Wirow

Ein WebRTC Konferenz und Webminar System

1) Leistung laut Anbieter

- Einfach zu installieren (ein Binary).
- Schnell und wenig RAM Verbrauch.
    - in C und C++ programmiert.
- Datei Upload.
- Integrierten Whiteboard.
- Benutzer Verwaltung.
- Raum Verwaltung.
- TLS Zertifikat von **Let's Encrypt** wird automatisch erzeugt.

Das Kompilieren stelle sich nicht als eine einfache Sache, es werden viele Komponente, wie beispielsweise Mediasoup (ein Media-Server) eingebunden. Unter Fedora 35 gelang das Erzeugen des ausführbaren Programms, auf ein Ubuntu-System, wie von der Readme Datei, nah gelegt gibt es nicht wirklich, einige Abhängigkeiten sorgten dafür, dass notwendige Komponente aus der Repositories nicht installiert werden konnten.

Als Server diente eine VPS (Ionos) mit Debian. Damit konnte für das erste Wirow nicht ausgeführt werden, einiges ist auf Debian-basierten Systeme restlos veraltet, dies führte zu Fehlern (libc.so Inkompatibilität).

Mit patchelf konnten die entsprechende Libraries überarbeitet werden und aud das Debian System installiert werden (eigene Verzeichnis). Neuerzeugen Binaries mussten auch, nach der Erzeugung (per Script) gepatch werden. Das Kopieren und Ausführen auf den Server gelang dann.

2) Galène

Ein weitere Server aus Frankreich, es wird im universitäre Bereich verwendet.

- Programmierung in go, daher kann es auf viele Platformen eingesetzt werden.
- Einfache Konfiguration.
- Große Flexibilität.
- Galène kann so eingestellt werden, dass alle Loginname akzeptiert werden, sie werden als Anzeigename genutzt. Das Betrieb ist sinnvoll wenn Gäste an ein Videokonferenz teilnehmen sollten.
- **Nachteil, ist dass die Oberfläche nur englisch ist** und eine Übersetzung sicherlich mit viel Arbeit verbunden wäre.
- Verwendete Kamera und Microfon, Login-Name usw. werden nicht local gespeichert.
- Eine weiterleitung auf https ist nicht vorgesehen, dies dürfte einige Anwendern stören.

Da eine Internationalisierung nicht vorhanden ist und einige kleinere Ungereimtheiten vorhanden sind scheidet Galène aus.

3) Funktionalität einem Videokonferenzsystem

Je nach Einsatzzweck können die Anforderungen sehr unterschiedlich sein.

Merkmal	Wirow	Jitsi	Kommentar
Raumreservierung	Ja	Nein	
Moderation	Nein	?	Bei Jitsi-Meet wäre der erste Anwender automatisch Moderator!
Mediageräte aus	Nein	Ja	Nicht nur der Moderator kann alle Kameras und oder Mikrofone abschalten
Hand heben	Nein	Ja	Wenn die Signallaufzeit groß sind, muss das Wort explizit erteilt werden
Chat	Ja	Ja	
Bildschirm teilen	Ja	Ja	Unterschiedliche Philosophien
Darstellung umschalten	Ja	Ja	Unterschiedliche Vorgehensweise
Hintergrund Änderung	Nein	Ja	Unter Jitsimeet scheint es zu Probleme zu kommen

3.1) Reservierung - Moderation

- Einsatzzweck:
  - Moderierte SHG Arbeit
  - Die Teilnehmer möchten Informationen erhalten, Frage stellen.
  - Sie sind im deutschsprechenden Raum verteilt.
  - Sie haben keine große Erfahrung (Video-Konferenz).

Der Videoraum sollte nur von Menschen, die die URL kennen erreichbar sein, am besten ohne Login. Ein Gebrauch des Angebotes sollte, für Fremden untersagt sein (Ressourcenverbrauch).

Damit solch ein Video Treffen gelingt, die Teilnehmer sich gegenseitig stören ist eine Moderation angebracht. Da die Laufzeiten recht groß werden könnnen, ist ein geregelten Ablauf des Treffens möglicherweise erschwert. Jemanden spricht, legt eine kurze Pause ein, ein andere nutzt die Gelegenheit umd das Wort zu nehmen und schon herrscht Chaos. Eine **Moderation ist notwendig!**

Als normale Anwender können die Kamera und oder Microphone der anderen Teilnehmer angeschaltet werden, die Moderartionsrechte bestehen offenbar nur im Setzen einen Passwort oder ähnliches.

3.2) Handheben

- Auf sich aufmerksam machen
  - Die moderierende Person kann es wahrnehmen.
  - Eine Steuerung der Gespräche ist leichter möglich.
  - Ersatzweise kann einer echten Hand erhoben werden, ist allerdings nicht optimal.

Dieser Punkt gehört eigentlich zu den wichtigsten Moderationshilfen, wenn die Teilnehmer ungeschult sind.

3.3) Chat

- Warum Chat
  - Die Chatfunktionalität ist wichtig.
  - Informationen können an alle Teinehmer in Schriftform gegeben werden (Links, Mitteilungen, …)

Im Laufe einen Video-Treffen können weitere Informationen angeboten werden, vertiefende Literatur, Link auf ein Webseite, bei dem man weitere Informationen usw. beziehen kann. Eine nur Oral vermittelte Information ist sehr schnell vergessen, kann aber den Chat entnommen werden.

Wenn dass Handheben nicht vorhanden ist, könnte es als Ersatz verwendet werden.

3.4) Bildschirm teilen

Anzeige von wichtige Informationen.

Durch Teilen des Bildschirmes (oder einem einzelnen Fenster) können Einige Informationen geteilt werden, allerdings als Bildmaterial.
Vorträge können leicht gehalten werden.

Auch wenn jeder, sowohl mit Jitsi-meet wie mit Wirow ihr Bildschirm teilen können, ist die Funktion, bei Wirow flexibler.

3.5) Darstellung

Was möchte man sehen:

- Bisherige Erfahrungen zeigen, dass die meisten Anwender alle anderen
- Teilnehmer auf das Bildschirm sehen möchten.
- Einige mögen eher, dass die sprechende Person den ganuen Bildschirm platz einnimmt.
- Wenige möchten sich auf eine einzelne Person konzentieren.

Jitsi-meet bittet, als erste Bildschirmdarstellung eine Ansicht in dem alle Teilnehmer in eine smalle Leiste am rechten Rand angezeigt wird. Der Anwender muss aktiv die Kacheldarstellung einstellen (falls er diese findet) oder ein der Teilnehmer anklicken um derjenige auf das gesamten Bildschirm (Browserfenster) zu bringen. Sollten nur 2 Teilnehmer sich unterhalten, ist der Gespächpartners auf das gesante Fenster abgebildet.

Wirow folgt ein anderen Ansatz, eine Kacheleinstellung ist das Default. Einzelne Teilnehmer können auf den vollen Platz durch klicken des entsprechenden Feld des übertragene Bild (Andeuten Symbol rechts, unten im jeweilige Kachel).

Beide Vorgehensweise dürfte als gleichwertig betrachtet werden, auch wenn jitsi-meet mehr (verwirrenden?) Einstellung bittet.

Die Bedienelemente (Mikrofon ein- / ausschalten, …) befinden sich bei Jitsi-meet mittig am unteren Rand des Fensters, werden jedoch ausgeblendet so, dass unerfahrene Anwender immer wieder damit Probleme haben.

Bei Wirow sind diesen Elementen in eine schmale vertikale Leiste am linken Fensterand angeordnet und werden immer angezeigt.

4) Technische Gesichtspunkte

- Jitsi.meet:
  - Jitsi-meet ist mit Java eine interpretierte Sprache (jedoch optimiert geschrieben).
  - Jitsi-Meet besteht aus getrennte Komponente die jeweils installiert und konfiguriert werden müssen.
- Wirow:
  - Wirow ist in C geschrieben, einige der Komponente in C++. Damit kann eine höhere Leistungsfähigkeit erreicht werden.
  - Installation und Konfiguration sind einfach.

Einiges spricht für Wirow, auch wenn Verbesserungen von Nöte sind. Manches ist unter Jitsi-Meet besser durchdacht, Wirow hat dafür bei andere Punkten das bessere Konzept.

Einige Beigaben, die interessant sein könnten, ist dass eine digitale Tafel vorhanden ist (Whiteboard) und dass Dateien durch angemeldeten Gäste hoch- und heruntergeladen werden können.

Das Screensharing kann, gleichzeitig von jeden Teilnehmer vorgenommen werden. Bei Jitsi-meet wurde es nicht probiert. Jitsi-meet erlaubt es Hintergrund Bilder einzustellen, es ist sehr schön, scheint aber viele Resourcen zu verbrauchen und einiges zu hindern.

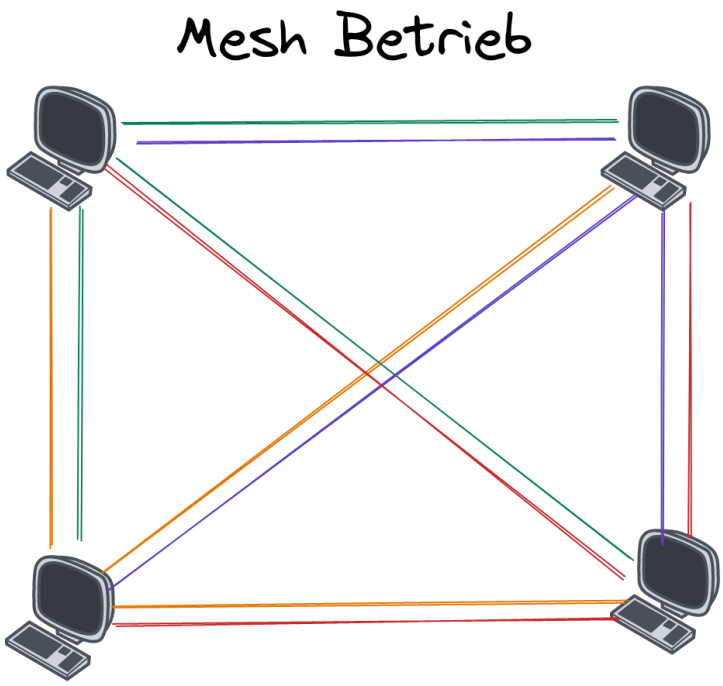
4.1) MESH, MCU, SFU, …

- MESH (Direkte Verbindung zwischen den Browser)
  - Anzahl der Verbindungen je Rechner 2 * N * (N-1), die einzelne Systeme werden schnell überfordert!
- SFU (Selective Forwarding Unit), der WebRTC Server erhält die Streams jeden einzelnen Rechner und sendet diesen an den Client.
  - Sowohl Wirow wie Jitsi-meet verwenden eine SFU (Selective Forwarding Unit)
- MCU Wie SFU aber mit Verarbeitung der Streams (Umkodierung, Mischung, …)
  - Eine MCU wird, im private Umfeld eigentlich nicht verwendet, sie erfordert sehr leistungsfähige Server.
  - Vorteil jede Client muss nur je ein Video/Audio Ausgangsstream aufbauen.
  - BigBlueButton verwendet einer Software MCU.

Jitsi-Meet verwendet sowohl das MESH wie das SFU Betrieb, die Umschaltung des Betriebes hängt vom Anzahl der Teilnehmer ab.

Wirow verwendet immer die SFU.

4.1.1) Mesh



Mesh-Betrieb

Problematisch ist die DSL-Geschwindigkeit, manche Teilnehmer verfügen nur auf ein 16/1 MBps Vertragsmöglichkeit, dies reduziert die Verwendbarkeit des Mesh-Betrieb drastisch.

4.1.2) Selective Forwarding Unit: Funktion

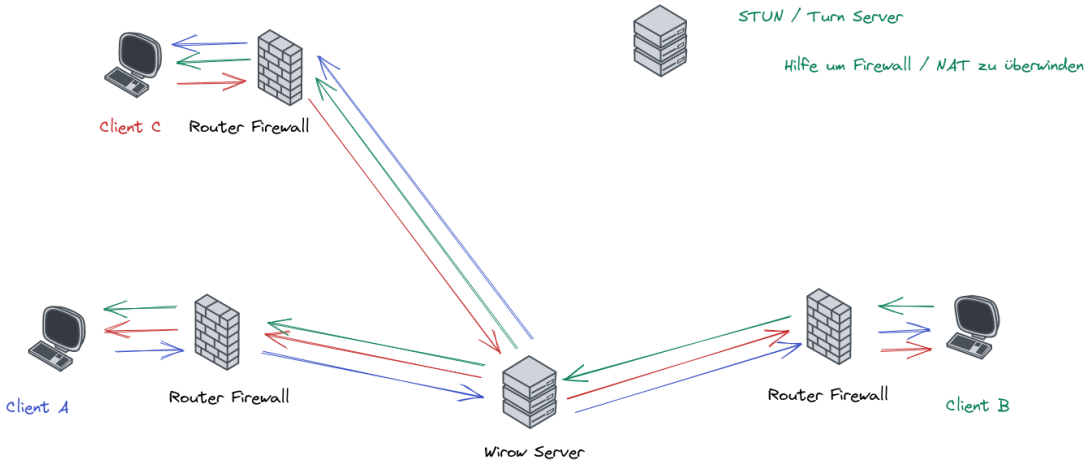
- Falls mehr als 2 Teilnehmer einer Konferenz beigetreten haben, spielt die SFU seine Vor. und Nachteile.
- Anzahl der Verbindungen
  - Je client n-1 Empfangs stream, 1 Sendestream.
  - bei einen Meshbetrieb sind 2 * (n-1) Streams notwendig.
- Die Latenz ist höher als im Mesh-Betrieb.
  - Es gibt immer eine Zwischenstation!
- Die Server Bandbreite und Leistungsfähigkeit muss ausreichend groß sein.
- Simulcast, wenn der Browser es erlaubt, werden 3 Streams mit unterschiedlichen bitraten zur SFU gesendet, die andere Browser erhalten ein der Streams entsprechend der eigener Netzwerkbandbreite.

Die Vorteile des SFU-Betriebs wiegen den Nachteile ab.

4.1.3) MCU

- Trankodierung (alle Clients können bedient werden)
  - Hohe CPU Last
  - Latenz erhöht sich
- Mischung der Streams
  - Clients können nur ein Downloadstream erhalten
    Dies ist von Vorteil wenn die zur Verfügung stehende Bandbreite sehr geign ist.

4.2) SFU und Wirow



Verbindungen und Funktionen bei Wirow

Die meisten Anschlüsse dürften mit IPv6 ausgestattet werden. Damit ist in solche Fällen eine direkte Verbindung zwischen SFU und Client möglich und ein Relay (TURN) erübrigt sich. Dies sollte sich positiv auf der Latenzzeit auswirken.

Die untersuchten Jitsi-Meet server erlaubten leider keine IPv6, im Test-Umfeld war es nicht unbedingt ein Nachteil, ein TURN Server war nicht konfiguriert, dennoch konnten auch Verbindungen zwischen ein IPv6 only und ein IPv4 only client aufgebaut werden. NAT wurde im Heim Router vorgenommen, damit kam es nicht zu zusätzliche Latenz. Falls ein CG-NAT Anschluss vorhanden ist oder die Video-Konferenz über ein Mobil-Funknetz stattfindet, ist eher mit Probleme zu rechnen. Unter Einsatz von IPv6 ist jeweils nur die Firewall zu überwinden, dies stellt kein Problem dar.

5) ICE Server

- Rolle des ICE Servers
  - Ein ICE Server sorgt dafür, dass die beteitligte Systeme Video und Audio Daten zwischen den verschiedene Systeme ausgetaucht werden.
  - Da ein **STUN** Server nicht immer aucreichend ist um eine Verbindung zu ermöglichen, sollte ein **TURN** Server vorhanden sein.

Stun wurde konzipiert um bidirektionalle Kommunikationen zwischen 2 Rechner, in dem da Firewall “durchgelöschert” und das NAT überwunden wird, zu ermöglichen. Dies gelingt nicht immer.

Die Lösung ist den Einsatz einem TURN Server. Falls der Aufbau der Verbindung mittels das STUN Protokoll nicht gelingt, meldet sich der Client an ein Relais und bezieht seine Daten, aktiv vom Relais.

Beim Einsatz von IPv6 Verbindungen, sollte ein STUN-Server ausreichend sein, dies setzt allerdings vorraus, dass das IPv6 Protokoll eine volle Unterstützung seitens des WebRTC Server vorhanden ist und dass alle Systeme eine IPv6 Verbindung aufgebaut haben. Der TURN Server muss zwischen IPv4 und IPv6 Systeme vermitteln.

Seitens Wirow wird für das Übertragen der Audio und Video Daten lediglich IPv4 zur Vergügung gestellt, falls die automatische Konfiguration vorgenommen wird. Der Aufwand

Wirow auf gleichzeitigen IPv4/IPv6 Betrieb umzustellen ist jedoch sehr gering, es müssen lediglich 2 Parameter konfiguriert werden.

5.1) ICE und Relay (TURN)

- Eine Verbindung zwischen 2 Systeme ist nicht immer gewährt
    - Daher ist es zwingend notwendig ein ICE Server, mit der Fähigkeit ein TURN Server zu installieren.

Wenn eine Verbindung zwischen ein oder mehrere Rechner nicht stattfinden kann, werden die Signale an ein TURN-Server gesendet und dieser hat die Aufgabe die Signale an den andere Rechner zu senden. Falls ein Anwender nur mit IPv4 and den anderen mit IPv6 unterwegs sind, erfolgt den Verkehr zwischen den Rechner auf jeden Fall über der TURN Relay. Manchen Mobilfunk Anbieter erschweren das Leben in dem eine direkte Verbindung zu einem weitere WebRTC Partner nicht möglich ist.

Durch das Relaybetrieb entstehen größere Latenzen (Verzögerungen) und die Verständigung kann darunter erheblich leiden. Ein gut erreichbare und nicht von vielen Menschen genutzte TURN Server sollte eingerichtet werden.

5.2) Wie funktioniert STUN mit WebRTC

Die Browser bauen eine Verbindung zu Stun-Server teilen mit welche interne IP-Adressen vorhanden sind und auch die unterstützen Formate. Der Stun Server bewerte die Informationen und liefert einige Informationen zurück, darrunter:

- Externe IP-Adresse deren Erreichbarkeit: host, srflx, prflx. relay
    - Host: Adresse ist öffentlich oder im selben LAN.
    - Srflx: NAT, Ports werden nicht verändert.
    - Rrflx: NAT, Ports werden verändert.
    - Relay: Die Verbindung muss über ein Relay stattfinden. Administratoren. Hersteller oder Anbieter haben eine tolle Arbeit geleistet!

5.2.1) Firewall / NAT überwinden

- A sendet UDP Telegramme vom und zum vereinbarte Ports
- B tut das Gleiche.

Die Firewall und oder das NAT merken, dass eine Verbindung aufgebaut wird und erlauben dementsprechend der Empfang. Da beiden Partner es praktisch durchführen wird spätestens nach dem zweiten Telegramm die Kommunikation aufgebaut.

5.3) WebbRTC und IPv6

- Auf den Clients müssen temporäre IPv6 Adressen vorhanden sein, sonst werden kein IPv6 Kandidaten über das stun Protokoll publiziert.
- Es ist möglich ein eigenen WebRTC Server hinter ein CG-NAT Anschluss zu betreiben, wenn die andere Clients über IPv6 Adressen verfügen, sollte es problemlos funktionieren.

5.4) Verschlüssellung der Übertragung

Dtls ist, für WebRTC, eigentlich Pflicht und wird von Wirow grundsätzlich verwendet, bei Jitsi-Meet muss es offenbar eiongeschaltet werden!

5.5) Wirow Konfiguration

```
[main]
ip = ::
domain_name = meet.example.org
session_cookies_max_age = -1

[servers]
ice_servers = stun:stun.1und1.de:3478 stun:stun.t-online.de:3478

[rtc]
listen_announced_ips =  192.0.2.123 2001:dead:beef:cafe::1

[whiteboard]
public_available = no
```

Dies ist eine minimale Konfigurationsdatei, viel einfacher geht es nicht!

Die verschiedene Konfigurationsparameter sind in einen Beispieldatei beschrieben, einiges ist allerdings nicht optimal.

“expire_session_timeout_sec” und “expire_guest_session_timeout_sec” geben an wie lange eine Sitzung dauern darf.

“room_data_ttl_sec” gilt für das Whiteboard und bezeichnet die maqximaler Dauer der zwischengespreicherte Daten des Whiteboard. Ein Satz wird nach dem Schließen des “Whiteboard-Tabs” angelegt und wird nach der konfigurierte Zeit gelöscht. Wenn die Zeit zu kurz ist, besteht die Gefahr ein leeres Whiteboard zu finden.

5.6) Übertragungsrate

5.6.1) Übertragungsrate Audio

Codec	**Bitrate kbps**	**Bemerkung**
Opus	6 bis 500	Adaptiv

5.6.2) Übertragungsrate Video

Codec	**Bitrate Mbps**
VP8	0,1 - 2

Auflösung / Fps	**Bitrate Mbps**
720 / 30	1 - 2
360 / 30	0.5 - 1
180 / 30	0.1 - 0.5

Die Qualität der Videobilder ist abhängig von der zur Verfügung stehende Bandbreite und der Einstellungen. Jitsi erlaubt es die Video in 4 Stufe zu übertragen, von keine Video bis maximale Qualität und größte Bandbreite. Ein DAU dürfte aber mit der Einstellung überfordert sein. Die vom Browser ausgewählte Auflösung basiert auf der ermittelte Bandbreite. Dies erfolgt nach ein “congestion” Protokoll.

5.6.2.1) Übertragungsrate und Videoformat

WebRTC Applikationen stellen üblicherweise die Kontrahenten in ein 4:3 Format dar.

Mögliche Formate	Std	C1	C2	C3	C4
960*720	*	*			
800*600		*			
640*480		*	*	*	*
480*360	*	*			
320*240		*	*		*
240*180	*				
160*120		*	*		

Die Tabelle zeigt die im 4:3 angebotenen Formate für einer USB Webcam Logitech C922 (C1) und die von 3 unterschiedliche Notebooks.

Mit das Kommando *4l2-ctl –list-formats-ext –device /dev/video**X*** kann die Liste der unterstütze Formate der Webcam gelistet werden. Damit, das richtige Format vom Browser gewählt wird, kann **guvcview** verwendet werden.

Std gibt die für WebRTC angegebenen bevorzugte Formate an, diese können möglicherweise vom Browser herunterskaliert werden (Simulcast mit VP8).

Formate wie beispielsweise 16:9 verschwenden nur Bandbreite und mit Rücksicht auf Teilnehmern, die nicht eine gute Internetverbindung haben, nicht genutzt werden.

5.7) Was kann erwartet werden

- **CPU**
  Virtual Server 2 VCore Prozessoren, 2 GB RAM.
  CPU Verbrauch 1 Client < 02 % (1,3)
  CPU Verbrauch 2 Client < 04 % (2,6 - 3,7)
  CPU Verbrauch 3 Client < 07 % (5,3 - 6,7)
  CPU Verbrauch 4 Client < 09 % (7,3 - 8,6)
  CPU Verbrauch 5 Client < 10 % (7,0 - 9,7)
- **Bandbreite**
  Bis zu 400 Mbps.
  Bandbreite je Client, ca 300 KBps - 2,4 KBps.
  Dies scheint der limitierende Faktor zu sein (11 - 30 Clients).
- RAM der Verbrauch an RAM war sehr gering.

Bei 11 Client mit jeweils Streams a 2,4 KBps würden wir ein gesamten Verkehr von rund 300 MBps, es bedeutet einem ausreichenden Abstand zur maximale Geschwindigkeit. Bei eine geringere Bandbreite des Systems von 300Kbs könnte über 30 Clients bedient werden.

Der Hersteller gibt als kleinste Voraussetzung 2 vCPU und 4GB RAM. Der Testserver hatte nur 2 GB RAM. Eine Begrenzung durch RAM Kapazität konnte nicht festgestellt werden, es lief praktisch kein weiteren Dienst, dafür waren die dynamischen Bibliotheken der Distribution, die zur Kompilierungszeit verwendet wurden (Fedora 36) installiert, so dass die vom Debianserver verwendete *.so Dateien nicht verwendet wurden.

6) Administration

- Rolle Admin:
  - Neue Anwender eintragen.
  - Räume anlegen.
  - Meeting starten.
  - An Meeting für den sie ein Link haben, teilnehmen.
- Rolle User:
  - Räume anlegen.
  - Meeting starten.
  - An Meeting für den sie ein Link haben, teilnehmen.
- Nicht eingetragenen Anwender:
  - An Meeting für den sie ein Link haben, teilnehmen.

Räume die von Anwender angelegt worden sind für das ersten privat, die andere Anwender haben kein Zutritt und sehen nicht diese Räume.

Wurde ein Anwender zu einem Meeting, der von ein anderen Anwender angelegt wurde, können sie, nachdem sie das Meeting besucht haben, das Meeting sehen und auch starten.

7) Tests Wirow und Jitsi-Meet - Aufbau

- Server
  - Wirow: eigene vServer (ionos)
  - Jitsi.meet: conference.ztl.team
- Clients
  - Notebook 1 im Heim-Lan angemeldet
  - Notebook 2 über Freifunk angemeldet.

7.1) Tests Wirow und Jitsi-Meet - Ergebnisse

- Die Verbindung über Wirow lief ein wenig besser, auf beide PC wurden die Videostreams übertragen. Mit Jisi-Meet war am Rechner der über Freifunk angemeldet das Bild vom “Heim Notebook” nicht immer zu sehen!
- Bei Verbindungsabbruch fordert Jitsi-Meet, sofern ein Passwort gesetzt ist, erneut die Eingabe der Logindaten.
  Wirow ist hier gutmütiger das Neuladen der Fenster führt direkt zum Meeting,
  - Das automatisches Wiederverbinden von Wirow, mit den alte Daten kann manchmal zu Nebeneffekte führen.

8) Audio und Videoformate.

- Standardmäßig werden nur **Opus** und **VP8** unterstützt. Siehe auch: https://developer.mozilla.org/en-US/docs/Web/Media/Formats/WebRTC_codecs
- Als Audio Codec ist Opus ein muss und hat wahrscheinlich die optimale Eigenschaften (Qualität: Telefon bis Hi-FI unf nahtlose Anpassung der Banddbreite)
- **H264** und **VP9** können auch unterstützt werden, dies muss allerdings konfiguriert werden. Dabei ist zu beachten, dass manche Client möglicherweise der Dienst verweigern.
- **AV1** wird weder von Jisti-meet noch von Wirow unterstützt (zu neu, auch keine Implementierung für Safari).

Vorteil von **VP9** und noch mehr von **AV1** ist die geringere notwendige Bandbreite, dafür sind die Anforderungen an der Hardware größer. Cisco gibt, für Webex mit **AV1**, dass der Client minimal 4 CPU-Kerne besitzen muss. **VP9** muss auch explizit in Firefox konfiguriert werden (auch bevorzugte Codec).

9) Browser und WebRTC

- Alle Browser erlauben **VP8**
- **Firefox**: **VP9** per Default abgeschaltet
- **Safari**: **VP9** wird nicht unterstützt
- **Blink** basierenden Browser (Google-Chrome, Chromium, Edge, Opera, Vivaldi) sollten alle Formate unterstüzen.
- **H264** (MP4) wird unter Android nicht unterstützt.
- Manche **Android** Geräte sind durch **VP9** überfordert.

10) Hintergrund Veränderung

> Es gibt Möglichkeiten, für Wirow, mit externe Programme das Hintergrund zu verändern.

Falls Wirow mit eine Ersetzung/Verarbeitung des Hintergrunds versehen soll kann es beispielsweise mit backscrub und das Kernel Module v4l2loopback realisiert werden.

Die nötige Prozessorleistung ist allerdings hoch. Auf ein ältere Notebook mit **Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz** (4 Kerne) steigt die Processorauslastung von ca. 2 % auf bis 30 % (nur backscrub). Auf ein modernere Notebook mit **11th Gen Intel(R) Core(TM) i9-11950H @ 2.60GH** (16 Kerne) steigt die CPU-Leistung um lediglich um ca. 5 %.

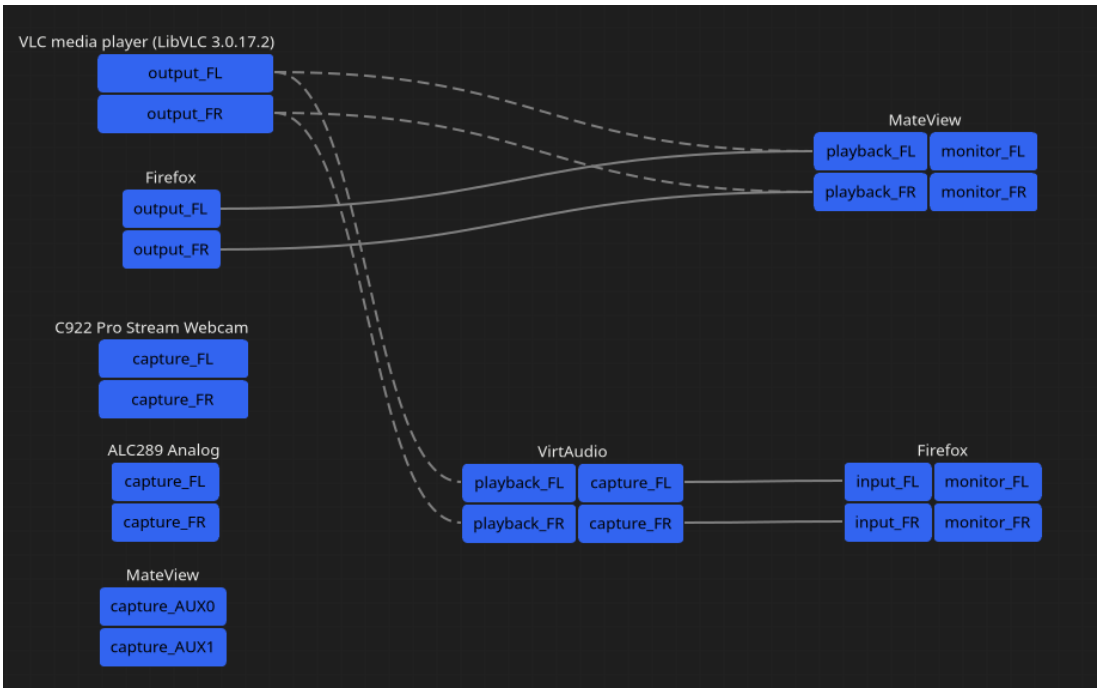
Auf das ältere Notebook ist das Leistungsvierbauch beim Einsetzen der Jitsi-meet Hintergrundveränderung ähnlich wie unter Einsatz von Wirow und backscrub.

11) Aufnahme der Konferenz

- **Datenschutz und Persönlichkeitrecht beachten**
  - Alle Teilnehmer müssen einverstanden sein!
- Wirow bietet für angemeldete Anwender solch eine Funktion
- Jitsi-meet möchte gerne die Daten bei Dropbox unterbringen.
- Alternative:
  - Chromium basierten Browser Extension **RecordRTC**
  - **vokoscreenNG** aus der jeweilige Repo, flexiblere Lösung!

12) Video und Audio übertragen

> Eine Übertragung von Video und Audio ist nicht vorgesehen, unter Linux kann es relativ leicht nachgerüstet werden.



Pipewire Konfiguration

12.1) Audio Konfiguration

```
pw-cli create-node adapter \
    '{ factory.name=support.null-audio-sink node.name=VirtAudio
        media.class=Audio/Duplex
        object.linger=true
        audio.position=[FL FR] }'
```

Mit das Kommando **pw-cli** wird ein Object vom Typ Audio/Duplex angelegt, der Name ist in dem Fall **VirtAudio**. Das Object hat Eingängen die mit ein Mikrofon oder, in dem Fall, mit den Ausgang vom Mediaspieler VLC verbunden werden kann.

Unter Fedora XFCE ist damit die Verbindung VirtAudio mit der Eingang von Firefox verbunden wird mit der Lautstärke Einstellung des Mikrofons, VirtAudio zu wählen.

12.2) Umstellen der Audioeingang

```
connectVLCToVirtAudio() {
    list=$(pw-cli ls | grep alias | grep "VLC")
    name=$(echo $list |sed 's/.*= "\(.*\):.*/\1/')
    pw-link $1 "$name":output_FR VirtAudio:playback_FR
    pw-link $1 "$name":output_FL VirtAudio:playback_FL
}
```

Ein Aufruf der Funktion connectVLCToVirtAudio erlaubt es die Verbindung (Link) zu erzeugen oder zu entfernen, falls das Parameter -d eingegeben wurde.

Für die Mikrofone reicht es der Name des Mikrofone, beispielsweise für Mateview nachstehende Funktion auszuführen:

```
connectMateviewToVirtAudio() {
    pw-link $1 MateView:capture_AUX0  VirtAudio:playback_FL
    pw-link $1 MateView:capture_AUX1  VirtAudio:playback_FR
}
```

Diese Lösung ist nicht benutzerfreundlich, erlaubt es in Ausnahmefälle ein Video zu Teilen.

Unter Windows oder MacOS sollten ähnliche Funktionalitäten vorhanden sein.

12.3) Umstellen der Videoeingang

Es reicht aus das entsprechende Fenster, in dem Fall VLC zu teilen.

- Mit die Tastenkombination [Strg]+[h] können Menü und Steuerungselemente ausgeblendet werden.
- Unter Zuhilfenahme von **devilspie2** (Siehe Paketmanagement) kann die Windowsdekoration eliminiert und auch die Größe des VLC Fenster gesetzt werden.