAT hole punching über STUN.
- Datenübertragung über TURN.
- Audio-, Video- und Datentransmission.
- Auswahl von Audio-/Videogeräten.
- Stereoton.
- Konfigurierbare Audio-/Videobandbreite zur Steuerung der Audio- und Videoqualität.
- Geräuschreduktion im Audio.
- Automatische Lautstärkeregelung im Audio.
- Echounterdrückung im Audio.
- Mehrere Audiospuren pro Stream.
- Mehrere Streams pro Verbindung.
- Gleichzeitige Verbindungen zu mehreren WebRTC Quellen (Server, Clients usw.).
- Peer to Peer Verbindung (direkt oder über Relay).
- Kommunikation mit Peers über Server (SFU, MCU).
- Verbesserung der Privatsphäre und Sicherheit.
- Und vieles mehr.

Wir wechseln zu einer selbst gehosteten WebRTC-basierten Sprachkommunikation, um diese Funktionen zu nutzen und die Tür für zukünftige Erweiterungen zu öffnen.

Da das Sprachuntersystem im Viewer modular ist, sind die wesentlichen Änderungen auf ein neues WebRTC Sprachmodul und eine unterstützende dynamische Bibliothek beschränkt, die die native WebRTC Bibliothek einschließt.

Änderungen im Vergleich zu Vivox umfassen:

- Räumliche Sprache für eine bestimmte Region wird von einem Second Life Voice Server in Verbindung mit dieser Region verwaltet.
- Ad-Hoc- und Gruppensprache wird von einem Pool von Second Life Voice Servern behandelt, die für diese Arten von Sprache gewidmet sind.
- Im Gegensatz zum Direktanrufmechanismus von Vivox wird WebRTC-P2P über den Ad-Hoc-Voice-Server-Pool geroutet, als wäre es eine Ad-Hoc-Voice-Konferenz mit zwei Personen.
- Es gibt keine separate SLVoice-Executable für WebRTC Voice (kurzfristig werden wir sowohl WebRTC Voice als auch Vivox Voice zulassen, damit SLVoice ausgeführt wird, um Vivox-

Unterstützung zu bieten).

WebRTC Voice-Schnittstelle

Die folgende Dokumentation beschreibt die Funktionen und die Datenschnittstelle für die WebRTC-Sprachnutzung in WebRTC-räumlich aktivierten Regionen und in Umgebungen, die WebRTC-P2P/Ad-Hoc/Gruppensprache ermöglichen.

Es wird davon ausgegangen, dass der Leser ein grundlegendes Verständnis für die WebRTC-Signalisierung hat.

Funktionen / Peer Verbindungsbereitstellung

Second Life WebRTC Voice verwendet zwei Fähigkeiten, die vom Viewer (Desktop oder Mobilgerät) an den Simulator übermittelt werden.
Diese beiden Fähigkeiten gelten sowohl für räumliche als auch für P2P/Ad-Hoc/Gruppensprache.

ProvisionVoiceAccountRequest

Die ProvisionVoiceAccountRequest Fähigkeit nimmt einen Beitrag mit einem der folgenden beiden Payloads entgegen, in LLSD (wie CAPS im Allgemeinen).

Räumliche Sprache

Um sich mit einer Sprachsession für eine Region (oder ein Grundstück) zu verbinden, sendet der Client folgendes:

<llsd>
<map>
<key>jsep</key>
<map>
<key>type</key>
<string>offer</key>
<key>sdp</key>
<string>....sdp content…</string>
</map>
<key>parcel_local_id</key>
<integer>2</integer>
<key>channel_type</key>
<string>local</string>
<key>voice_server_type</key>
<string>webrtc</string>
<map>
</llsd>

Das JSEP (Javascript Session Establishment Protocol) enthält die Informationen, die für die Verhandlung einer WebRTC-Verbindung zwischen dem Client und dem Second Life Voice Server benötigt werden. Es handelt sich um ein "Angebot" in der WebRTC-Terminologie.
Die parcel_local_id ist eine optionale Ganzzahl zur Identifizierung des Grundstücks, die auf die Region beschränkt ist.
Der channel_type ist entweder lokal für räumliche Verbindungen oder multiagent für Ad-hoc-/P2P-/Gruppenverbindungen.

voice_server_type sollte ausschließlich webrtc sein.

Die oben genannte LLSD wird an den Simulator übergeben, der dann mit dem Second Life Voice Server verhandelt, um eine Antwort sdp abzurufen, die anschließend wie folgt als LLSD zurückgegeben wird.

<llsd>
<map>
<key>jsep</key>
<map>
<key>type</key>
<string>answer</key>
<key>sdp</key>
<string>....sdp content…</string>
</map>
<key>viewer_session</key>
<string>...viewer session…</string>
</map>
</llsd>

Das Angebot sollte von WebRTC beim Erstellen einer Peer-Verbindung abgerufen werden und sollte so verändert werden, dass die Zeile a=fmtp: einen Wert von minptime=10;useinbandfec=1;stereo=1;sprop-stereo=1;maxplaybackrate=48000 hat.

Weitere Informationen zu den SDPs im Angebot und in der Antwort finden Sie hier.

Verbindungen werden durch Senden des folgenden LLSD an ProvisionVoiceAccountRequest geschlossen:
<llsd>
<map>
<key>logout</key>
<boolean>true</boolean>
<key>voice_server_type</key>
<string>webrtc</string>
<key>viewer_session</key>
<string>...viewer session…</string>
</map>
</map>

HINWEIS: Die CAP ist aus historischen Gründen als "ProvisionVoiceAccountRequest" benannt, da Vivox so eingerichtet ist, dass ein Sprachkonto erstellt wird, das mit einem Sprachkanal über SIP verbunden ist. Technisch gesehen erstellt WebRTC einfach einen Sprachkanal, der mit dem Second Life-Konto verbunden ist.

Die 'logout'-Anforderung kann einen 404-Fehler zurückgeben, wenn die Verbindung bereits durch andere Mittel geschlossen wurde, wie z.B. das Verlassen des Avatars, das Herunterfahren der WebRTC-Verbindung usw.

AdHoc/Gruppen-/P2P-Sprache

Die Multiagenten- (AdHoc/Gruppen-/P2P-) Sprache verwendet einen Pool von nicht raumfähigen Second Life Voice Servern, um Sprachsessions zwischen zwei oder mehr Personen herzustellen.

Das Chat Untersystem in Second Life verhandelt einen Anruf, der Informationen liefert, die zum Verbinden mit der Sitzung benötigt werden, wie z.B. die Kanal-ID, Anmeldeinformationen, Sprachservertyp usw.
Diese Informationen werden über die ProvisionVoiceAccountRequest in folgender Form übermittelt:
Kanal - Kanal-ID
Anmeldeinformationen - Kanal-Anmeldeinformationen.

<llsd>
<map>
<key>jsep</key>
<map>
<key>type</key>
<string>offer</key>
<key>sdp</key>
<string>....sdp content…</string>
</map>
<key>channel</key>
<string>...channel id…</string>
<key>credentials</key>
<string>...channel credentials…</string>
<key>channel_type</key>
<string>multiagent</string>
<key>voice_server_type</key>
<string>webrtc</string>
<map>
</llsd>

Die Verbindung wird über den gleichen Schließmechanismus geschlossen, der oben für räumliche Sprache verwendet wurde.

VoiceSignalingRequest

Die andere CAP, die bei WebRTC räumlicher Sprache verwendet wird, ist die VoiceSignalingRequest CAP. Sie wird ausschließlich verwendet, um ICE Kandidaten nach Abschluss der Sitzung tröpfeln zu lassen.

Es handelt sich um eine POST-Anforderung, die das folgende LLSD für neue ICE Kandidaten annimmt:

<llsd>
<map>
<key>candidates</key>
<array>
<map>
<key>sdpMid</key>
<string>sdp mid value</string>
<key>sdpMLineIndex</key>
<integer>index</integer>
<key>candidate</key>
<string>ice candidate string</string>
</map>

<map>
<key>sdpMid</key>
<string>sdp mid value</string>
<key>sdpMLineIndex</key>
<integer>index</integer>
<key>candidate</key>
<string>ice candidate string</string>
</map>
…
</array>
<key>voice_server_type</key>
<string>webrtc</string>
<key>viewer_session</key>
<string>...viewer session…</key>
</map>
</llsd>

Jeder ICE Kandidat wird durch eine Kandidatenzeichenfolge definiert, die hier detailliert beschrieben ist.

Es wird erwartet, dass eine oder mehrere dieser ICE-Kandidaten-POST-Aufrufe NACH dem Empfang der Antwort aus dem Angebot von ProvisionVoiceAccountRequest gemacht werden. Alle ICE-Kandidaten, die vor dem Empfang der Antwortantwort gesammelt werden, sollten zwischengespeichert und nach dem Empfang der Antwort gesendet werden.

Das Signalisieren von ICE Kandidaten nach Erhalt der Antwort wird als ICE Trickling bezeichnet.

Wenn die Sammlung abgeschlossen ist, sollte das Folgende über die VoiceSignalingRequest-CAP gesendet werden.
<llsd>
<map>
<key>candidate</key>
<map>
<key>completed</key>
<boolean>true</boolean>
</map>
<key>voice_server_type</key>
<string>webrtc</string>
<key>viewer_session</key>
<string>...viewer session…</key>
</map>
</llsd>

Datenkanalschnittstelle für WebRTC Sprache

Der Server hört auf und sendet Daten über den "SLData" Datenkanal. Clients müssen den "SLData"-Datenkanal vor der Verhandlung erstellen, genauso wie Clients die Audio Kanäle erstellen. Die Kommunikation über die Datenkanäle erfolgt in Form von JSON Strings (im Gegensatz zu Binärdaten).

Client -> Mixer
Das Basismodul des JSON Objekts, das an den Server gesendet wird, kann eines der folgenden

Elemente enthalten:
- "j": join (ein Objekt) - gibt an, dass dieser Client einer Sitzung beitritt. Das einzige Element des Objekts ist eine Option "p", die angibt, dass es sich um eine primäre Verbindung handelt. Primäre Verbindungen sind diejenigen, für die Audiolautstärken an den Client zurückgestreamt werden. Der Standardwert für "p" ist false.

- "l": leave (ein boolescher Wert, immer true) - gibt an, dass dieser Client eine Sitzung verlässt.

- "sp": Senderposition (ein Objekt) mit den ganzen Zahlen "x", "y" und "z" für die Koordinaten des Senders. Diese Koordinaten sind Weltkoordinaten (nicht regionsbezogen) in Zentimetern. (nur für räumliche Sprache)

- "sh": Senderausrichtung (ein Objekt) mit den ganzen Zahlen "x", "y", "z" und "w" für die Quaternionenwerte (multipliziert mit 100). (nur für räumliche Sprache)

- "lp": Zuhörerposition (ein Objekt) mit den ganzen Zahlen "x", "y" und "z" für die Koordinaten des Zuhörers. Diese Koordinaten sind Weltkoordinaten (nicht regionsbezogen) in Zentimetern. (nur für räumliche Sprache)

- "lh": Zuhörerausrichtung (ein Objekt) mit den ganzen Zahlen "x", "y", "z" und "w" für die Quaternionen (multipliziert mit 100). (nur für räumliche Sprache)

- "m": Agenten stumm- oder lautzuschalten. Ein Objekt, das Agenten-IDs mit true/false (stumm/laut) verknüpft.

- "ug": Einstellen der Verstärkung bei Agenten. Ein Objekt, das Agenten-IDs mit ganzen Zahlen für die Verstärkung (Verstärkung * 200) verknüpft.

Alle Werte sind für ein Update nicht erforderlich. Es wird erwartet, dass Werte nur gesendet werden, wenn sie geändert werden, um die Bandbreite zu reduzieren.

Beispiel:

{
“j”: { “p”: true },
“sp”: { “x”: 37000, “y” : 2400, “z” : 37201 },
“lh” : { “x”: 100, “y” : 50, “z”: 1, “w”: 100 },
“m” : {
“98d0084e-a6eb-45ce-b847-4da34ea823a8”:true,
“95c8ef1a-edcb-467c-88fe-ea503e5b974e”: false
},
“ug”: {
“98d0084e-a6eb-45ce-b847-4da34ea823a8”:27,
“95c8ef1a-edcb-467c-88fe-ea503e5b974e”: 125
},
}

Mixer -> Client
Der Mixer sendet JSON an jeden Client, das ein Objekt enthält, wobei jedes Element ein Objekt ist, das nach der Agenten-ID des Peers indiziert ist. Die folgenden Werte werden in diesen pro Peer-Objekten erscheinen:
- "p": Leistungsniveau (eine Ganzzahl), die RMS * 128 entspricht (für die individuellen VU-Meter).
- "V": VAD (Voice Activity Detected). Der Wert ist true, wenn der Peer spricht.

- "j": join (ein Objekt) - gibt an, dass dieser Client einer Sitzung beitritt. Das einzige Element des Objekts ist eine Option "p", die angibt, dass es sich um eine primäre Verbindung handelt. Primäre Verbindungen sind diejenigen, für die Audiolautstärken an den Client zurückgestreamt werden.
- "l": leave (ein boolescher Wert, immer true) - gibt an, dass dieser Peer eine Sitzung verlassen hat.

Ein Beispiel sieht wie folgt aus:

{
“98d0084e-a6eb-45ce-b847-4da34ea823a8” : { “p” : -43, “v”: true },
“b27a917b-fc3a-4bf4-bb50-e179d23e47c4” : {“j”: { “p”: false } },
“95c8ef1a-edcb-467c-88fe-ea503e5b974e” : {“l”: true}
}

Diese Werte und Benachrichtigungen werden in relativ schneller Folge (alle 100 ms) gebündelt und an den Client gesendet.

Räumliche Sprache über Regionsgrenzen hinweg

Um Sprache über eine Regionsgrenze hinweg zu handhaben, wird erwartet, dass der Client sowohl eine Verbindung zu seiner eigenen Region als auch zu benachbarten Regionen herstellt, die sich im Hörbereich befinden. Der Client sollte dann Positionsupdates an alle verbundenen Regionen senden. Der Client sollte Audio von allen verbundenen Regionen wiedergeben, jedoch sollte nur Audio an seine primäre Region gesendet werden, wobei das Senden von Audio an andere Regionen deaktiviert ist. Um dem Client bei der Verwaltung zu helfen, implementiert der Server das Konzept der "primären" Regionen. Wenn ein Client seiner eigenen Wohnregion beitritt, sollte er sie als primär markieren, indem er in seiner "j" (join)-Nachricht über den Datenkanal "p" auf true setzt. Andere Regionen sollten mit "j->p" auf false markiert sein. Der Server wird das Streamen von Audio zu/von Regionen je nach "primärer" Einstellung entsprechend durchsetzen, aber auch vom Client erwartet, um den Audio-Bandbreitenverbrauch zu reduzieren.

Teleportation

Beim Teleportieren zu einer anderen Region oder beim Überqueren einer Regionsgrenze sollte der Client zuerst feststellen, ob er bereits eine Verbindung zu dieser Region (oder den benachbarten Regionen) geöffnet hat. Falls ja, kann der Client einfach das Ausgehende Audio von der alten Region zum Client deaktivieren und ausgehendes Audio für die neue Region aktivieren. Wenn der Client keine Verbindung zur neuen Region hat, sollte er eine herstellen und die Verbindung zur alten Region trennen. Ein Algorithmus, der verwendet werden kann, um die benachbarte Region und das Teleportationsverhalten zu bestimmen, ist:
1) Bestimmen Sie die Liste der nahe gelegenen Regionen.
2) Stellen Sie eine Verbindung zu allen Regionen her, die derzeit nicht verbunden sind.
3) Trennen Sie die Verbindung zu allen verbundenen Regionen, die nicht in dieser Liste enthalten sind.

Grundstückssprache

Beim Betreten eines Grundstücks mit aktivierter Sprachfunktion, die jedoch auf sich selbst beschränkt ist, sollten alle anderen Verbindungen geschlossen und eine neue Verbindung zu diesem Grundstück hergestellt werden. Wenn die Sprache auf dem Grundstück deaktiviert ist, sollten alle Verbindungen geschlossen oder minimal deaktiviert werden.