HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

swisscom

Bachelorarbeit, Abteilung Informatik

VoIP-Qualitätsparameter
Mouth-to-Ear Delay

Hochschule für Technik Rapperswil

Frühlingssemester 2016
17. Juni 2016

Autoren:	Max Obrist & Pascal Meier
Betreuer:	Prof. Dr. sc. techn. Peter Heinzmann
Co-Referent:	Prof. Oliver Augenstein
Experte:	Dr. Th. Siegenthaler, CSI Consulting AG
Industriepartner:	Swisscom
Arbeitsperiode:	22.02.2016 - 17.06.2016
Arbeitsumfang:	360 Stunden (12 ECTS) pro Student
Link:	`https://bitbucket.org/ba_voip_p2p/ba_voip_p2p`

Abstract

Die analoge Telefonie hat langsam aber sicher ausgedient. Die Swisscom sowie viele andere Provider, im In- und Ausland, planen in der nahen Zukunft den kompletten Umstieg auf All-IP. Dabei wird das gesamte Telefongespräch über das Internet abgewickelt.

Dieser Umstand bringt neben vielen Vorzügen auch einige neue Herausforderungen mit sich. Die Umstellung auf eine IP-basierte Telefonie macht die Kommunikation anfälliger für Transport- und Netzwerkstörungen.

Im Rahmen dieser Arbeit wurde analysiert, welche Umstände zu Qualitätseinbussen führen. Speziell wurde der Mouth-To-Ear (M2E) Delay analysiert. Neben fixen Delays die auf Grund von Codecs oder Netzwerkgegebenheiten auftreten, gibt es im Gesamtsystem einige dynamische Elemente, die nicht so einfach berechnet werden konnten.

Mithilfe von Messungen mit Mikrofon und Aufnahmegerät wurde im eignen Testlabor mittels SIP-Server umfangreiche Tests durchgeführt. Dies führte zum Ergebnis, dass die stark abweichenden M2E-Delays aufgrund von unterschiedlich implementieren Jitter-Buffer zu Stande kommen.

Um aussagekräftigere und raschere Qualitätsangaben zu machen, wurde der bekannte VoIP-Client *Jitsi* so angepasst, dass sich Echtzeitdaten über die wichtigsten Parameter sammeln lassen. Auf einen Schlag ist ersichtlich, wie gross beispielsweise der momentane Jitter-Buffer ausfällt.

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

Bachelorarbeit
VoIP-Qualitätsparameter – Mouth-to-Ear Delay

Studiengang:	Informatik (I)
Institut:	ITA: Internet-Techn. und Anwendungen
Gruppe:	Pascal Meier, Max Obrist
Betreuer:	Heinzmann, Peter
Koreferent:	Prof. O. Augenstein, HSR
Experte:	Dr. Th. Siegenthaler, CSI
Industriepartner:	Swisscom, Reto Hauri

Ausgangslage

Die heute im Internet noch dominierenden Client-Server Konzepte könnten in den nächsten Jahren durch Peer-to-Peer Konzepte abgelöst werden. Bereits wird Peer-to-Peer als das «Internet der Zukunft» gehandelt, weil es gegenüber den Client-Server Konzepten bezüglich Sicherheit, Verfügbarkeit, Leistungsfähigkeit und Datenschutz wesentliche Vorteile bringt. Peer-to-Peer Anwendungen gibt es für File Sharing, Information Searching, Social Networks und Voice-over-IP (VoIP). Letzterem gilt besondere Beachtung, weil diverse Telekomanbieter ihre Sprachdienste bis 2017 vollständig auf VoIP umstellen wollen. Besonders interessant sind auch Ansätze wie TOR Fone, bei welchem, sichere, anonyme Voice Kommunikation über TOR Netze ermöglicht werden soll.

Ziel

Im Rahmen dieser Studienarbeit sind Verfahren zu entwickeln, welche die Bestimmung der Dienstqualität bei VoIP erlauben. Insbesondere soll genau aufgezeigt werden, welche Faktoren den Mouth-to-Ear (MTE) Delay bestimmen. Der MTE-Delay soll für verschiedene VoIP-Dienste und VoIP-Clients (inklusive TOR Fone) bestimmt und verglichen werden. Aus den gewonnenen Erkenntnissen sind Empfehlungen für die Wahl von VoIP-Diensten in verschiedenen Netzwerkumgebungen abzuleiten.

Aufgaben

1. Einarbeitung Theorie
   - Verständnis der Grundkonzepte bei der VoIP-Kommunikation.
   - Verständnis der Prinzipien zum Ersatz von analogen Telefonie Anschlüssen durch All-IP-Anschlüsse.
   - Verständnis der Verfahren zur Bestimmung der Qualität bei VoIP.
   - Erstellung einer Übersicht zum aktuellen Stand im Bereich P2P.

- Umfeldanalyse, Literaturrecherche in Zusammenarbeit mit HSR-Bibliotheksdienst

2. Analyse, Praxis (VoIP-QoS Messungen)
   - Aufbau eines lokalen VoIP-Testplatzes (mit Asterisk und verschiedenen VoIP-Clients)
   - Durchführung von Messungen zur Bestimmung wichtiger VoIP-Qualitätsparameter bei verschiedenen VoIP-Diensten:
     i. Verzögerung bei verschiedenen Netzen und Geräten (Mobile, Wireline und Satelliten Verbindungen) – Delay, Jitter, Mouth-to-Ear-Delay
     ii. Netzwerkemulation zur Bestimmung des Einflusses von Verzögerung und Packet Loss
     iii. Untersuchung an ATA-Schnittstellen (Analog-Telephone Adapter)
   - Inbetriebnahme von TOR Fone und von anderen VoIP-Diensten.
   - Aufzeigen der Auswirkungen verschiedener Leistungsparameter (Antwortzeiten, Datenraten, Paketverlust) auf die Auslieferung von VoIP (Beispiele: Skype, Swisscom VoIP, upc VoIP, TOR Fone)
3. Realisierung
   - Entwicklung eines Systems zur Anzeige von Qualitätsparametern bei VoIP
4. Vergleichsmessungen mit ausgewählten VoIP-Systemen
   - Erklärung von Unterschieden
   - Qualitätsbeurteilung
   - Verbesserungsvorschläge

Referenzen

1. Hinweise zur Durchführung von Studienarbeiten: https://dl.dropboxusercontent.com/u/4679041/SABA-Web-Anleitungen_Heinzmann.zip
2. Peer to Peer, Das neue Internet, Die Zeit Online, 31.1.2012 http://www.zeit.de/zeit-wissen/2012/05/Das-alternative-Netz
3. Michel Bauwens, Gründer der Stiftung P2P http://p2pfoundation.net/Michel_Bauwens , http://p2pfoundation.net/images/255426317-What-Does-the-P2P-Foundation-Do-2015.pdf , https://www.diigo.com/user/mbauwens/Bauwens-Lectures
4. TORFone - voice add-on for TorChat, http://torfone.org/
5. Richard Jobson, «Methods to Objectively Evaluate Speech Quality», Overview Paper, Teraquant Corporation. http://teraquant.com/sites/default/files/whitepapers/perceptual_speech_quality_measurements.pdf
6. Irina Cotanis, «Understanding the Transition From PESQ to POLQA», Ascom Network Testing White Paper, 2011. http://www.ascom.com/en/understanding-the-transition-from-pesq-to-polqa.pdf
7. Irina Cotanis, «Voice Service Quality Evaluation Techniques and the New Technology POLQA»; Ascom Network Testing White Paper, 3.11.2010. http://www.ascom.com/en/tems-voice-service-quality-evaluation-techniques-and-polqa.pdf
8. Perceptual Evaluation of Speech Quality (PESQ,) https://en.wikipedia.org/wiki/PESQ
9. Analog Telephone Adapter (ATA), https://en.wikipedia.org/wiki/Analog_telephone_adapter
10. Colin Perkings , "RTP: Audio and Video for the Internet", Addison Wesley, 2003 ISBN:0-672-32249-8. http://read.pudn.com/downloads121/ebook/515800/RTP.pdf

V2.0,
Rapperswil, 12.4.2016

Prof. Dr. P. Heinzmann

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Management Summary

1 Ausgangslage

Die Festnetztelefonie befindet sich in einem starken Wandel. Die konventionelle Telefonbuchse für die Analogtelefonie hat langsam aber sicher ausgedient und wird von der IP-Technologie abgelöst. Die Swisscom sowie viele andere Provider, im In- und Ausland, planen in der näheren Zukunft den kompletten Umstieg auf All-IP. Dabei wird das Telefongespräch über das gewöhnliche Internet abgewickelt.

Die meisten Kunden der Swisscom können ohne weitere Probleme auf All-IP umgestellt werden, da sie bereits heute mit einem ausreichenden Internetzugang ausgestattet sind. Einzelne Kunden können jedoch nicht problemlos auf neue Internettechnologien umgestellt werden. Dies betrifft z.B. Spezialanwendungen wie Lifttelefone – diese dürfen z.B. bei einem Stromausfall nicht ausfallen – oder sogenannte *Long Liner* Kunden, welche zu weit vom nächsten Verteiler wohnen, wodurch die minimale Datenrate nicht erreicht werden können.

In solchen Fällen gibt es zwei Möglichkeiten. Sofern die Abdeckung mit mobilem Internet ausreichend ist, können sogenannte Wireless Home Connection (WHC) Router eingesetzt werden. In diesem Fall wird das Internet über das normale Mobilfunknetz übertragen, und damit auch die Voice over IP Telefonie. Sofern auch die mobile Abdeckung nicht ausreicht, können Internetverbindungen auch mittels Satellitenverbindungen sichergestellt werden. Damit wird es zwar möglich, die Schweiz flächendeckend mit Internet zu versorgen, bietet aber im Bereich der IP-Telefonie gewisse Nachteile, da bei Verbindungen über Satellit relativ hohe zusätzliche Verzögerungen von rund 360 ms zwischen Sender und Empfänger entstehen. Diese können einen Einfluss auf die wahrgenommene Qualität von Telefongesprächen haben.

Swisscom hat in Zusammenarbeit mit cnlab bereits erste Messungen bezüglich der Verzögerung vom Mikrofon des Senders zum Lautsprecher des Empfängers durchgeführt, dem sogenannten Mouth-To-Ear-Delay. Dabei fiel auf, dass relativ unerwartete Faktoren einen grossen Einfluss auf die gemessenen Verzögerungen haben. So kommen z.B. bereits durch diverse Analog-Digital-Wandlungen sehr unterschiedliche Verzögerungen zu Stande, aber auch das verwendete System und

der zum Einsatz kommende Codec können einen messbaren Einfluss haben.

Im Rahmen dieser Arbeit werden deshalb verschiedene Faktoren analysiert, die einen Einfluss auf die gemessenen M2E-Verzögerungen haben können. Diese sollen möglichst genau verstanden werden und es sollen Empfehlungen erarbeitet werden, welche für einen stabilen und möglichst verzögerungsarmen Betrieb eines VoIP Netzwerkes eingehalten werden sollten.

2 Vorgehen/Technologien

In einem ersten Schritt wurde das theoretische Wissen erarbeitet, welches benötigt wurde für ein genaues Verständnis von VoIP-Netzwerken allgemein, den damit zusammenhängenden Protokollen SIP, RTP und RTCP sowie den verwendeten Audiocodes der ITU-T G-Series.

Mit dem so erarbeiteten Wissen wurde ein einfaches Testlabor aufgebaut, um der Frage nachzugehen, wie M2E-Delay entsteht und von welchen Faktoren dieser abhängt. Es wurde das Private Branch Exchange (PBX) System Asterisk PBX eingesetzt, eine Software zur Steuerung und Vermittlung von VoIP-Anrufen, um eine eigene Telefonanlage betreiben zu können. So konnten sich verschiedene Clients an dieser Telefonanlage registrieren und es konnten Telefonverbindungen zwischen verschiedenen Clients erzeugt werden. Mithilfe eines WAN-Emulators wurden des weiteren verschiedene Netzwerkparameter zwischen den verbundenen Clients emuliert, z.B. Verzögerungen, Paketverluste oder Jitter.

Mit diesem Versuchslabor konnten umfangreiche Tests des M2E-Delays durchgeführt werden. Für diese Mouth-To-Ear (M2E) Messungen wurde ein relativ einfaches Verfahren eingesetzt, welches aber in der Praxis häufig eingesetzt wird und in Abbildung 1 skizziert ist.

Schrittweise wurden nun verschiedene Elemente aus der Theorie in der Praxis abgebildet und auf ihre Plausibilität verifiziert, wobei immer wieder Messungen durchgeführt wurden, um das Verhalten zu analysieren.

In einem weiteren Schritt wurde basierend auf Jitsi, einem Open-Source VoIP Client, ein Tool entwickelt. Dieses erlaubt es, Telefonstatistiken in Echtzeit darzustellen und auszuwerten. Jitsi wurde dabei modifiziert, um wichtige Leistungsparameter, die von Jitsi gesammelt werden, in Echtzeit darzustellen. Damit kann der Administrator eines VoIP-Netzwerks Probleme im Netzwerk nachvollziehen und so allfällige Probleme analysieren und beheben.

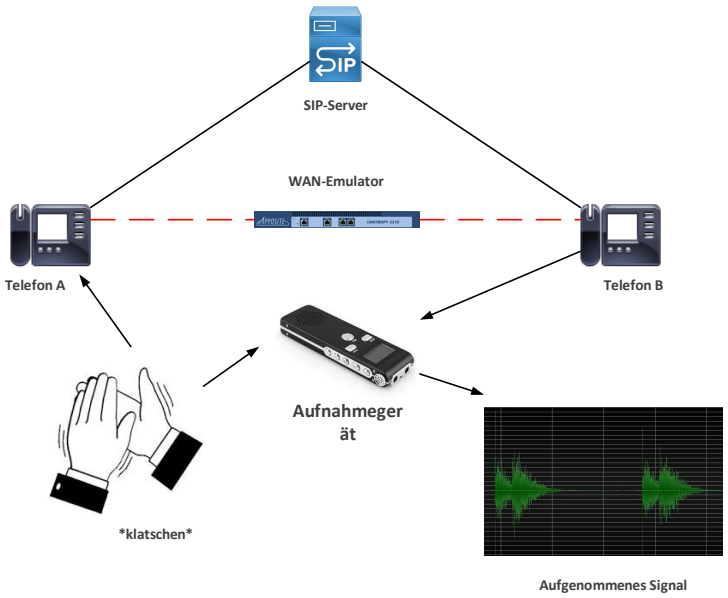


Abbildung 1: Vorgehen für M2E-Messungen

3 Ergebnisse

Es ist erstaunlich, wie viele Faktoren die Qualität und die wahrgenommene Verzögerung in einem Telefongespräch beeinflussen.

Sowohl Einbussen in der Qualität sowie ein zunehmender M2E-Delay können auf drei Kategorien zurückgeführt werden. Für das Klangerlebnis ist der verwendete Audio-Codec der entscheidende Faktor. Dieser bestimmt die Abtastrate eines analogen Signales und damit, wie exakt die menschliche Stimme digitalisiert wird. Dabei handelt es sich immer um einen Kompromiss zwischen der gehörten Qualität eines Audiosignals, sowie der Datenrate, welche für die digitalisierte Version des Audiosignales benötigt wird.

Ein weiterer Faktor ist die Geschwindigkeit der Internetanbindung sowie die Netzwerkkonfiguration des Providers. Nur durch Priorisierung von Echtzeitanwendungen, besonders der Internettelefonie, können spürbare Schwankungen der auftretenden Verzögerungen minimiert werden, welche aufgrund eines in den meisten Clients verwendeten Jitter-Buffers zu weiteren Verzögerungen führen können. Der Jitter-Buffer ist dafür verantwortlich, Schwankungen der eintreffenden Signale auszugleichen und sorgt damit für eine gleichmässige Tonwiedergabe.

Zu guter Letzt hat auch der eingesetzte VoIP-Client einen massgeblichen Einfluss auf den gemessenen M2E-Delay. Je nach Implementation der Endanwendung können sehr unterschiedliche Verzögerungen gemessenen werden. Dies hängt mit

der Effizienz der Implementation zusammen, mit der Art und Weise, wie der Jitter-Buffer implementiert ist und auch der Hardware, auf welcher der Client läuft. Einer ausführlichen Evaluation des eingesetzten Clients kommt damit eine grosse Bedeutung zu.

4 Ausblick

Die erarbeiteten Erkenntnisse können bei der Einrichtung von neuen VoIP-Netzen oder beim Evaluieren neuer Clients verwendet werden, um diese von Beginn weg auf einen möglichst kleinen M2E-Delay zu trimmen. Mit dem erstellten Tool für Echtzeitanalysen können bestehende VoIP-Netze analysiert werden, um herauszufinden, weshalb in einem Netz bestimmte Verzögerungen entstehen.

Der Zeitaufwand um verschiedene Messungen durchzuführen, wurde zu Beginn der Arbeit etwas unterschätzt. Aufgrund des eingeschränkten Zeitbudgets konnten deshalb nicht alle Messungen durchgeführt werden, die für eine vollständige Analyse nötig wären. So wurden nur Messungen mit dem G.711 Codec durchgeführt, wobei heutzutage auch diverse andere Codecs der ITU-T G-Series eingesetzt werden, wie G.726 und G.729. Um das Verhalten dieser Codecs bei eingeschränkten Bandbreiten oder grossem Jitter genauer zu verstehen, sollten deshalb ähnliche Messungen, wie sie im Verlauf der Arbeit gezeigt werden, mit den erwähnten Codecs durchgeführt werden.

Auch wurden verschiedene Netzwerkparameter ausschliesslich mit dem eingesetzten WAN-Emulator getestet. Dies erlaubt zwar ein feingranulares Testen, schliesst aber allfällige Einflüsse von bestimmten Netzwerken aus. Es sollten deshalb weitere Messungen durchgeführt werden, die über Satellitenverbindungen gehen oder bei denen die VoIP-Verbindung über das Mobilfunknetz übertragen werden, um spezifische Eigenschaften dieser Netzwerke zu analysieren.

Bericht

1 Einleitung

Die Swisscom ist als Grundversorgungskonzessionärin durch das Bundesamt für Kommunikation (BAKOM) dazu verpflichtet, die Grundversorgung der Schweizer Bevölkerung mit Telefonie und Internet sicherzustellen.[1]

Die Swisscom will in nächster Zeit alle ihre Netze auf All-IP umstellen. Dies bedeutet, dass sämtliche von der Swisscom angebotenen Dienste über ein IP-Netz angeboten werden. Dieser Schritt soll bis 2018 vollzogen werden.[2] Darunter fällt auch die erwähnte Telefonie, weshalb der Telefonzugang aller Swisscom-Kunden auf Voice over IP (VoIP) umgestellt werden soll.

Ein Grossteil der Kunden ist bereits heute im Besitz eines IP-fähigen Telefons. Solche Kunden dürften von der Umstellung der Swisscom nicht viel mitbekommen oder sind bereits jetzt mittels VoIP angeschlossen.

Die meisten Kunden, welche nicht im Besitz eines VoIP-fähigen Telefons sind, erhalten einen Router mit einem integrierten Session Initiation Protocol (SIP)-Client. Dadurch können diese Kunden ihr Analogtelefon am Analog Telephone Adapter (ATA) Port des Routers anschliessen.

Es gibt aber auch Telefone, die bestimmte Möglichkeiten der analogen Telefonanschlüsse nutzen, namentlich die integrierte Stromversorgung, welche nach der VoIP-Umstellung nicht mehr oder nur in beschränktem Umfang verfügbar sein werden. Vor allem Geräte wie Lifttelefone, Alarmanlagen o.ä. nutzen die Stromversorgung der Analogleitung. Diese integrierte Stromversorgung ist in einem All-IP Netz nicht mehr möglich.

Ist eine integrierte Stromversorgung notwendig, müssen andere Wege gefunden werden, die Funktionsfähigkeit des Gerätes im Ernstfall sicherzustellen. Diese Anpassungen werden jedoch nicht von der Swisscom durchgeführt, sondern durch den jeweiligen Geräteanbieter.

Für die bisher beschriebenen Kundenkategorien gibt es Lösungen, welche einen weiteren Betrieb des Telefonsystems ermöglichen – unter Umständen mit einigen Anpassungen.

Es gibt aber noch eine weitere Kategorie Kunden – die sogenannten *Long Liner*.

Vorteile	Nachteile
• Bessere Sprachqualität (HD) • Vereinfachte Verwaltung – Anrufweiterleitung – Sperren von Nummern • Zusätzliche Funktionen – Telefonnummer mit Telefonbuch abgleichen • Mehrere Rufnummern nutzbar • Grundgebühr fällt weg • Alternative: Skype / SIP-Provider	• Telefonanbindung benötigt Router/Modem • Impulswahlverfahren nicht mehr unterstützt • ISDN wird nicht mehr unterstützt • Anfälliger für Transport- und Netzwerkstörungen • Keine Stromversorgung via Kupferkabel

Tabelle 1: Eine Übersicht über die Vor- und Nachteile von VoIP und in All-IP Netzen

Etwa 7000 bis 10 000 Kunden fallen bei der Swisscom in diese Kategorie, siehe dazu im Appendix L.1. Diese Kunden können mit dem bestehenden Leitungsnetz nicht mit der minimalen geforderten Datenrate von 2000/200 kbit/s versorgt werden.[1] Gemäss Aussagen von Swisscom, sollen rund 2/3 der Kunden (also 4600 bis 6600) mittels einem Wireless Home Connection (WHC) Router über das Mobilnetz angeschlossen werden.

Der letzte Drittel der Kunden (2400 bis 3400 Kunden) sollen mittels Satellit angeschlossen werden. Damit wird die benötige Datenrate von 200 kbit/s in beide Übertragungsrichtungen erreicht, allerdings beträgt die Round Trip Time (RTT) zu einem Kommunikationspartner via einem Satelliten im geostationären Erdorbit rund 720 ms, wie in Abschnitt A berechnet. Diese relativ hohe RTT beeinträchtigt die Kommunikationsqualität.

Swisscom und das cnlab haben die Verzögerung vom Mikrophon des Senders (Mouth Schallevent) zum Lautsprecher des Empfängers (Ear Schallevent), den sogenannten Mouth-To-Ear-Delay (M2E) für verschiedene Netze und Konfigurationen gemessen. Dabei fiel auf, dass alleine durch die diversen Analog-Digital-Wandlungen auch im heute verwendeten Telefonnetz M2E-Delays von ca. 100 ms bis 300 ms zu beobachten sind. Während diesen Messungen wurden auch Unterschiede zwischen verschiedenen VoIP-Clients gefunden.[9] So beobachtet man z.B. bei einem Anruf, bei dem die Basisstation des Anrufers mittels dem von Swisscom vertriebenen WHC-Router HuaweiB315/ATA angeschlossen ist und der Anruf über das SCSIP System übertragen wird, je nach Empfänger sehr unterschiedliche M2E-Delays. Ist der Empfänger in Festnetz zuhause, entstehen dabei Delays von durchschnittlich

93 ms, während bei einem Empfänger im Mobilfunknetz (iPhone 6s, 3G) Delays von durchschnittlich 183 ms gemessen werden.

Im Rahmen dieser Arbeit sollen die verschiedenen Faktoren aufgezeigt werden, welche die M2E-Delays bestimmen. Der Einfluss von Protokollen, Kompressionsalgorithmen, Digitalisierungsverfahren, Datenraten, Verschlüsselungsalgorithmen, Netzbelastungen etc. soll theoretisch beschrieben und messtechnisch verifiziert werden.

Dazu wird ein WAN-Emulator eingesetzt, der es erlaubt, verschiedene Einflüsse, die in einem Netzwerk erscheinen können, zu emulieren. Darunter fallen z.B. Schwankungen in der Bandbreite, Verzögerungen im Netzwerk, Paketverluste oder Jitter. Letztendlich unterliegt die Kontrolle über den grössten Teil des Netzwerkes aber dem Provider. So kann auf einen grossen Teil der VoIP-Kommunikation keinen Einfluss genommen werden.

Aus diesen Analysen und Messungen sollen Empfehlungen abgeleitet werden, worauf bei der Wahl eines VoIP-Produktes zu achten ist und welche Optimierungen mittels der Konfiguration des Netzwerkes möglich sind.

2 Theorie

Führen zwei Teilnehmer ein Telefongespräch über ein IP-Netzwerk, wird das Gespräch in mehreren Schritten von einem akustischen Signal (Mouth-Signal) in ein IP-Paket überführt, bevor es auf dem Empfangssystem wieder in ein analoges Audiosignal zurückübersetzt wird, wie es in Abbildung 2 dargestellt ist.

Nachdem das Audiosignal beim Telefon des Senders A vom Mikrofon aufgenommen wurde, muss es mittels einem Analog-Digital-Wandler in digitale Audiosamples gewandelt werden. Dies geschieht meist mit einem an Pulse Code Modulation (PCM) angelehnten Verfahren.

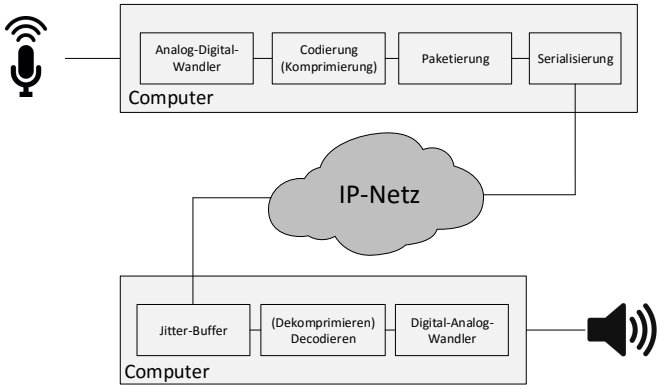


Abbildung 2: Basic Voice Flow

Nach der Codierung wird bei einigen Algorithmen zusätzlich eine Komprimierung der Samples durchgeführt. Diese Daten werden nun paketiert und auf ein Netzwerkinterface serialisiert, bevor sie mittels LAN/WAN zum Empfänger B übermittelt werden, wobei die Pakete auf den verschiedenen Netzwerk-Hops (z.B. Router) mehrmals neu geroutet werden müssen.

Beim Empfänger müssen in die erhaltenen Frames wieder dekomprimiert werden und mittels einem Digital-Analog-Wandler wieder in analoge Audiosignale umgewandelt werden, bevor das Audiosignal einem Lautsprecher interpretiert und in Schallwellen umgewandelt werden kann.

Bei all diesen Zwischenstationen können diverse Qualitätsprobleme auftreten, die sich grundsätzlich in zwei Kategorien aufteilen lassen. Einerseits benötigt jeder Schritt eine gewisse Zeit zur Durchführung. Selbst die reine Übertragung der Daten benötigt Zeit, was insbesondere bei Satellitenverbindungen zum Tragen kommt. Am Ende benötigt ein Tonsignal eine gewisse Zeit, um vom Mund des Senders bis zum Ohr des Empfängers zu gelangen, was man den Mouth-To-Ear (M2E) Delay nennt.

Andererseits führen die verschiedenen Schritte auch zu Einbussen in der Sprachqualität. Das PCM Verfahren führt zum Beispiel zu einem Prinzip bedingten Qualitätsverlust, da es nicht unbegrenzt genau ist. Nur Frequenzen, die halb so gross sind, wie die PCM-Abtrastrate, können auch wiederhergestellt werden.[5] Auch bei der (verlustbehafteten) Kompression sowie der Digital-Analog-Wandlung können Qualitätseinbussen in der Tonqualität auftreten.

Gemeinsam kombinieren sich diese beiden Faktoren zu einer allgemeinen Gesprächsqualität. Da die Wahrnehmung der Qualität eines Gespräches äusserst subjektiv ist, hat man besonders im historischen Kontext meist sogenannte Mean Opinion Score (MOS) Messungen durchgeführt.[6] Dabei werden Nutzer nach der wahrgenommenen Gesprächsqualität auf einer Skala zwischen 1 (am schlechtesten) und 5 (am besten) befragt. Der gemittelte Wert entspricht dann dem Mean Opinion Score (MOS).

Der Nachteil dieser Methode liegt in der Subjektivität der Messresultate. In VoIP gibt es jedoch Möglichkeiten, sich auf objektive Messungen zu stützen. Die beiden heute gebräuchlichsten Messmethoden sind das E-Model und Perceptual evaluation of speech quality (PESQ). Auf deren Funktionsweise wird im Rahmen dieser Arbeit aber nicht weiter eingegangen.

Der MOS dieser beiden Verfahren ist jedoch nicht direkt vergleichbar. Durch technischen Fortschritt kann ein PESQ-MOS, welcher schlechter ist als ein im selben Netz gemessener klassischer MOS, tatsächlich besser klingen. Dasselbe gilt für das E-Model

Im Folgenden wird analysiert, welche Schritte in der VoIP-Telefonie zu welcher Art von Qualitätsproblemen führen kann.

2.1 M2E-Delays

In einem VoIP-Netzwerk entstehende Delays können in sechs Kategorien eingeteilt werden.[8]

Coder Delay/Algorithmic Delay Delay der durch das Sampling des Audiosignales sowie der Komprimierung der gesampleten Audiodaten entsteht.

Packetization Delay Delay, der dadurch entsteht, dass das gesamplete und allenfalls komprimierte Audiosignal in ein Datenpaket verpackt wird.

Serialization Delay Zeit, die benötigt wird, um die einzelnen Bits des Datenpakets auf das Netzwerkinterface zu serialisieren.

Queuing/Buffering Delay Delay, der durch überlastete Netzwerkgeräte entsteht. Mit modernen Quality of Service (QoS)-Massnahmen vernachlässigbar, sofern VoIP-Daten höchste Priorität auf den Router/Switches haben.

Network Switching Delay Delay, der im Netzwerk durch Switching/Routing entsteht sowie durch die physikalische Übertragung der Daten von A nach B.

Jitter Delay Delay, der dadurch entsteht, um andere variable Delay-Quellen auf Empfängerseite wieder auszugleichen.

2.1.1 Coder Delay/Algorithmic Delay

Coder Delay ist die Zeit, die benötigt wird, um ein Audiosignal mittels PCM zu samplen sowie die Samples zu komprimieren.

Der *Coder Delay* ist davon abhängig, wie hoch die Abtastraste der PCM ist. Bei einer Abtastrate von 8000 Hertz ergibt sich, dass alle 0.125 ms ein PCM-Audiosample erzeugt wird, wie in Gleichung 1 berechnet. Dies bedeutet auch, dass der Coder erst mit einer Verzögerung von Δt mit der Audiocodierung beginnen kann.

$$\begin{aligned} \Delta t &= \frac{1s}{8000} \\ \Delta t &= 1.25 * 10^{-4} s \\ \Delta t &= 125 \mu s \end{aligned} \tag{1}$$

Der behandelte Sprachcodec (G.711) überträgt normalerweise 160 Samples in einem einzigen Frame, was einem Audiosignal mit einer Dauer von 20 ms entspricht.

Einige Codecs führen während der Kompression noch einen weiteren *Algorithmic Delay* ein. Um die PCM-Samples korrekt zu komprimieren, muss der Kompressionsalgorithmus die Sprachcharakteristiken des Audioblock N analysieren, was einen kurzen Moment dauern kann. Einige Algorithmen benötigen dafür zusätzliche Informationen über die Sprachcharakteristiken in Block $N+1$. Sie müssen also in den nächsten Audioblock hineinschauen können, um das Sprachsignal des aktuellen Audioblockes zu komprimieren und später auch um es zu rekonstruieren. Dieser sogenannte *Lookahead*, sowie die Zeit zur Kompression des Signals wird auch *Algorithmic Delay* genannt, da er aus den Eigenschaften des Kompressionsalgorithmus hervorgeht.

Die Zeit für die Dekompression der Audiosignale wird ebenfalls zum allgemeinen Coder Delay dazugerechnet, auch wenn dieser Schritt erst beim Empfänger dazukommt. Dies ist nicht weiter problematisch, da es sich bei all diesen Delays um konstante Delays handelt. Die Dekompressionszeit eines Blockes beträgt rund 10% der benötigten Kompressionszeit, multipliziert mit der Anzahl Samples pro Frame.[8]

2.1.2 Packetization Delay

Packetization Delay ist die Zeit die benötigt wird, um ein Datenpaket zu bilden, welches danach über das Netzwerk übertragen wird. Bei Conjugate Structure Algebraic Code-Excited Linear Prediction (CS-ACELP) werden typischerweise je zwei Blöcke zu je 10 ms zusammen übertragen. Abbildung 3 verdeutlicht dabei, dass neben dem Zeitaufwand für das samplen, zusätzlich noch Zeit für das komprimieren und für den Lookahad einberechnet werden müssen, wodurch am Ende ein Delay von 27.5 ms entsteht.

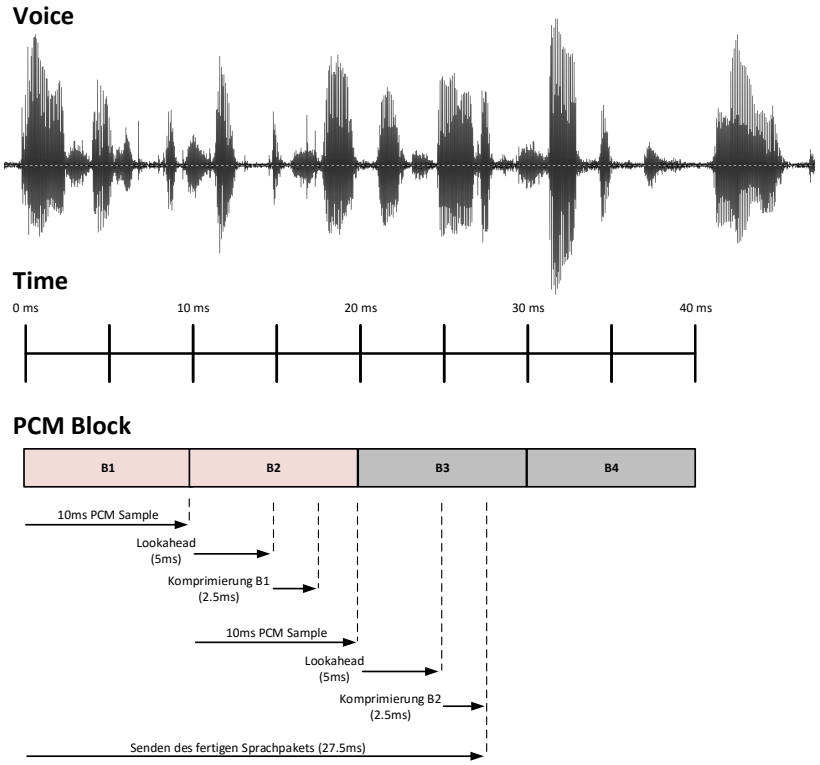


Abbildung 3: Packetization Delay bei CS-ACELP

Häufig wird dem *Packetization Delay*, wie auch in der Abbildung 3 ersichtlich, auch der *Coder Delay* hinzugerechnet. Grund dafür ist, dass diese Schritte häufig parallel ablaufen, und eine genaue Unterscheidung deshalb schwierig ist.

2.1.3 Serialization Delay

Serialization Delay beschreibt die Zeit die benötigt wird, um das Datenpaket mit den Sprachsamples komplett auf das Netzwerkinterface zu serialisieren. Dabei gilt

die Formel aus Gleichung 2.

Nimmt man an, dass ein 64 Byte (512 Bit) grosses Datenpaket in einem 100 Mbit/s Netzwerk serialisiert werden soll, so berechnet sich die effektive Zeit für die Serialisierung wie folgt:

$$\begin{aligned} \textit{Serialization Delay} &= \frac{\textit{Frame Size}}{\textit{Link Speed}} \\ \textit{Serialization Delay} &= \frac{512bit}{10^{8\text{bit}/s}} \\ \textit{Serialization Delay} &= 5.12 \cdot 10^{-6}\text{s} \\ \textit{Serialization Delay} &= 5.12\mu s \end{aligned} \tag{2}$$

In Tabelle 2 sind die *Serialization Delays* für verschiedene Verbindungsgeschwindigkeiten und Paketgrössen angegeben. Es ist ersichtlich, dass bei hohen Bandbreiten der durch die Serialisierung entstehende Delay vernachlässigbar wird. Aber besonders bei tiefen Bandbreiten von 100 kbit/s bis 400 kbit/s, wie sie heutzutage z.B. in Satellitenverbindungen immer noch anzutreffen sind, hat der *Serialization Delay* sehr wohl einen Einfluss. Beim vorherigen Beispielpaket mit 64 Byte würde es in einem 100 kbit/s Netzwerk rund 5.12 ms dauern, bis es auf das Netzwerk serialisiert wurde.

Frame Size [Bytes]	**Link Speed [kbit/s]**					**Link Speed [Mbit/s]**	
	100	*200*	*400*	*1000*	*2000*	*10*	*1000*
64	5.12	2.56	1.28	0.512	0.256	0.0512	0.000512
128	10.24	5.12	2.56	1.024	0.512	0.1024	0.001024
256	20.48	10.24	5.12	2.048	1.024	0.2048	0.002048
512	40.96	20.48	10.24	4.096	2.048	0.4096	0.004096
1 024	81.92	40.96	20.48	8.192	4.096	0.8192	0.008192
2 048	163.84	81.92	40.96	16.384	8.192	1.6384	0.016384

Tabelle 2: Serialization Delay *für verschiedene Paketgrössen und Link-Speed (alle Werte in ms)*

2.1.4 Queuing/Buffering Delay

Wenn Sprachdaten im Netzwerk nicht mittels QoS absolut höchste Priorität zugewiesen bekommen, ist mit *Queuing/Buffering Delay* zu rechnen. Dieser entsteht durch überlastete Netzwerkknoten, wenn ein Paket darauf warten muss, dass andere Pakete vor ihm durch den Router oder Switch geschleust werden. In korrekt konfigurierten Netzwerken ist dies heutzutage nur noch selten ein Problem, da mit

modernen QoS-Massnahmen das Routing/Switching von VoIP-Paketen bevorzugt ausgeführt wird.

2.1.5 Network Delay

Die weitaus grösste Quelle für Delay in einem VoIP-Netzwerk sind die öffentlichen Netzwerke. Einerseits fällt hier die physikalische Geschwindigkeit der Datenübertragung ins Gewicht (siehe Appendix A.1), aber auch das Peering zwischen verschiedenen Internet Service Provider (ISP) kann zu einem messbaren Delay führen.

Der *Network Delay* kann je nach Anzahl der Netzwerkknoten, Übertragungsgeschwindigkeiten, QoS-Einstellungen, der physikalischen Distanz zwischen Sender und Empfänger und weitern Faktoren stark variieren.

2.1.6 Jitter Delay

Die Sprachwiedergabe setzt eine konstante Datenrate voraus. Die Latenz in einem Netzwerk hat aber auch variable Faktoren, wodurch Datenpakete nicht immer gleichmässig ankommen. Diese *Packet Delay Variation* wird auch Jitter genannt. Würden die Pakete bei Erhalt sofort und ohne Ausgleich der Verzögerung abgespielt werden, würde das Audiosignal hörbar verzerrt werden. Teilweise würde die Sprachausgabe kurz unterbrochen, da auf den Erhalt des nächsten Audiopakets gewartet werden muss.

Damit die Sprachausgabe regelmässig und ohne Unterbrüche stattfindet, muss der Jitter herausgefiltert werden. Dazu wird beim Empfänger ein Jitter-Buffer unterhalten. Ankommende Sprachpakete werden für kurze Zeit zurückgehalten, bevor sie nach einer bestimmten Zeit (je nach Grösse des Jitter-Buffer) aus dem Jitter-Buffer entlassen und abgespielt werden. Das bringt die Sicherheit, dass das analoge Sprachsignal auf Empfängerseite gleichmässig reproduziert werden kann.

2.1.7 Unaccounted Delay

Bereits in den ersten durchgeführten Messungen in Abschnitt 4 hat sich herausgestellt, dass mittels den bisher beschriebenen Delay-Varianten nicht der vollständige M2E-Delay beschrieben werden konnte. Dieser Unterschied zwischen Theorie und Praxis hat sich als äusserst variabel herausgestellt und ist hauptsächlich abhängig vom eingesetzten Client.

Ein Hauptaspekt dieser Arbeit ist deshalb die Analyse, welche Komponenten für diesen zusätzlichen Delay verantwortlich sind und wie dieser zustande kommt.

2.1.8 Empfehlung für die Delay-Grenze

Die International Telecommunication Union (ITU) hält ihre Empfehlung für den Delay von Sprachapplikationen im Dokument G.114 fest.[11] Dieses enthält einen Leidfaden über die Auswirkung eines *End-to-End One-Way Delay* (auch Latenz genannt) auf die Sprachqualität eines Gespräches.

Unabhängig von der Art der Anwendung wird geraten, einen *One-Way Delay* von 400 ms nicht zu überschreiten. Ab dann wird der hohe Delay von den einzelnen Telefonteilnehmern als störend empfunden und man unterbricht sich gegenseitig während dem Sprechen. Idealerweise hält sich der Delay unter 150 ms.

Es ist aber durchaus bekannt, dass diese Grenze in einigen Fällen zwangsläufig überschritten wird. Als Beispiel für solch eine Ausnahme ist die Verwendung eines Satelliten, wo bereits durch physikalische Übertragung des Signals Verzögerungen von rund 360 ms entstehen, wie in Appendix A.1 berechnet wird. Falls schwer zu erreichende Regionen telefonisch abgedeckt werden müssen, sind solche Szenarien unvermeidlich.

In Hinblick auf die Benutzerakzeptanz wird das Modell in Abbildung 4 hinzugezogen. Dabei spiegelt die Abszissenachse den MOS, also das Benutzerempfinden, ab und die Ordinatenachse den M2E-Delay (in Millisekunden) wider. Es zeigt sich, dass das Empfinden erst bei einer Grenze von ca. 180 ms nahezu linear abnimmt.

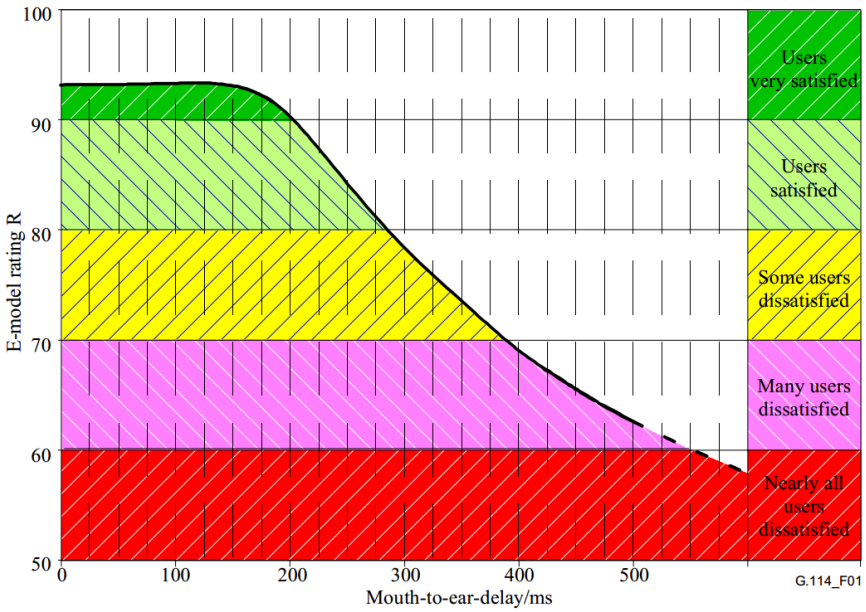


Abbildung 4: M2E-Einfluss auf MOS[11]

2.2 ITU-T G Series

In dieser Arbeit werden vor allem Audio-Codecs aus der ITU-T G Series analysiert. Die wichtigsten sollen hier kurz vorgestellt werden.

G.711 (PCM) G.711 ist der Name der ITU für das normale Pulse Code Modulation Sampling eines Audiosignales mit 8000 Hertz. Die PCM-Daten werden im Anschluss noch kompandiert. In den USA und Japan wird dazu das μ-Law eingesetzt, während im Rest der Welt das a-Law Verfahren eingesetzt wird.[31] Die Unterschiede der beiden Verfahren haben einen kleinen Einfluss auf die Sprachqualität, nicht aber auf die Performance der Algorithmen. Die Datenrate der Sprachdaten beträgt 64 kbit/s.

G.711.0 Die kompandierten PCM-Daten werden zusätzlich mit einem verlustlosen Kompressionsvorgang komprimiert. Dadurch wird die Grösse der PCM-Daten um rund 50% reduziert.

G.711.1 Ist eine Erweiterung von G.711 um die Fähigkeit der Breitbandkommunikation. Damit die normal verwendete Abtastrate von 8000 Hertz auf 12 000 oder 16 000 Hertz erhöht. Die ersten 8000 Hertz werden dabei identisch behandelt, wie im G.711 Standard, während die zusätzlichen 4000 bis 8000 Hertz dabei mit dem aus G.711.0 bekannten Verfahren um 50% komprimiert werden. Die zu übertragende Datenrate erhöht sich damit auf 80 oder 96 kbit/s.

G.726 Der G.726 Standard verwendet Adaptive Differential Pulse Code Modulation (ADPCM) um die benötigte Bandbreite von PCM weiter zu reduzieren. Dies wird erreicht, indem der Quantisierungsschritt während der Kompandierung (welcher bei G.711 durch das a-Law, bzw. das μ-Law vorgegeben ist) der PCM-Daten dynamisch gewählt wird.

G.729 Der G.729 Standard verwendet Conjugate Structure Algebraic Code-Excited Linear Prediction (CS-ACELP) und überträgt die reinen Audiodaten mit einer Datenrate von nur 8 kbit/s, wobei eine ähnliche Sprachqualität wie bei PCM erreicht wird. Das CS-ACELP Verfahren ist relativ kompliziert und soll hier nicht genauer beschrieben werden. Der Codec ist lizenzpflichtig.

Die Tabelle 3 zeigt berechenbare Delays, welche bei der Benutzung eines bestimmten Codecs entstehen. Der *Serialization Delay* wurde hier nicht berücksichtigt, da dieser abhängig von der verfügbaren Bandbreite ist. Bei den Werten wird jeweils angenommen, dass mit einem einzelnen Paket insgesamt 20 ms Audiodaten übertragen werden.

Codec	Frame Duration	Coder Delay	Packetization Delay	Algorithmic Delay	Compression Delay	Total Delay
G.711	20	0.125	20	0	0	20.125
G.726	20	0.125	20	0	2.5 - 10	22.625 - 30.125
G.729	20	0.125	20	5	2.5 - 10	27.625 - 35.125

Tabelle 3: Berechenbare Delays mit verschiedenen Codecs (alle Werte in ms)[8]

2.3 VoIP-Protokolle

Wenn ein Gespräch über einen Voice over IP-Kanal geführt werden soll, spielen mehrere Protokolle eine Rolle. Diese sollen hier genauer Analysiert werden.

2.3.1 SIP & RTP

Jede VoIP-Verbindung ist im Grundsatz ein Zusammenspiel zwischen zwei Protokollen, Session Initiation Protocol (SIP) und dem Realtime Transport Protocol (RTP). Dabei wird SIP für den Aufbau einer Verbindung sowie deren Terminierung verwendet, während RTP für die eigentliche Kommunikation verwendet wird. Es ist zu berücksichtigen, dass beide Protokolle nicht ausschliesslich für VoIP verwendet werden.

So ist das Session Initiation Protocol ein allgemeines Protokoll zum Sessionaufbau. Auch wenn das Protokoll am häufigsten für den Verbindungsaufbau im Zusammenhang mit Internettelefonie verwendet wird, kann man es grundsätzlich immer nutzen, wenn es um den Aufbau, die Modifikation und die Terminierung von Verbindungen für einen oder mehrere Medienstreams geht.

Das Realtime Transport Protocol hingegen kann immer dann verwendet werden, wenn es um die Übertragung von Medienstreams zwischen mehreren Clients geht. Auch hier ist der häufigste Anwendungsfall die Internettelefonie mittels Audio und Video. Es gibt aber auch andere Anwendungszwecke, z.B. im Bereich des Internetfernsehens.

RTP wird zudem durch das RTP Control Protocol (RTCP) unterstützt. Während RTP den Medienstream überträgt, wird RTP Control Protocol (RTCP) dazu verwendet, Verbindungs- und QoS-Statistiken zu übertragen. Damit hilft es den eingesetzten Client bei der Synchronisation mehrerer RTP-Streams und der Ermittlung bestimmter Leistungsparameter. RTCP ist im Abschnitt 2.3.2 genauer beschrieben.

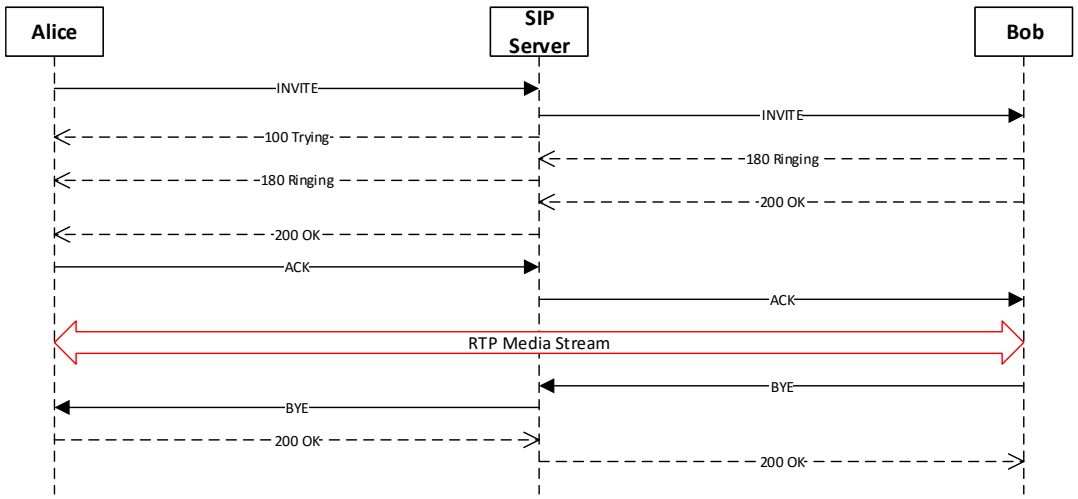


Abbildung 5: SIP/RTP Verbindungsablauf[27]

Verbindungsaufbau In Abbildung 5 ist der vollständige Ablauf einer SIP-Session ersichtlich. Die Anruferin Alice meldet sich beim SIP-Server, und teilt ihm mit, dass er einen bestimmten anderen Client anrufen will, in diesem Fall Bob. Dazu übermittelt Alice einen SIP `INVITE` Request an den Server. Der Server leitet dann einerseits den `INVITE` weiter an Bob, während er Alice mitteilt, dass er versucht eine Verbindung mit Bob aufzubauen. Dies wird Alice mittels dem SIP Statuscode `100` (Trying) mitgeteilt.

Sofern Bob erreichbar ist, schickt dieser dem Server als Antwort auf den `INVITE` den Statuscode `180` (Ringing) an den Server. Damit bestätigt Bob, dass bei ihm ein SIP Client verfügbar ist, welcher das Gespräch entgegennehmen könnte, und dass es bei diesem nun klingelt. Der entsprechende Benutzer hat aber noch nicht angenommen.

In diesem Zustand verharrt das System, bis entweder ein Timeout erreicht wird, oder Bob den Anruf bestätigt. In diesem Fall schickt Bob den Statuscode `200` (OK) an den Server, welcher diesen Code wiederum an Alice weiterleitet. Diese bestätigt den Verbindungsaufbau mit einem SIP `ACK` Request. Dieser geht wieder über den Server zu Bob weitergeleitet wird.

Nun sind alle Informationen zwischen Alice und Bob übermittelt, die benötigt werden, um einen direkten Kommunikationskanal zwischen den beiden aufzubauen. Über diesen Kommunikationskanal wird dann das eigentliche Gespräch geführt.

Um das Gespräch zu beenden schickt einer der beiden Clients einen SIP `BYE` Request an den Server, welcher den Request wie immer an den anderen Gesprächsteilnehmer weiterleitet. Dieser quittiert den Gesprächsabschluss mit dem Statuscode

`200` (OK) beim Server. Nachdem auch dieser Statuscode weitergeleitet wurde, ist die Verbindung terminiert.

Es ist wichtig, dass ausschliesslich der Verbindungsaufbau mittels SIP einen Zwischenhalt beim Server verlangt, während die eigentliche Kommunikation zwischen Alice und Bob in einer direkten Verbindung stattfinden kann. Die Audiodaten machen also keinen Umweg über den SIP-Server. Es könnte konfiguriert werden, dass auch die Kommunikation mittels RTP über den Server umgeleitet wird. Im Rahmen dieser Arbeit wird dies aber nicht weiter angeschaut. Für den M2E-Delay sind deshalb nur Netzwerkkomponenten relevant, die auch tatsächlich zwischen den beiden Teilnehmern liegen.

RTP Protokoll Analyse Um später die entstehenden M2E-Delays genau zu verstehen, ist eine Analyse des Realtime Transport Protocol unabdingbar.

Da es sich bei RTP, wie es der Name schon sagt, um ein Echtzeit-Protokoll handelt, werden RTP-Frames innerhalb eines User Datagram Protocol (UDP) Datagrams gekapselt und verschickt. RTP ist zwar auch zur Verwendung mit Transmission Control Protocol (TCP) standardisiert worden, da dieses aber Reliability über Timeliness setzt, wird TCP nur bei einer Minderheit der RTP-Verbindungen eingesetzt. RTP-over-TCP wird deshalb im Rahmen dieser Arbeit nicht weiter analysiert.

Die vollständige Struktur eines gekapselten RTP Pakets inklusive Header und Payload sieht damit wie folgt aus:

Ethernet Header	IP Header	UDP Header	RTP Header	RTP Payload (PCM Modulated Data)

Tabelle 4: RTP Paketstruktur

Für die Delayanalyse ist wiederum hauptsächlich die Paketgrösse relevant, bzw. der Overhead pro verschicktem Paket. Da die RTP-Payload ausschliesslich aus Nutzdaten besteht (und allenfalls einem Padding, um algorithmische Vorgaben zu erfüllen, z.B. für Verschlüsselung), sind ausschliesslich die diversen Frameheader der Media Access Control (MAC), Internet Protocol (IP), UDP und RTP Pakete relevant, wie in Tabelle 5 ersichtlich.

Es lässt sich also festhalten, dass für jedes RTP Paket ein Overhead von 54 Bytes entsteht. Der genaue prozentuale Overhead pro Frame lässt sich jedoch nur unter Kenntnis des tatsächlich eingesetzten Codecs sowie einigen Parametern berechnen. So variiert zum Beispiel die Dauer und Anzahl der gesampleten Audioblöcke pro übermitteltem Paket, sowie die benötigte Bitrate, um ein Audioframe darzustellen. Nur unter Berücksichtigung dieser Parameter kann der tatsächliche relative Overhead berechnet werden.

Protokoll	Headergrösse
Ethernet	14 Byte
IPv4	20 Byte
UDP	8 Byte
RTP	12 Byte
Total	54 Byte

Tabelle 5: RTP Overhead

Als Beispiel soll der G.711 Codec herhalten. Dieser übermittelt in der Standardkonfiguration 20 ms pro Frame. Da der Codec mit einer Abtastrate von 8000 Hertz arbeitet, finden sich in 20 ms insgesamt 160 Samples wieder. Jedes dieser Samples wird mit 8 Bit oder 1 Byte codiert. Die Nutzlast beträgt somit 160 Byte. Zusammen mit allen Headern werden pro Frame insgesamt 214 Byte übertragen. Das zu übertragende Paket wird somit um 33.75% aufgeblasen, womit rund 25.2% der übertragenen Daten Overhead darstellt.

2.3.2 RTCP

Bei der Verwendung einer Echtzeitanwendung wird grundsätzlich auf zwei verschiedene Protokolle zurückgegriffen, die eng miteinander verzahnt sind. Zum einen RTP, welches für den Transport der Daten verantwortlich ist und zum anderen RTP Control Protocol (RTCP), welches der Kontrolle dient.[14] Durch den periodischen Austausch von Berichten zwischen allen Sitzungsmitgliedern, können unter anderem folgende Aufgaben erfüllt werden:

- Überwachung der Übertragungsqualität
- Identifizierung der Teilnehmer
- Synchronisationsinformationen bei zusammenhängenden Daten-Streams

Die RTCP-Pakete beinhalten ebenfalls direkte Informationen über Paketverluste, Delays und Jitter. Dadurch werden auch Rückschlüsse auf die Round Trip Time ermöglicht.

RTCP Paketformat In den Spezifikationen von RTP sind fünf Paketformate definiert, die wie folgt lauten:

- Sender Report (SR)
- Receiver Report (RR)
- Source Description (SDES)

- Goodbye (BYE)
- Application-defined packet (APP)

Jedes dieser RTCP Paket beginnt mit einem fixen Header-Teil, welcher sich zwischen den unterschiedlichen Formaten nicht unterscheidet, und in Abbildung 6 gezeigt wird. Die ersten 2 Bit werden für die Version (V) verwendet, welche die RTP Version identifiziert. Mit (P) ist definiert, ob zusätzliche Padding Informationen am Ende des Pakets angehängt werden. Der Reception Report Count (RC) zeigt die Anzahl an *Reception Report* Blocks an, die das Paket beinhaltet. Packet Type (PT) definiert das RTCP-Format und im Feld Länge (Length), wird die Grösse des RTCP Pakets angegeben.[14]

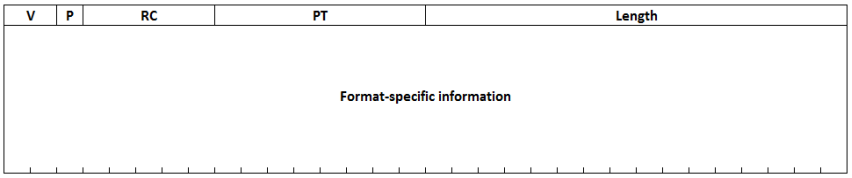


Abbildung 6: RTCP Basis Header

Die Anzahl an Paketformaten welches ein einzelnes RTCP-Paket beinhaltet, muss mindestens 2 betragen. Informationen wie RR oder SR werden niemals alleine versendet.

Sender Report Im Folgenden soll das Format des Sender Report genauer analysiert werden. Das RTCP-Paket SR wird periodisch vom Sender übermittelt, um dem Empfänger über die Qualität der Übertragung zu informieren. Typischerweise ist der Sender Report in drei Teile gegliedert.[14]

- RTCP Header
- Sender-Information
- Report Blocks

Der erste Teil, der 8 Byte umfasst, wurde bereits kurz erläutert. Dabei handelt es sich um den fixen Header-Teil, der bei allen RTCP-Paketen vorkommt. Der zweite Teil ist 20 Byte lang und existiert in jedem SR. Dieser Teil liefert Informationen über die Sendeaktivität. Die einzelnen Felder haben folgende Bedeutungen:

NTP Timestamp: 64 Bit Zeigt die Uhrzeit auf, zu der dieser Sender Report verschickt wurde. Dabei wurde das Format so gewählt, dass die Sekunden seit dem 1. Januar 1900 gezählt wurden. 32 Bit für die ganzzahligen Sekunden und 32 Bit für Gleitkommazahlen.

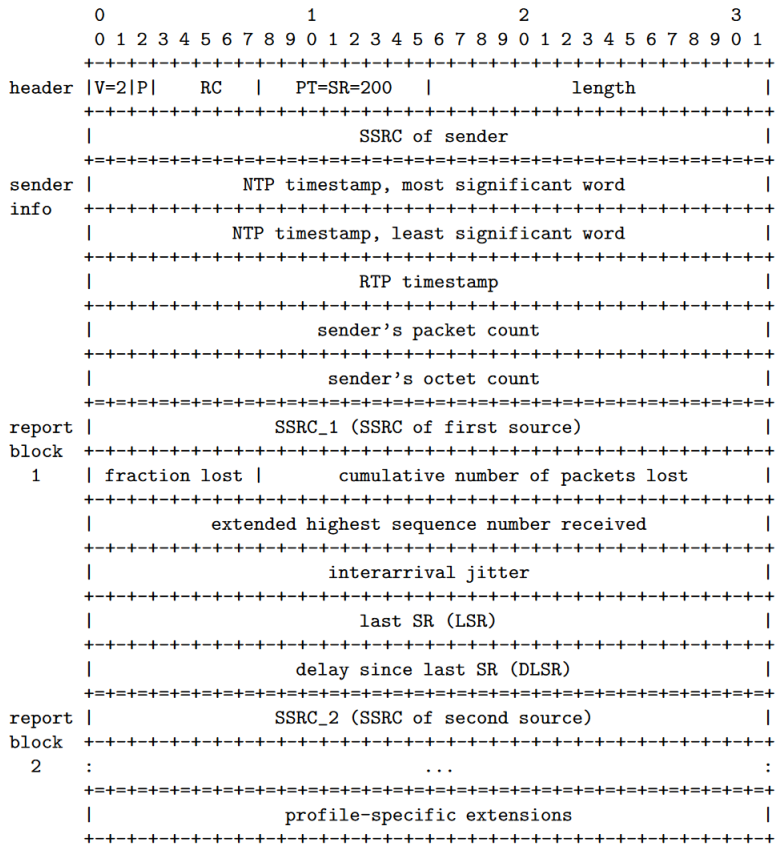


Abbildung 7: Sender Report[14]

RTP Timestamp: 32 Bit Referenziert ebenfalls die Sendezeit, jedoch in derselben Zeiteinheit, wie sie die RTP-Pakete benutzen.

Sender's Packet Count: 32 Bit Die Höhe entspricht der Anzahl RTP-Pakete, welche bis jetzt in dieser Session gesendeten wurden. Der Zähler sollte zurückgesetzt werden, falls sich die Synchronization Source (SSRC) ändert.

Sender's Octet Count: 32 Bit Das Feld gibt die Anzahl Bytes der gesendeten RTP-Pakte an (ohne Header und Padding). Der Zähler sollte zurückgesetzt werden, falls sich die SSRC ändert.

Der dritte Teil beinhaltet zwischen 0 und 31 Report Blocks. Jeder Bericht beschreibt die Empfangsqualität einer einzelnen Quelle. Die Felder sind wie folgt definiert:

SSRC: 32 Bit Dieses Feld dient zur Identifikation des Senders.

Fraction Lost: 8 Bit Gibt die Verlustrate im Verhältnis der verlorenen zu den insgesamt erwarteten RTP-Paketen an. Der Wert bezieht sich auf den letzten

Receiver Report oder Sender Report.

Cumulative number of packets lost: 24 Bit Gibt die Zahl aller verlorenen RTP-Pakete seit Beginn der Session an.

Extended highest sequence number received: 32 Bit Die niederwertigen 16 Bit, repräsentieren die höchste empfangene Sequenznummer in dieser Session. Mit den restlichen 16 Bit kann angegeben werden, wie oft sich dieser Zyklus bereits wiederholt hat.

Interarrival Jitter: 32 Bit Der *Interarrival Jitter J* ist eine Schätzung über den mittleren Jitter der zeitlichen Differenz T zwischen zwei hintereinander empfangenen und den zugehörigen gesendeten RTP-Paketen. Der erste Teil der Formel ist in Gleichung 3 beschrieben.[14]

$$\begin{aligned} D_{i,j} &= \left(R_j - R_i\right) - \left(S_j - S_i\right) \\ D_{i,j} &= \left(R_j - S_j\right) - \left(R_i - S_i\right) \end{aligned} \tag{3}$$

Dabei stellt R die Ankunftszeit (*Time of Arrival*) und S den RTP Timestamp für ein Paket dar. Die Veränderung des Jitters wird nach jedem RTP-Paket berechnet. Um kurzfristige Schwankungen im endgültigen *Interarrival Jitter* zu vermeiden, wird die Berechnung gemäss Gleichung 4 geglättet.[14]

$$J_i = \frac{J_{i-1} + \left(\left|D_{(i-1),i}\right| - J_{i-1}\right)}{16} \tag{4}$$

Ein Beispiel für die Berechnung des *Interarrival Jitters* anhand einer realen Wireshark Aufzeichnung findet sich in Appendix A.2.

Last SR timestamp (LSR): 32 Bit Dieses Feld beinhaltet die mittleren 32 Bit des zuletzt empfangenen NTP-Timestamp. Falls noch kein SR Empfangen wurde, wird dieser Wert auf 0 gesetzt. Diese Angabe kann zur Abschätzung der Round Trip Time dienen.

Delay since last SR(DLSR): 32 Bit Die Zeit seit dem zuletzt empfangenen SR in 1/65 536 Einheiten. Die Angabe kann zur Abschätzung des *Round-Trip Delays* dienen.

Wireshark – RTP Stream Analyse Mithilfe der Erkenntnisse aus Abschnitt 2.3.2 lassen sich alle Werte aus der Wireshark-Statistik wie in Abbildung 8 herleiten.

Als Grundlage für die folgenden Beispiele dienten die Wireshark Auswertungen, die auch in Abbildung 8 ersichtlich sind. Dazu wird zusätzlich der *RTP Timestamp* sowie die *Frame Arrival Time* benötigt, welche in Tabelle 6 hinterlegt sind.

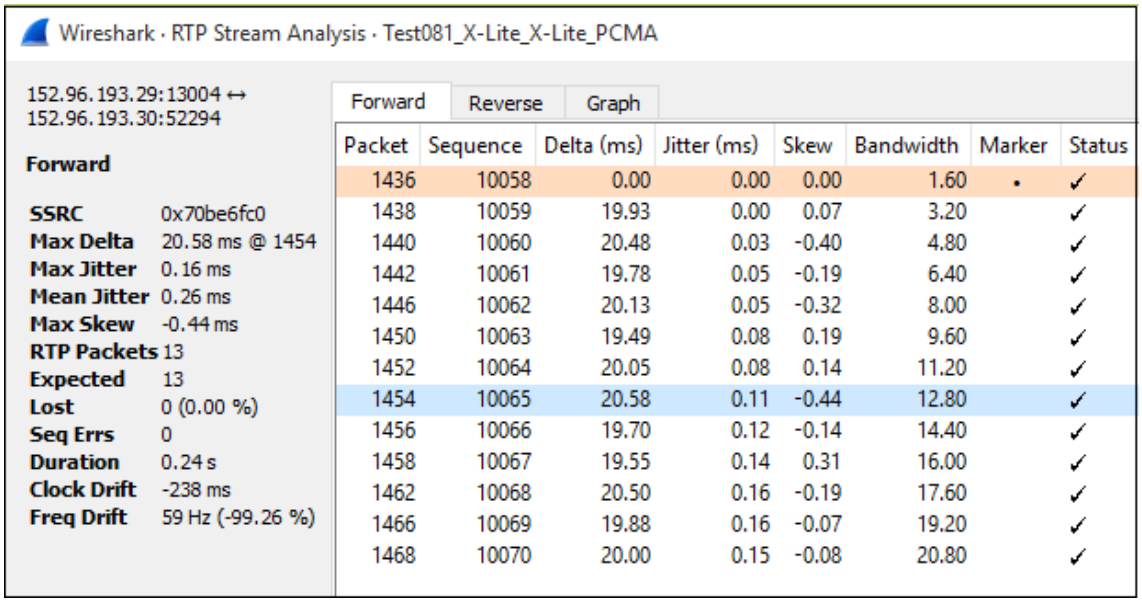


Abbildung 8: Wireshark-RTP Stream Analyse

Frame i	*Frame Arrival Time* R_i	*RTP Timestamp* S_i
1436	Mar 20, 2016 14:37:10.645880000	837 055 260
	30 645.880 ms	31 907.5 ms
1438	Mar 20, 2016 14:37:10.665806000	837 055 420
	30 665.806 ms	31 927.5 ms
1440	Mar 20, 2016 14:37:10.686285000	837 055 580
	30 686.285 ms	31 947.5 ms

Tabelle 6: Wireshark Frame Timesamps

Die Spalte *Delta(ms)* gibt dabei die Differenz zwischen dem aktuellen und dem zuvor empfangenen Paket an. Dabei kann für die Berechnung auf die Ergebnisse aus Gleichung 3 zurückgegriffen werden.

$$\text{Packet 1438} : Delta(ms) = 20ms + (-0.07ms) = 19.93ms$$
$$\text{Packet 1440} : Delta(ms) = 20ms + (0.48ms) = 20.48ms$$

Bei der nächsten Spalte, *Jitter(ms)*, handelt es sich um die einzelnen J_i Werte.

$$J_{1436} = 0$$
$$J_{1438} = 0$$
$$J_{1440} = 0.3$$

Der Parameter *Skew* gibt an, wie früh bzw. spät das Paket in Bezug auf die Erwartung ankommt. Bei einer Paketrate von 50 Paketen pro Sekunde sollten 20 ms zwischen den aufeinanderfolgenden Paketen vergehen. Falls ein Paket 19 ms nach dem vorhergehenden eintrifft, wird ein Skew-Wert von -1 ms angegeben. Zur Berechnung kann die Formel aus Gleichung 3 genommen werden. Dabei muss beim Ergebnis das Vorzeichen geändert werden.

$$-(D_{1436,1438}) = 0.07$$
$$-(D_{1438,1440}) = -0.40$$
$$-(D_{1440,1442}) = -0.19$$

Die Spalte *Bandwidth* in der RTP Stream Statistik zeigt die Bandbreite auf IP-Ebene auf. Dabei wird die Grösse des Pakets ohne den Ethernet-Header genommen.

$$Framesize \cdot 8^{bit/B} \cdot n\frac{Frame}{s} = Bandwidth,$$

wobei n der Anzahl Frames in den letzten 1000 ms entspricht (zu Beginn ist n somit relativ klein und steigt in der ersten Sekunde an, bis es den normalen Wert von 50 Frames erreicht), und

$$Framesize = IP\text{-}Header + UDP\text{-}Header + RTP\text{-}Header + RTP\text{-}Payload$$

ist. Damit entspricht die Bandbreite nach dem dritten Paket

$$(20 + 8 + 12 + 160)^{B/Frame} \cdot 8^{bit/B} \cdot 3^{Frame/s} = 4800^{bit/s} = 4.8^{kbit/s}.$$

Für den *Mean Jitter* wird nicht wie erwartet das arithmetische Mittel der Jitter-Werte verwendet, sondern der Durchschnitt der einzelnen absoluten Differenzen ($|D_i|$).

3 VoIP-Clients

Um eine Analyse von M2E-Delays in VoIP-Netzen durchzuführen, werden verschiedene Clients benötigt, um Messungen durchzuführen. Die Anforderungen an diese Clients sind in Abschnitt 3.1 zu finden. In diesem Abschnitt wird auch erwähnt, dass bei der Evaluierung der verschiedenen SIP-Clients kein Client gefunden wurde, welcher die Möglichkeit bietet, die Grösse des Jitter-Buffer manuell zu setzen.

In Abschnitt 4.3.2 wird zudem erwähnt, dass die Analyse der Resultate eine gewisse Schwierigkeit bietet, da sehr wenig darüber bekannt ist, was genau softwareseitig innerhalb eines VoIP-Clients passiert. Die bis zu diesem Zeitpunkt getroffenen Analysen betrafen jedoch allesamt Aspekte, die netzwerkseitige Ursachen hatten. Somit konnten bis dahin auch ohne genaue Kenntnisse der Software Rückschlüsse über das allgemeine Verhalten getroffen werden.

Um ein genaueres Verständnis der internen Vorgänge in einem SIP-Client zu erhalten, wurde die Entscheidung getroffen, einen eigenen SIP-Client zu entwickeln. Dieser soll die Möglichkeit bieten, einige Vorgänge zu analysieren, die sich innerhalb der Software abspielen. Diese Analysen sind im Abschnitt 3.4 beschrieben.

Im weiteren Verlauf fiel zudem die Entscheidung, dass es sinnvoll ist, einen SIP-Client zu entwickeln, mit welchem während einer VoIP-Verbindung Echtzeitstatistiken dargestellt werden können. Diese sollen einem versierten Anwender erlauben, z.B. einem Netzwerkadministrator, mehr Informationen über die aktuelle Verbindung zu erhalten. Dies soll dabei helfen, Schwachstellen und Probleme in VoIP-Verbindungen zu finden und beheben.

3.1 SIP-Clients

Für die durchgeführten Messungen wurden unterschiedliche Softphones auf SIP-Basis evaluiert. Als Hauptkriterien mussten zumindest folgende Punkte erfüllt werden:

- Basierend auf Windows
- G.711 Unterstützung
- Kostenfrei

Leider wurde vergeblich nach einer Software gesucht, bei der sich der Jitter-Buffer manuell einstellen lässt. Alle betrachteten Softphone-Anbieter setzen auf dynamische Buffer und geben nur sehr ungenaue Angaben, wie sich der Client im Einsatz verhält.

Eine Übersicht der ausgewählten Clients ist in Abbildung 9 ersichtlich. Darin sind ebenfalls drei Produkte enthalten, die über eine Verschlüsselungsfunktion verfügen.

Programm	Plattform	Audio-Codecs	Verschlüsselung	Features
X-Lite	MAC OS, Windows	G.711, GSM, OPUS, Speex	-	Acoustic Echo Cancellation (AEC), Background Noise Reduction (BNR), Automatic Gain Control (AGC)
Zoiper	Android, iOS, Windows Phone, Windows, Linux MAC OS, Browser	G.711, GSM, Speex, iLBC	SRTP, TLS	Automatic Gain Control (AGC), Acoustic Echo Cancellation (AEC), Noise Suppression
Jitsi	MAC OS, Windows, Linux	G.711, G.722, G.723, GSM, Opus, Speex, iLBC, SILK	SRTP, TLS, ZRTP	Acoustic Echo Cancellation (AEC), Background Noise Reduction (BNR), Automatic Gain Control (AGC)
Phoner-Lite	Windows	G.711, G.722, G.726, GSM, Opus, Speex, iLBC	SRTP, TLS, ZRTP	Acoustic Echo Cancellation (AEC), Stille-Erkennung

Abbildung 9: SIP-Client Charakteristik

3.2 Anonymisierung

Bis anhin wurde der Fokus ausschliesslich auf das Verständnis der Internettelefonie und der Übertragungsqualität gesetzt. Vernachlässigt wurde dabei die Sicherheit und Privatsphäre von einzelnen Telefonteilnehmern. Dieser Abschnitt soll sich nun mit der Verschlüsselung und Anonymisierung der Telefonie über das Tor-Netzwerk befassen.

3.2.1 Tor-Netzwerk

Tor ist ein verteiltes *Overlay-Netzwerk*, welches dafür entwickelt wurde, anonymisierte Verbindungen aufzubauen. Dabei werden TCP-Verbindungen eingesetzt, welche beispielsweise für Web-Browsing, SSH und Messanger Dienste verwendet werden können.[15] Der Client sucht sich dabei seinen Weg zum Ziel durch mehrere Netzwerkknoten, Onion Router (OR) genannt. Jeder dieser Knoten kennt dabei ausschliesslich seinen Vorgänger sowie Nachfolger und stellt eine verschlüsselte Verbindung her. Das soll den Nutzer davon schützen, dass der Datenverkehr analysiert werden kann. Zur Anonymisierungstechnik wird die Idee von Onion-Routing umgesetzt. Wie der Begriff bereits zu verstehen gibt, kann das Verfahren als Zwiebelprinzip angesehen werden. Die Verbindung zweier Knoten stellt dabei eine Schale dar. Im Verlauf einer Übertragung fallen dabei mehrere Schalen an.

3.2.2 Funktionsweise

Um das Tor-Netzwerk verwenden zu können, wird ein eigens dafür entwickelter Client benötigt, den sogenannten Onion Proxy (OP). Zu Beginn lädt sich das Pro-

gramm eine Liste aller nutzbaren Tor-Server von einem Directory-Server herunter. Mithilfe einer digitalen Signatur kann sicherstellt werden, dass es sich bei Serverliste um authentische Einträge handelt. Als Public-Key wird ein in die Software integrierter Schlüssel verwendet.[22] Der OP des Benutzers baut Schrittweise eine Verbindung zu den einzelnen OR auf. Dabei wird jedes Mal ein symmetrischer Schlüssel ausgehandelt. Im Detail sieht dies wie folgt aus:

Zu Beginn einer neuen Verbindung sendet der OP (beispielsweise Alice) *create cell* an den ersten Knoten (beispielsweise Bob). Der Inhalt der Anfrage beinhaltet den ersten Teil des Diffie-Hellman (DH)-Handshakes (g_x), wobei dieser durch den Key von Bob verschlüsselt ist. Bob antwortet darauf mit dem zweiten Teil des Handshakes (g_y) und mit dem ausgehandelten Hash-Wert $K = g_{xy}$.[23]

Sobald die Verbindung aufgebaut wurde, können Alice und Bob weitere Knoten in die Verbindung einbinden. Dazu sendet Alice einen *relay extend cell* an Bob. Daraufhin gibt es erneut einen Handshake zwischen Bob und dem neuen Knoten (beispielsweise Carol).

3.3 TORFone

TORFone ist eine auf Windows-basierte Applikation und wird zur sicheren und anonymen Internettelefonie verwendet. Zum Betrieb wird keinerlei Installation vorausgesetzt und kann beispielsweise auf einem mit TrueCrypt[17] geschützten Datenträger liegen.

Bei der Entwicklung wurde die ursprüngliche Idee des PGPFone[19] von Philip R. Zimmermann aufgegriffen und mit der Anonymisierungstechnik des Tor-Netzwerks erweitert. Für den Aufbau einer Telefonverbindung verzichtet man vollständig auf einen zentralisierten Server. Die Gespräche finden gleich zu Beginn direkt zwischen den Endgeräten statt. Dazu verwendet TORFone seine eigenen Protokolle, die nicht auf den klassischen SIP und RTP Protokollen basieren. Als Transportprotokoll kommt TCP zum Zuge, wobei UDP ebenfalls möglich ist.

Die gewonnene Sicherheit und Privatsphäre hat aber auch seinen Preis. Mit der Verwendung von Onion Router einstehen erhöhte Delays, die je nach Knotennetz bis zu 5 Sekunden zu Buche schlagen. Falls die Anonymität nicht benötigt wird, gibt es ebenfalls eine Möglichkeit, eine reine Point-to-Point Verbindung mithilfe einer IP-Adresse oder DNS-Namen aufzubauen.[20]

3.3.1 Betrieb

Um anonymisierte Anrufe über das Tor-Netzwerk abzusetzen, wird ein zusätzliches Programm benötigt. Erst mithilfe einer Software wie dem Tor-Chat, kann man eine Verbindung ins Tor-Netzwerk aufbauen und wird dabei gleichzeitig erreichbar für andere Teilnehmer.

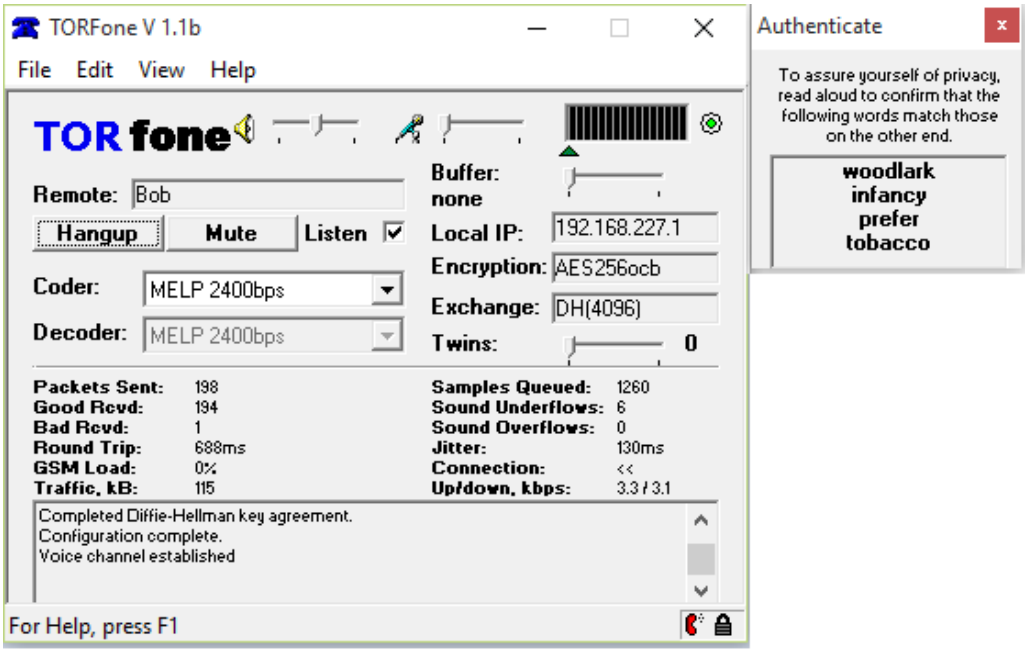


Abbildung 10: Aktiver TORFone Anruf zwischen Alice und Bob

Für letzteres ist der sogenannte *Hidden Service* verantwortlich. Beim Start des Tor-Chats wird eine einzigartige alphanumerische ID generiert, welche aus 16 Zeichen besteht. Diese ID wird zufällig generiert und dient als .onion Adresse. Durch diese Anonymisierung ist es nicht länger nötig, die reale IP-Adresse zu kennen, um mit einem Endgerät zu kommunizieren.

Falls keine oder nur eine einseitige Anonymität gewünscht wird, ist es ebenfalls möglich, den Modus "Tor2Peer" oder "Peer2Peer" zu wählen. In diesem Fall gibt es keine Anonymisierung, aber die Vertraulichkeit wird jederzeit sichergestellt. Ebenfalls ergeben sich bessere Delay-Werte durch das fehlende Tor-Netzwerk.[21]

Für den Austausch des symmetrischen Schlüssels wird Diffie-Hellman (DH) mit 4096 Bit verwendet. Zur Verschlüsselung kommt anschliessend AES-256-OCB (128 Bit MAC) zur Anwendung.

3.4 Realisierung eigener VoIP-Client (Lumisoft)

Wie in Abschnitt 3, sollte ein Client entwickelt werden, der es erlaubt, Informationen auf interne Vorgänge von SIP-Clients zu erhalten. Diese Entwicklung diente ausschliesslich zu Analysezwecken. Das Ziel war ausdrücklich nicht, eine Software zu entwickeln, welche als Produkt gesehen werden konnte. Auf ein klassisches Software-Engineering Vorgehen mit Projektplan, Domänenanalyse, Unittests etc. wurde deshalb bewusst verzichtet.

3.4.1 Anforderungen

Die Anforderungen an den Client sind folgendermassen:

- Rückschlüsse auf interne Vorgänge von SIP-Clients ziehen
- Einfluss von externen Faktoren auf dem ausführenden Computer analysieren
- Kurze Entwicklungszeit

Diese Anforderungen sollen im Folgenden etwas genauer analysiert werden.

Rückschlüsse auf interne Vorgänge Bis zu diesem Zeitpunkt waren die verwendeten Clients aus Sicht der erstellten Messungen immer eine Blackbox. Es konnten zwar Vorgänge analysiert werden, deren Ursachen im Netzwerk liegen, also alles, was sich zwischen der sendenden sowie der empfangenden Netzwerkkarte abspielt. Wie sich aber in Abschnitt 4.3.1 herausgestellt hatte, kamen die grössten Einflüsse nicht vom Netzwerk, sondern von den eingesetzten Clients.

Das Ziel war es also, besser zu verstehen, was sich auf dem Computer zwischen der Soundkarte und der Netzwerkkarte abspielte. Es ist nicht zu erwarten, aus diesen Analysen ein vollständiges Verständnis der Vorgänge in den Clients zu erarbeiten. Das Ziel war mehr, ein besseres Verständnis für mögliche Problemstellen zu erhalten.

Einflüsse von externen Faktoren Ein weiteres Ziel dieser Entwicklung war, besser verstehen zu können, wie sich die Auslastung des Computers, auf welchem die SIP-Clients ausgeführt wurden, auf den Client auswirken könnte. Beispielsweise, welchen Einfluss ein ausgelasteter Prozessor oder das gleichzeitige Abspielen von Sound auf die Performance der SIP-Clients haben könnte.

Kurze Entwicklungszeit Da es sich bei diesem Client nicht um ein Kernprodukt dieser Arbeit handelte, war ein zentraler Aspekt, dass die Entwicklungszeit nicht zu gross wurde.

3.4.2 Evaluation

Aufgrund der Anforderung, dass die Entwicklungszeit nicht zu gross werden durfte, wurde entschieden, dass die Entwicklung des Clients in C# stattfinden sollte. Dies hauptsächlich, weil die Stärke des eingesetzten Entwicklers im C#-Bereich lag. Ausserdem wurde der Entschluss getroffen, dass nicht ein vollständiger Client entwickelt, sondern ein bestehender Client um die gewünschten Features erweitert werden sollte.

Nach Evaluation verschiedener Software-Stacks fiel die Wahl schliesslich auf Lumisoft.NET.[28] Grund dafür war, dass andere analysierte Stacks nicht vollständig Open-Source waren. Besonders die Implementation der SIP- und RTP-Protokolle der Stacks waren meistens doch proprietär und nur entweder gegen hohe Kosten oder gar nicht verfügbar. Um also zu verstehen, wie sich ein Client verhält, waren diese Software-Stacks eher ungeeignet.

Ein weiterer Grund war, dass bei Lumisoft.NET eine Demoapplikation und einige gute Beispiele verfügbar waren, welche die Entwicklung weiter vereinfachten. Als Basis der Entwicklungen wurde die Demo Applikation Lumisoft.NET UA[29] verwendet.

Wie sich im späteren Verlauf herausgestellt hat, befinden sich im Lumisoft Client aber auch eklatante Schwächen. Der Client hat sich als eher instabil herausgestellt. Es gab diverse Abstürze. Es stellte sich auch heraus, dass der Lumisoft Client nur in Zusammenarbeit mit Jitsi überhaupt funktionierte. Mit Zoiper wurde der Anruf jeweils sofort wieder beendet, mit X-Lite hingegen wurde zwar eine Verbindung aufgebaut, allerdings wurde kein Ton übertragen. Ausserdem war es nicht möglich, den Lumisoft Client als Empfänger zu verwenden. Der Anruf musste immer von Lumisoft ausgehen. Auch die Messungen waren eher instabil, und es gab grosse Diskrepanzen zwischen verschiedenen Messreihen, wie in Tabelle 7 ersichtlich ist. Diese Punkte waren zwar problematisch, standen aber dem eigentlichen Ziel, Client-interne Analysen durchzuführen, nicht im Wege.

Sender	Empfänger	Messung M2E-Delay [ms]					
		#1	#2	#3	#4	⌀	σ
Jitsi	Lumisoft	155	130	127	289	**175**	**67**

Tabelle 7: Testmessungen zwischen Jitsi und Lumisoft (alle Werte in ms)

3.4.3 Analyse

Die Analyse soll das Verständnis des RTP-Paketflusses verbessern. Dafür wurde analysiert, welche Schritte während des Sendens und Empfangens durchgeführt werden müssen. Für alle Testmessungen wurde dabei der G.711 Codec, also PCM, mit einer Sampling-Rate von 8000 Hertz verwendet. Die Audioframes waren dabei jeweils 20 ms lang. Folgende relevanten Schritte wurden dafür selektiert:

- Senden

  Audio Frame erhalten Die Soundkarte leitet ein Audioframe an die Applikation weiter

  Vor RTP Paketerzeugung Kurz bevor das RTP-Paket erstellt wird

Vor PCM Codierung Direkt nach der Erzeugung des RTP Pakets und vor der PCM Codierung des Audioframes

Nach PCM Codierung Direkt nach der PCM Codierung des Audioframes

Vorbereitung für Senden Beginn der Vorbereitungen, um das Paket ans Netzwerk zu senden

Vor Senden Direkt bevor das Paket an die Netzwerkkarte geschickt wird

Nach Senden Direkt nachdem das Paket an die Netzwerkkarte geschickt wurde

Nachbereitung nach Senden Nachdem alle Nachbearbeitungen nach dem Senden durchgeführt wurden

- Empfangen

RTP Paket erhalten Ein RTP Paket wurde von der Netzwerkkarte an die Applikation weitergeleitet

Vor RTP Dekodierung Direkt bevor das Audioframe im RTP Paket decodiert wird

Nach RTP Dekodierung Nachdem das Audioframe im RTP Paket decodiert wurde

Audioframe an Soundkarte geschickt Nachdem die Soundkarte das Audioframe von der Applikation entgegengenommen hat (das Abspielen selbst erfolgt asynchron)

An den identifizierten Stellen im Code wurden entsprechend Messpunkte gesetzt. An diesen Messpunkten wurde jeweils die genaue Zeit gemessen. Basierend auf diesen Messungen wurde einerseits berechnet, wie lange der vorhergehende Schritt dauerte (im Folgenden *Schrittdauer* genannt), und andererseits, wie regelmässig die Schritte durchgeführt wurden, ob also bereits im Client ein relevanter Jitter entstand (Client-Jitter).

3.4.4 Testverfahren

Das Testverfahren wurde relativ simpel gehalten. Vom Lumisoft Client wurde einem Jitsi Client angerufen. Die eigentlichen M2E-Delays waren für die Analysen nicht relevant. Es wurde ein kurzer Anruf durchgeführt, wobei Geräusche erzeugt wurden, damit auch etwas übertragen werden musste. Ein Anruf dauerte mindestens 20 Sekunden, bevor die Statistiken erzeugt wurden, was 1000 Paketen entspricht. Wie sich gezeigt hatte, zeigten sich vor allem in den ersten sowie in den letzten Paketen einige Ausreisser. Deshalb wurden nur die Pakete von 100 - 899 zur Analyse verwendet.

Wie in Abschnitt 3.4.1 deklariert, sollten ausserdem Messungen mit externen Einflüssen durchgeführt werden. Diese externen Einflüsse waren einerseits das gleichzeitige Abspielen von Sound auf der Soundkarte, andererseits das Auslasten des Prozessors mit dem Tool StressMyPC.[30] Durch das parallele Abspielen von Musik sollte herausgefunden werden, ob dies einen Einfluss auf Client Jitter oder die *Schrittdauer* hatte. Die Auslastung des Prozessors wurde erst später zum Testverfahren hinzugefügt, um ein interessantes Verhalten bei der *Schrittdauer* zu analysieren. Dazu später mehr.

Client-Jitter Die Frage im Zusammenhang mit Client-Jitter ist, wieviel Einfluss der Client auf den gesamten messbaren Jitter hat. Deshalb ist die Jitter Analyse vor allem auf Senderseite interessant. Es ist davon auszugehen, dass der Jitter auf Empfangsseite hauptsächlich auf die Einflüsse des Netzwerks zurückzuführen wäre, weshalb eine Analyse dort wenig Sinn macht.

Es hat sich herausgestellt, dass auch der eher suboptimale Lumisoft Client keinen nennenswerten Einfluss auf den Jitter hatte, auch wenn tatsächlich ein geringer Jitter entstand, wie in Tabelle 8 ersichtlich. Die Werte entstanden, indem jeweils die Zeit seit demselben Schritt beim vorherigen Paket gemessen wurde. Davon wurde die Standardabweichung berechnet. Werden die Resultate nicht um Ausreisser bereinigt, liegt die Standardabweichung bei 0.3 bis 1.5 ms.

Es hat sich aber gezeigt, dass jeweils einige wenige Ausreisser die Standardabweichung massiv beeinflusst haben. In Tabelle 9 wurden deshalb alle Messpunkte entfernt, die um mehr als 5000 μs vom erwarteten Mittelwert von 20 000 μs abwichen. In keiner Messreihe wurden mehr als 4 Messpunkte entfernt. Es zeigte sich, dass mit eliminierten Ausreissern der Client-interne Jitter durchwegs auf unter 1 ms fiel, in den meisten Fällen gar unter 0.5 ms. Nur bei ausgelasteter CPU wurde ein etwas höherer Jitter von 0.8 ms gemessen, was aber noch immer keinen relevanten Einfluss auf den in einem Netzwerk entstehenden Gesamtjitter hat.

Schrittdauer Da es bei der *Schrittdauer*-Analyse darum ging, wieviel zusätzlicher Delay im Client entsteht, macht in diesem Fall die Analyse sowohl auf Sender- wie auch auf Empfängerseite Sinn. Bei der *Schrittdauer*-Analyse müssen aber die Messungen als Ganzes betrachtet werden. Diese sind wiederum im Anhang zu finden. Die Resultate sind in Tabelle 10 zusammengefasst. Es werden jeweils die Anzahl 0-Zeilen, der durchschnittliche Wert über alle Messungen, sowie die Standardabweichung aller Zeilen, die keine 0-Zeilen sind, angegeben. Auch für die *Schrittdauer* Messungen wurden jeweils 800 Messpunkte ausgewertet.

Sowohl auf Sender- wie auch auf Empfängerseite fällt auf, dass es sehr viele 0-Zeilen hat. 0-Zeilen sind jene Zeilen, bei denen die Gesamtdauer aller Schritte 0 μs beträgt (bei einer Auflösung von 100 ns). Nur vereinzelt dauerte die Verarbeitung

Messung	Audio Frame erhalten	Vor RTP Paketerzeugung	Vor PCM Codierung	Nach PCM Codierung	Vorbereitung für Senden	Vor Senden	Nach Senden	Nachbereitung nach Senden
Ohne Ton 1	1 166	1 152	1 152	1 153	1 154	1 154	1 148	1 146
Ohne Ton 2	349	347	347	348	348	347	360	354
Mit Ton 1	1 478	1 478	1 478	1 477	1 477	1 477	1 517	1 519
Mit Ton 2	1 225	1 225	1 225	1 224	1 224	1 224	1 333	1 333
CPU ausgelastet	904	903	902	902	902	901	1 186	1 186

Tabelle 8: Client-interner Jitter beim Senden (alle Werte in µs)

Messung	Ausreisser entfernt	Audio Frame erhalten	Vor RTP Paketerzeugung	Vor PCM Codierung	Nach PCM Codierung	Vorbereitung für Senden	Vor Senden	Nach Senden	Nachbereitung nach Senden
Ohne Ton 1	2	319	324	324	329	329	331	310	303
Ohne Ton 2	0	349	347	347	348	348	347	360	354
Mit Ton 1	4	124	123	123	118	118	111	282	290
Mit Ton 2	3	339	338	337	336	335	334	588	587
CPU ausgelastet	2	226	222	221	221	218	215	802	802

Tabelle 9: Client-interner Jitter beim Senden, Ausreisser entfernt (alle Werte in µs)

Messung	Empfangsseite			Senderseite		
	0-Zeilen	∅	σ	0-Zeilen	∅	σ
Ohne Ton 1	626	111.82	206.14	296	317.11	66.89
Ohne Ton 2	607	132.15	249.38	410	229.81	40.1
Mit Ton 1	537	164.5	194.42	511	188.17	147.21
Mit Ton 2	588	153.74	845.36	441	234.01	500.26
CPU Stresstest	610	336.83	4 296.28	591	156.83	1 032.91

Tabelle 10: Schrittdauer-*Messungen zusammengefasst (alle Werte in μs)*

einzelner Pakete länger. Dies zeigt, dass die Pakete im Allgemeinen in einem Rutsch verarbeitet werden können und keine Wartezeiten entstehen, wenn z.B. das Paket an die Netzwerkkarte geschickt oder das Audioframe an die Soundkarte geschickt wird.

Wenn die gesamte Verarbeitungszeit eines Paketes aber grösser als 0 μs ist, fällt auf, dass sich die Werte immer um 500 μs, bzw. einem Mehrfachen davon bewegen. Nach einiger Unschlüssigkeit entstand die Annahme, dass es sich hierbei um das Threading-Verhalten des Betriebssystems handelte. Vor allem die Regelmässigkeit der Werte deutete stark auf dieses Szenario hin. Ein weiterer Indikator war, dass die *Schrittdauer* meistens (aber nicht immer) dann zunahm, wenn asynchrone Operationen durchgeführt wurden. Beim Senden ist dies namentlich der Moment, indem das RTP-Paket an die Netzwerkkarte geschickt wird, beim Empfangen hingegen, wenn das Audioframe an die Soundkarte weitergeleitet wird. Die Methoden, welche das Paket an die Netzwerkkarte schicken, bzw. an das Frame an die Soundkarte, retournieren, sobald das Paket/Frame abgeliefert wurde. Abgespielt und serialisiert wird es hingegen asynchron.

Weiter erhärtet wird diese Vermutung, wenn die Standardabweichung aller Messungen grösser 0 auf Empfängerseite analysiert wird. Die Werte sind viel breiter gestreut, wenn gleichzeitig Ton abgespielt wird. Dies könnte daraufhin deuten, dass die Soundkarte gerade von einer anderen Applikation mit Daten gespiesen wird und der Client einen kurzen Moment warten muss, bis die Soundkarte wieder Daten entgegennehmen kann. Für das Betriebssystem ist dies ein optimaler Zeitpunkt, um den Prozessor kurzzeitig einer anderen Applikation zur Verfügung zu stellen.

Sollte es sich tatsächlich um Thread-Switches handeln, müsste sich die Anzahl und vor allem die Dauer der Messungen mit einer *Schrittdauer* grösser als 0 ms erhöhen, wenn die CPU ausgelastet ist. Dies, weil es häufiger zu Thread-Switches kommt und vor allem die CPU länger von anderen Prozessen blockiert würde. Um das Szenario zu bestätigen wurde derselbe Versuch noch einmal durchgeführt, allerdings wurde dieses Mal mit dem Tool StressMyPC[30] die CPU ausgelastet.

Die Messungen bestätigten das erwartete Szenario. Interessanterweise konnte

zwar nicht bestätigt werden, dass es häufiger zu 0-Zeilen kam, aber die Werte Zeilen grösser 0 waren massiv breiter gestreut. So lag die Standardabweichung auf Empfangsseite bei 4296 μs, statt normalerweise weniger als 300 μs, auf der Senderseite lag sie bei 1032 μs anstatt normalerweise unter 200 μs. Der Durchschnitt der zusätzlichen Verzögerungen lag aber auch bei ausgelasteter CPU bei weniger als 1 ms.

Der Einfluss auf die M2E-Verzögerungen durch den Client waren aber auch beim eher suboptimalen Lumisoft Client nur sehr gering und bewegten sich im Durchschnitt bei weniger als 1 ms (Sende- und Empfangsseite summiert). Die Vermutung, dass die Client-Implementation selbst einen grossen Einfluss auf den M2E-Delay hat, konnte damit nicht bestätigt werden, obwohl natürlich nur ein einzelner Client analysiert werden konnte.

Es ist aber zu berücksichtigten, dass der Lumisoft Client keinen Jitter-Buffer implementiert hat. Die Vermutung liegt deshalb nahe, dass hauptsächlich die Art und Weise, wie ein Jitter Buffer implementiert ist, einen grossen Einfluss auf den M2E-Delay hat. Mehr Informationen darüber, wie ein Jitter Buffer implementiert werden kann, finden sich in Abschnitt 3.5.6 und Abschnitt 3.5.7.

3.5 Erweiterung Jitsi-Client

In einem weiteren Schritt sollte ermöglicht werden, zur Laufzeit detaillierte Information über eine VoIP-Verbindung darzustellen. Ursprünglich war die Idee, den Lumisoft Client weiterzuentwickeln und einige Parameter aus den RTCP-Paketen darzustellen. Wie in Abschnitt 3.4 beschrieben, war der Lumisoft-Client aber sehr instabil. Den Client so weit zu bringen, dass dieser stabil genug für ein Echtweltszenario wäre, hätte einen vergleichsweise grossen zeitlichen Aufwand bedeutet.

Es wurde deshalb die Entscheidung getroffen, den Open-Source Client Jitsi zu verwenden, der auch schon für andere Tests eingesetzt wurde, und entsprechend weiterzuentwickeln. Wie aus diesen Tests bekannt, gab Jitsi schon in der Standardkonfiguration einige nützliche Informationen aus, wie sie in Abbildung 11 dargestellt sind.

3.5.1 Anforderungen

Die von Jitsi standardmässig angebotenen Statistiken bringen einem Netzwerkadministrator aber nur bedingt etwas, da diese eine statische Momentaufnahme darstellen und auch nur sehr eingeschränkte Statistiken angeboten werden. Die wichtigste Anforderung an den erweiterten Jitsi Client war, dass die Informationen in Echtzeit dargestellt werden können. Ausserdem sollten neben den standardmässig dargestellten Informationen weitere Statistiken visualisiert werden.

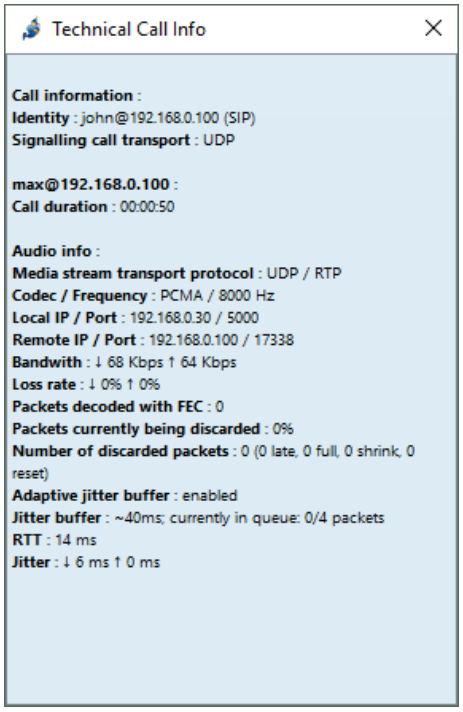


Abbildung 11: Jitsi Statistiken

3.5.2 Jitsi Struktur

Es soll hier nicht zu stark auf den Aufbau eingegangen werden. Wichtig ist in diesem Zusammenhang vor allem, dass Jitsi auf der Freedom for Media in Java (FMJ) Bibliothek basiert. Die gesammelten Statistiken werden eigentlich von dieser Bibliothek erzeugt. Jedem Media-Stream, wozu auch eine RTP Verbindung zählt, wird eine Instanz der Klasse `MediaStreamStats` zugewiesen. Diese sammelt diverse Statistiken zu Paketverlusten, Jitter und weiteren nützlichen Informationen.

3.5.3 Jitsi Entwicklung

Es wurde entschieden, die aktuelle Entwicklerversion von Jitsi einzusetzen. Dies hauptsächlich aus dem Grund, dass Jitsi in der Version 2.9 sehr stark modularisiert wurde, was es einfacher macht, einzelne Komponenten zu ersetzen. Es können einfach einzelne JAR-Files ersetzt werden, es muss also nicht die gesamte Installation modifiziert werden.

Wie sich leider erst spät herausstellte, hatte diese Entscheidung aber einen grossen Nachteil. So hat die aktuelle Entwicklungsversion von Jitsi zurzeit noch Probleme mit den RTCP-Paketen, bzw. Jitsi verschickt in den aktuellen Entwicklerversionen keine solchen. Das führt dazu, dass gewisse Statistiken, die auf RTCP

basieren, nicht korrekt berechnet werden. Dies betrifft Statistiken zu RTT und dem Up- und Download Jitter. Allerdings wird sich dieses Problem mit der Zeit von selbst lösen, da die Entwicklung von Jitsi relativ schnell vorangeht. An diesem Punkt zeigt sich dann auch der grosse Vorteil der modularen Installation. Sobald dieser Teil von Jitsi wieder korrekt funktioniert, kann einfach eine Neuinstallation durchgeführt werden, wie in Appendix B.2 beschrieben wird. Die neuen Features funktionieren dann automatisch.

3.5.4 Gesammelte Statistiken

In einem nächsten Schritt ging es darum, herauszufinden, welche Statistiken neben den bereits dargestellten zusätzlich gesammelt werden. Es gibt grundsätzlich zwei Arten von Statistiken. Einerseits sind dies Zähler, die verschiedene Arten von Information zählen. Darunter fallen gesendete und empfangene Bytes, gesendete und empfangene Pakete, verlorene Pakete sowie Pakete, die von Jitsis Empfangsbuffer verworfen wurden. Die zweite Kategorie umfasst Werte, die sich im Verlauf eines Telefonats nicht zu stark verändern. In diese Kategorie fallen der gemessene Jitter, die RTT, sowie die aktuelle Grösse von Jitsis Empfangsbuffer.

Es ist zu beachten, dass nicht alle Statistiken von Jitsi alleine erstellt werden können. So ist Jitsi für die Berechnung des Upload Jitters sowie der RTT auf die RTCP Empfängerreports angewiesen. Erstellt der gegenüberliegende Client aber nur mangelhafte RTCP Reports, kann Jitsi diese Informationen nicht selbst berechnen.

Im Folgenden werden die dargestellten Statistiken sowie deren Herkunft etwas genauer beschrieben. Screenshots der Implementation finden sich in Appendix B.3

Kilobytes sent/received Die über die gesamte VoIP-Verbindung gesendeten, bzw. erhaltenen Bytes aufsummiert.

Packets sent/received/total Die über die gesamte VoIP-Verbindung gesendeten, bzw. erhaltenen Pakete aufsummiert, sowie das Total.

Packets lost while sending/receiving/total Die über die gesamte VoIP-Verbindung während dem Senden, bzw. Empfangen verlorenen Pakete, sowie das Total.

Packets decoded with FEC Die während der VoIP-Verbindung mittels Forward Error Correction (FEC) decodierten Pakete.

Discarded Packets Pakete die aufgrund der Implementation des Jitter-Buffer von Jitsi verworfen wurden. Gründe dafür sind: Volle Buffer Queue, späte Ankunft, Queue wird resetted, Queue wird geschrumpft. Zudem wird das Total angezeigt.

Jitter Buffer Queue Size Grösse der verwendeten Queue für den Jitsi-Internen

Jitter Buffer. Die Funktionsweise der Jitsi-Jitter-Buffer Queue wird in Abschnitt 3.5.6 noch genauer beschrieben.

Packets in Queue Anzahl der Pakete in der Queue.

Approx Buffer Size ms/Packets Geschätzte Grösse des Jitter Buffers. Diese Information wird von Jitsi nur geschätzt angegeben, da der Klasse, welche die Statistiken sammelt, gewisse Informationen für eine genaue Angabe fehlen. Die Schätzungen werden abgeleitet aus dem verwendeten Codec und der aktuellen Grösse der Queue.

Round Trip Time Die aus den RTCP Paketen heraus berechnete RTT.

Upload/Download Jitter Der aktuelle Jitter im Up- und Download. Aus den RTCP Paketen heraus berechnet.

Min/Avg/Max Upload/Download Jitter Minimaler, durchschnittlicher und maximaler Jitter im Up- und Download. Aus den RTCP Paketen heraus berechnet.

3.5.5 Implementation

Um die Entwicklung zu vereinfachen, wurde bei der Implementation ein Weg gewählt, der die vorzunehmenden Anpassungen am bestehenden Code möglichst minimiert. Diese Anpassungen werden im Folgenden auf Basis des Klassendiagramms in Abbildung 12 illustriert.

Package `net.java.sip.communicator.impl.gui.main.call.stats.model` In diesem Package sind die Model-Klassen hinterlegt, in welchen die gesammelten Statistiken gespeichert werden. Die Klasse `CallConferenceStats` ist die Basisklasse der Struktur und enthält einige allgemeine Statistiken über verwendete Protokolle, die bisherige Anrufdauer und den lokalen SIP-User.

Ausserdem hält die Klasse eine Referenz auf `CallPeerStats`, worin Informationen und Statistiken zu einem bestimmten Peer (Kommunikationspartner) gesammelt werden. Diese sind, gruppiert nach verschiedenen Kriterien, in diversen Klassen abgelegt, welche im Klassendiagramm nicht alle einzeln aufgelistet sind.

Package `net.java.sip.communicator.impl.gui.main.call.stats` In diesem Package liegt die Klasse `StatsCollector`, welche für das Sammeln der eigentlichen Statistiken zuständig ist. Dafür hält sie eine Referenz auf eine `CallConference` (welche eine VoIP-Verbindung kapselt), welche sie im Konstruktor entgegennimmt. Sobald die Methode `collectStats()` aufgerufen wird, iteriert sie durch alle relevanten Informationen der `CallConference`, füllt diese in eine neue Hierarchie von `CallConferenceStats` ab und returniert diese Hierarchie.

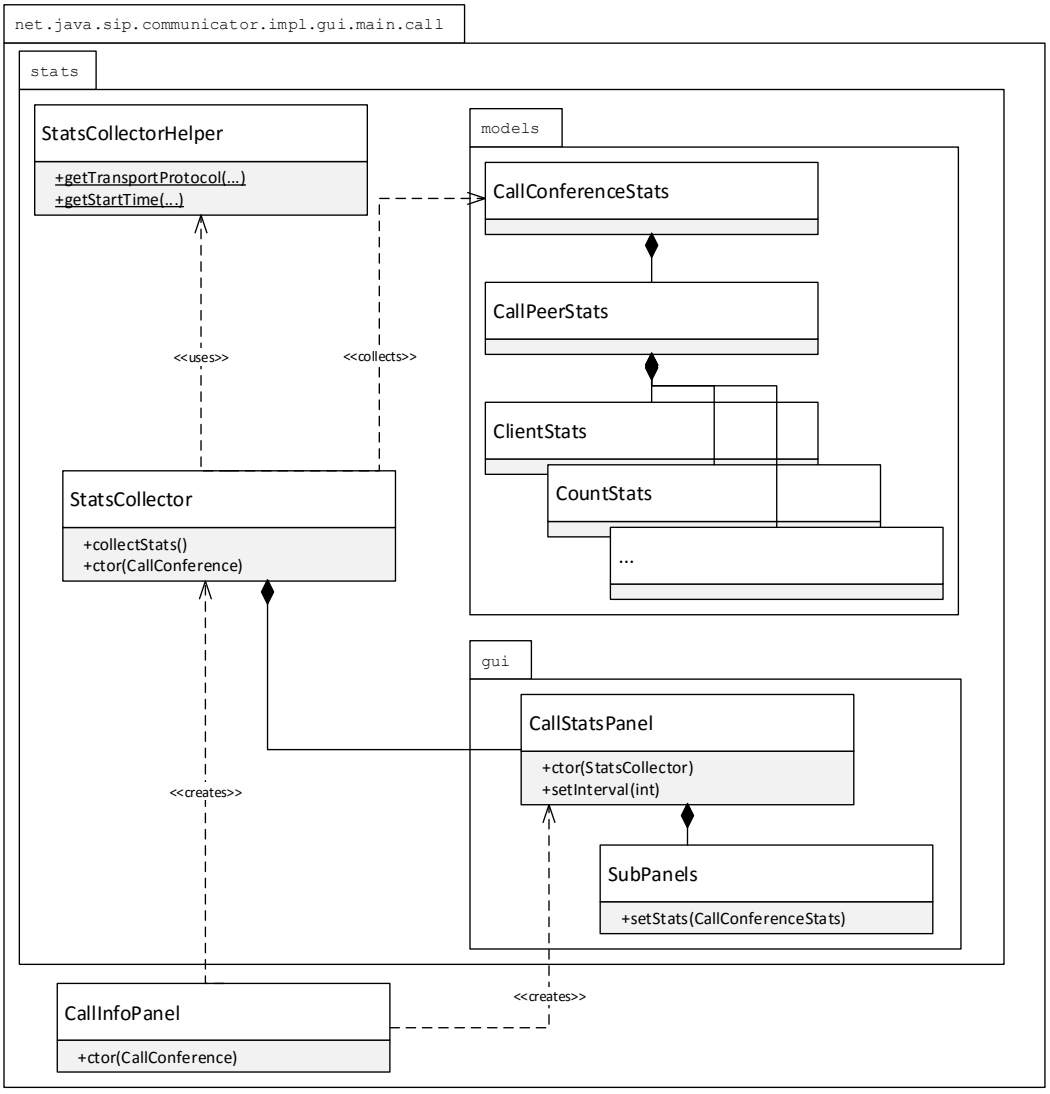


Abbildung 12: Klassendiagramm Jitsi Anpassungen

Die Klasse `StatsCollectorHelper` bietet ausserdem einige statische Hilfsmethoden an, die vom `StatsCollector` verwendet werden.

Package `net.java.sip.communicator.impl.gui.main.call.stats.gui` In diesem Package sind die GUI-Klassen hinterlegt, welche die gesammelten Statistiken darstellen. Die wichtigste Klasse ist hier `CallStatsPanel`, welche im Konstruktor eine Instanz von `StatsCollector` entgegen nimmt.

Das `CallStatsPanel` besteht aus weiteren GUI-Klassen, welche die einzelnen Bereiche des GUIs darstellen. `CallStatsPanel` ruft in regelmässigen Abständen die `collectStats()` Methode von `StatCollector` auf und erhält dafür eine Instanz von `CallConferenceStats` zurück. Diese wird den Sub-Panels mit der Methode `setStats(CallConferenceStats)` weitergeleitet, welche dann die Daten darstellen.

Die Klasse `CallStatsPanel` bietet sie zudem eine öffentliche Methode `setStats(int ms)` an, mit welcher der zeitliche Abstand geändert werden kann, in dem die Statistiken neu geladen werden.

Package `net.java.sip.communicator.impl.gui.main.call` Dieses Package gehört zur Standardauslieferung von Jitsi. Die meisten Klassen in diesem Package wurden nicht angepasst, mit Ausnahme der Klasse `CallInfoPanel`. Diese Klasse zeigte bis anhin statische Informationen über einen Anruf an und wurde nun dahingehend erweitert, dass sie die neuen Echtzeitstatistiken von Jitsi darstellen kann. Die Klasse ist deshalb verantwortlich für die Instanziierung von `StatsCollector` sowie `CallStatsPanel`. Sobald die Instanziierung abgeschlossen ist, hat diese Klasse keine weiteren Aufträge, und wird nur noch als Frame für die GUI-Panels benötigt.

3.5.6 Jitsi Jitter-Buffer

Da die Art und Weise, wie ein Jitter-Buffer implementiert ist, einen relativ grossen Einfluss auf den M2E-Delay haben kann, soll hier kurz erläutert werden, wie sich dieser in Jitsi verhält. Diese Informationen wurden grösstenteils mittels Analysen des Jitsi Sourcecodes erstellt. Da dieser aber relativ komplex ist, mussten einige Annahmen getroffen werden, welche aber mithilfe von Messungen in Abschnitt 4.3.7 bestätigt werden konnten. Der Jitsi Jitter-Buffer ist adaptiv implementiert. Dies bedeutet, dass Jitsi auf sich verändernde Netzwerkbedingungen reagieren kann, und nötigenfalls den Jitter-Buffer erhöht oder verringert.

Im erweiterten Client gibt es 4 Statistiken, die im Zusammenhang mit dem Jitter-Buffer relevant und teilweise auch voneinander abhängig sind. Dies sind

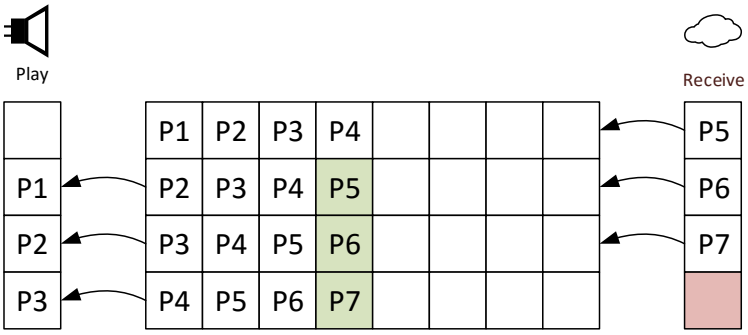


Abbildung 13: Buffer Queue, wenn Pakete regelmässig ankommen

die *Jitter Buffer Queue Size*, die *Packets in Queue*, sowie die *Approx Buffer Size* in Millisekunden, bzw. in Paketen.

Jitsi beginnt jeweils mit einem minimalen Jitter-Buffer, welcher auf 40 ms eingestellt wird. Dies entspricht mit dem G.711 Standard exakt zwei RTP-Paketen. Dies widerspiegelt sich in der Statistik *Approx Buffer Size*. Der Wert kann nicht exakt berechnet werden, kann aber anhand des verwendeten Protokolls, dessen Parameter sowie der aktuellen *Jitter Buffer Queue Size* abgeleitet werden. Es kann aber vorkommen, dass der Wert bei speziellen Protokollen allenfalls nicht korrekt ist, deshalb muss der Wert als eine Annäherung betrachtet werden.

Anhand der angestrebten Dauer des Jitter-Buffers wird dann die Grösse der *Jitter Buffer Queue* konfiguriert. Die Queue kann jeweils doppelt so viele Pakete aufnehmen, wie die Dauer des Jitter-Buffers in Paketen. Soll die Dauer t_{Buffer} des Jitter Buffers z.B. 80 ms betragen und arbeitet der eingesetzte Codec mit G.711 mit einer Framedauer t_{Frame} von 20 ms pro Paket, wird die Buffer Queue mit einer Länge $Size_{Queue}$ von 8 Paketen konfiguriert, wie in Gleichung 5 dargelegt.

$$\begin{aligned} \frac{t_{Buffer}}{t_{Frame}} \cdot 2 &= Size_{Queue} \\ \frac{80ms}{20^{ms/Packet}} \cdot 2 &= 8Packet \end{aligned} \tag{5}$$

Damit passen doppelt so viele Pakete in die Queue, wie die Grösse des Jitter-Buffers ist. Dies hat den Grund, dass der Jitter eine Unregelmässigkeit in der Ankunft der Pakete darstellt. Nochmals zurück zum vorherigen Beispiel mit einer Buffer Queue von 8 Paketen und einem Jitter Buffer von 80 ms pro 4 Paketen. Kommen die Pakete regelmässig an, so sind immer ungefähr 4 Pakete à 20 ms in der Queue. Im perfekten Szenario, welches in Abbildung 13 dargestellt ist, wird

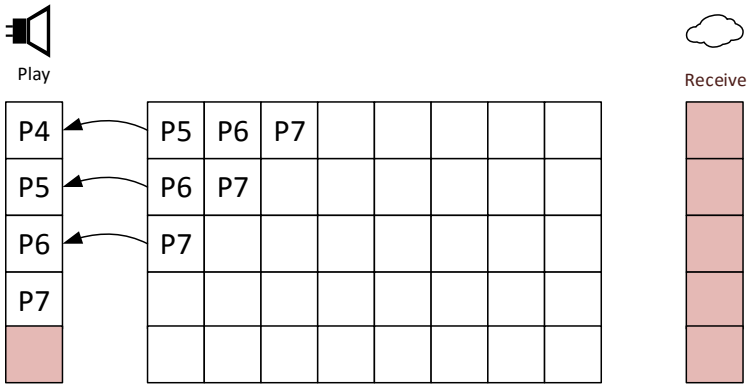


Abbildung 14: Buffer Queue, wenn Pakete verzögert ankommen

jeweils in dem Moment, indem das älteste Paket abgespielt wird, das nächste Paket empfangen und die Queue gelegt. In diesem Szenario verändert sich die Anzahl der Pakete in der Queue wenig bis gar nicht. Meist schwankt die Anzahl der sich in der Queue befindlichen Pakete trotzdem leicht, im gezeigten Beispiel zwischen 3 und 5. Grund dafür ist der Jitter, da so ein Paket mal zu spät und mal zu früh ankommt.

Jetzt kann es aber vorkommen, dass es im Netzwerk zu einer Störung kommt. In Abbildung 14 ist zu sehen, was passiert, wenn aufgrund einer Störung plötzlich keine Pakete mehr ankommen. In diesem Fall spielt Jitsi solange die noch in der Queue vorhandenen Pakete ab, wie sich noch Pakete in der Queue befinden. Sobald aber das letzte Paket *P7* abgespielt wurde, sind keine weiteren Pakete mehr in der Queue vorhanden. Ab diesem Zeitpunkt hört der Benutzer auch keinen Ton mehr. Jitsi wartet nun, bis ihm von der Gegenseite wieder neue Pakete zugestellt werden.

Was in diesem Fall passiert, ist in Abbildung 15 ersichtlich. Als die Störung im Netzwerk verschwindet, kommen die verzögerten Pakete alle gleichzeitig an (mit Ausnahme von *P8*, welches im Netzwerk verloren ging. Jitsi lädt nun sofort alle diese Pakete in die Queue und spielt das erste Paket *P9* sofort ab. Die Pakete *P16* und *P17* kommen ebenfalls noch zusammen an, sozusagen ein Nachspiel der Störung. Danach kommen wieder alle Pakete regelmässig an. In diesem Szenario ist besonders interessant, was passiert, wenn sich die Ankunft der Pakete wieder normalisiert. Waren zu Beginn des Beispiels jeweils 4 Pakete in der Queue, pendelt sich die Queue nun bei 7 Paketen aus. Die Verzögerung aufgrund des Jitter Buffers ist damit nun bei 140 ms.

Daraus wird ersichtlich, dass die eigentliche Dauer des Jitter Buffer nur indirekt gesteuert wird. Wird eine Buffer Dauer von 80 ms angepeilt, wird eine Queue erstellt, welche für 160 ms Platz bietet. Die Pakete werden dann frühestens nach 80 ms abgespielt. Sobald aber mehrere Pakete gleichzeitig ankommen (z.B., weil eine

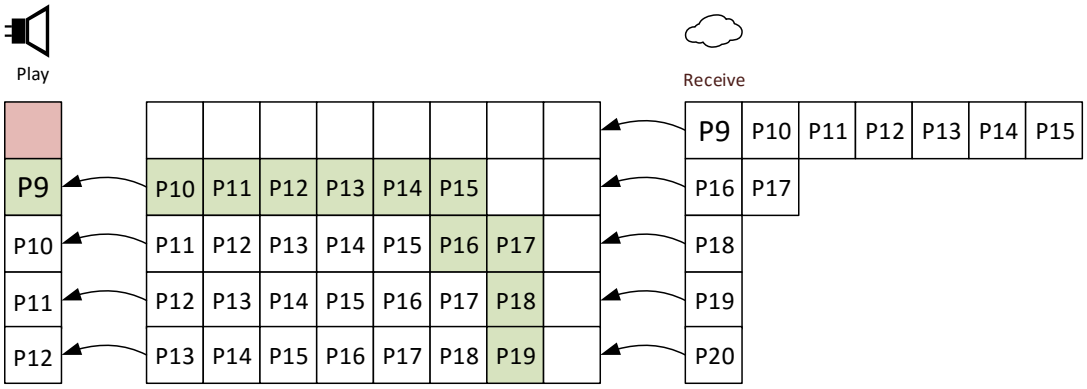


Abbildung 15: Buffer Queue, wenn viele Pakete in kurzen Abständen ankommen

neue Route gewählt wird, bei welcher der Delay kleiner ist, oder wie im Beispiel ein kurzer Netzwerkunterbruch auftritt), kann sich der Buffer vergrössern.

Um den Jitter-Buffer nicht übermässig gross zu halten, obwohl dies nicht nötig wäre, analysiert Jitsi weiterhin die Ankunftszeit der Pakete. Stellt der Client fest, dass die Pakete mit einer hohen Regelmässigkeit eintreffen, wird die Queue nach einer gewissen Zeit reduziert oder zurückgesetzt.

Ein weiteres Szenario ist, dass die Reihenfolge von Paketen im Netzwerk geändert wird, also ein Paket ankommt, bevor das Vorgängerpaket eingetroffen ist, wie in Abbildung 16 dargestellt. Jitsi legt in diesem Fall ein relativ einfaches Verhalten an den Tag. Der Zeitstempel jedes ankommenden Pakets wird mit demjenigen des zuletzt zur Queue hinzugefügten Pakets verglichen. Hätte das neu ankommende Paket vor dem sich in der Queue befindlichen abgespielt werden sollen, wird das Paket verworfen. Das Paket taucht dann als *Late Arrival* in der Statistik auf.

In der Grafik ist ersichtlich, dass sich die Queue genau anders herum verhält, wie bei Abbildung 15. Statt dass sich die durchschnittliche Zahl der *Packets in Queue* vergrössert, wird sie kleiner. Sollte es regelmässig zu solchen *Late Arrivals* kommen, wird Jitsi die Queue deshalb vergrössern. Eine Strategie kann auch sein, ein Paket doppelt abzuspielen, um zu verhindern, dass die *Packets in Queue* zu stark reduziert werden. Dieses Verhalten konnte bei Jitsi aber nicht nachvollzogen werden.

3.5.7 Jitter Buffer Allgemein

Es gibt weitere Möglichkeiten, einen Jitter Buffer zu implementieren. Jitsi hat einen sogenannten *Adaptiven Jitter Buffer* implementiert, womit es dem Client möglich ist, sich den Umständen anzupassen und auf grösseren oder kleineren Jitter im Netzwerk zu reagieren. Damit kann Jitsi den kürzest möglichen Jitter-Buffer wählen,

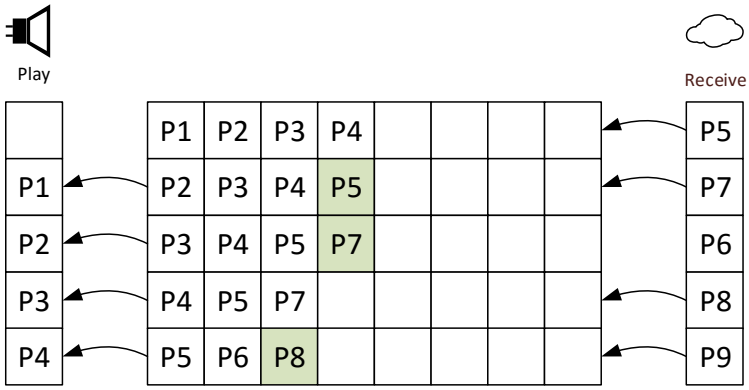


Abbildung 16: Buffer Queue, wenn viele Pakete umsortiert ankommen

bei welchem möglichst keine Pakete verloren gehen.

Adaptive Jitter Buffer können auf zwei Arten implementiert werden. Bei Jitsi wird der Jitter Buffer nach einer gewissen Zeit mit wenig Jitter und ohne Störungen im Netzwerk wieder reduziert. Einige adaptive Jitter Buffer sind jedoch so implementiert, dass der Buffer nur ansteigt. Dies hört sich auf den ersten Moment stabiler an (sollte nach langer Zeit wieder dieselbe Störung auftreten ist der Jitter Buffer bereit), kann aber zu einem unnötig hohen Delay führen, falls es sich um eine einmalige Störung handelte, z.B. aufgrund eines Routenwechsels im Netzwerk.

Eine weitere Option ist es, einfach einen statischen Jitter Buffer zu implementieren. In diesen Fällen wird aber normalerweise von Anfang an ein relativ hoher Jitter-Buffer gewählt, was dazu führt, dass der Jitter Buffer in den meisten Fällen unnötig hoch ist. Dafür ist die Implementation vergleichsweise einfach.

4 Messungen

Mittels den in der theoretischen Analyse gewonnenen Informationen zu den zu behandelnden Codecs, wurde das weitere Vorgehen abgesteckt. Der Ansatz war, mittels einem einfachen Versuchsaufbau zur Messung des M2E-Delays eines VoIP-Gespräches zu beginnen und diesen dann inkrementell zu erweitern, um auch komplexere Szenarien abzudecken.

4.1 Versuchsaufbau

Um auch sicher die vollständige Kontrolle über die gesamte Dauer des VoIP-Gespräches zu erhalten, also vom Verbindungsaufbau über das eigentliche Gespräch, bis hin zum Beenden desselbigen, wurde ein eigener SIP-Server mit Asterisk aufgesetzt,[16] der auf einen Ubuntu 15.10 installiert wurde.

Anschliessend wurde auf zwei weiteren Rechnern je ein SIP-Client installiert. Die ersten Tests wurden dabei mit Zoiper[25] durchgeführt, wobei noch keine Client-Evaluation stattfand. Diese beiden Clients haben sich dann beim SIP-Server registriert. Eine VoIP-Verbindung zwischen den beiden Clients war von nun an möglich.

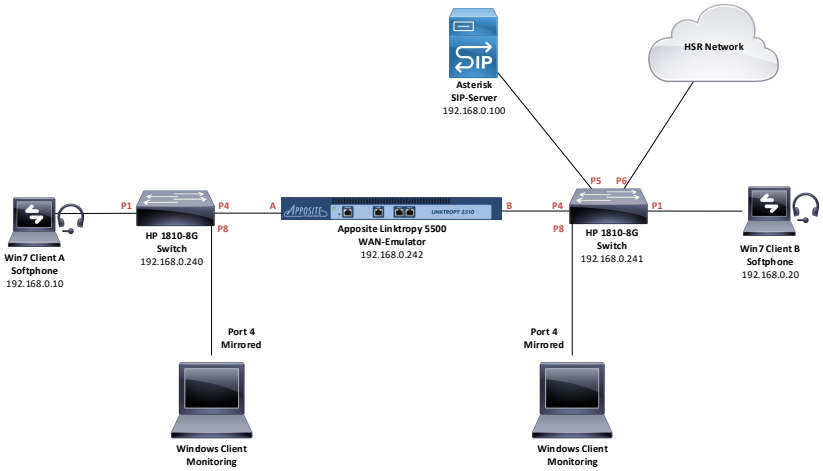


Abbildung 17: Messaufbau im HSR-Netz

Weiter bestand der Anspruch an das Netzwerk, dass einerseits möglichst keine Einflüsse von anderen Netzwerkteilnehmern die Messungen stören sollten, andererseits sollten die Maschinen, die sich im Testaufbau befanden, weiterhin Zugriff auf

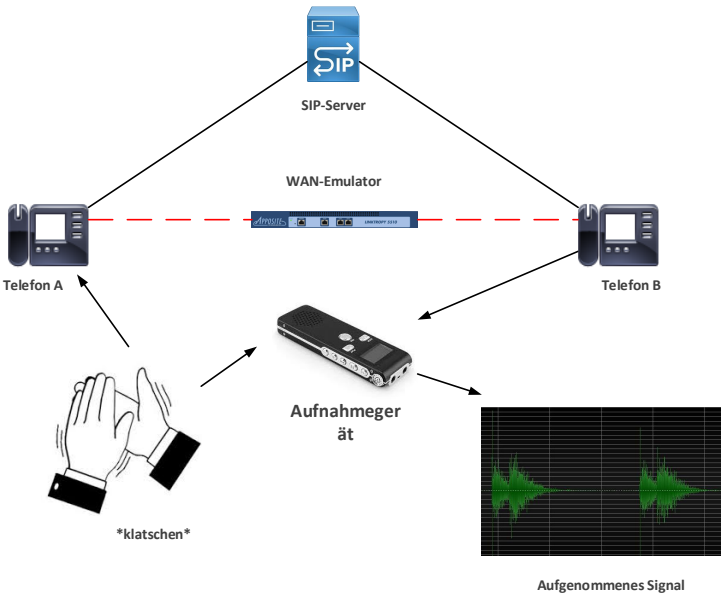


Abbildung 18: Vorgehen für M2E-Messungen

das Internet haben. Trotzdem sollte möglichst viel der HSR eigenen Infrastruktur wiederverwendet werden können.

In Abschnitt 2.3.1 wurde festgestellt, dass nach dem Verbindungsaufbau mittels SIP sämtlicher RTP-Datenverkehr direkt übertragen wird. Somit reicht es, wenn das HSR-LAN mittels eines einfachen Switches mit dem Testnetzwerk verbunden ist. Der Switch sorgt dafür, dass sämtliche Kommunikation zwischen den beiden Clients ohne Umweg über allgemein verwendete Netzwerkinfrastruktur geht. Sowohl die beiden Clients, wie auch der Asterisk-Server, hängen allesamt an diesem Switch, um die Kommunikationswege möglichst gering zu halten. Allfällige Pakete aus dem HSR-Netz, wie ARP- oder DHCP-Requests sind bei diesem einfachen Test mit nicht eingeschränkter Bandbreite ignorierbar.

Um später auch den Einfluss weiterer Netzwerkparameter auf den M2E-Delay zu analysieren, wurde mit dem Apposite Linktropy 5500 ein WAN Emulator zwischen die beiden Clients geschaltet. Mit diesem lassen sich im späteren Verlauf Parameter wie Latenz, Jitter, Packet Loss, Bit Error Rate sowie Netzwerkauslastung konfigurieren, wobei dies in einem ersten Testlauf noch unterlassen wurde. In Abbildung 17 ist die Netzwerktopologie des Versuchsaufbaus beschrieben.

Der M2E-Delay wurde mit einem relativ einfachen Testaufbau gemessen, wie er in Abbildung 18 beschrieben ist. Dazu ruft der eine der beiden Clients den jeweils anderen an. Beim anrufenden Client werden die Lautsprecher, beim Empfänger das Mikrofon stummgeschaltet, um eine Rückkopplung zu vermeiden. Der Lautsprecher

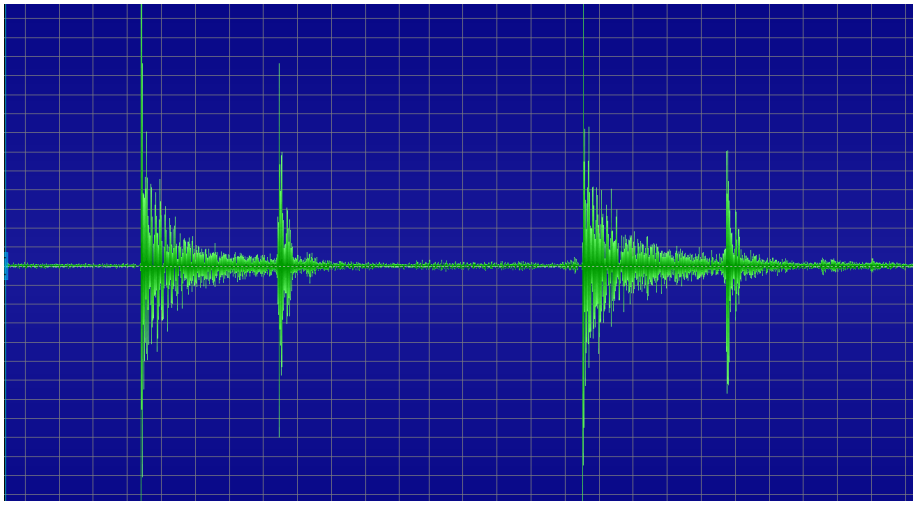


Abbildung 19: Visualisierung einer Audiodatei mit zwei Testgeräuschen. Bei beiden übertragenen Geräuschen sind zwei Spitzen erkennbar, einmal für das eigentliche und einmal für das über die VoIP-Verbindung übermittelte Geräusch.

des Empfängers sowie das Mikrofon des Senders werden nebeneinandergelegt. Es ist aber darauf zu achten, dass sich keine Audiosignale rückkoppeln. Ein gewisser Abstand muss also beibehalten werden.

Ebenfalls neben Lautsprecher und Mikrofon wird ein Aufnahmegerät gelegt. Ein einfaches Smartphone reicht hier vollkommen aus. Auf diesem wird vor dem eigentlichen Test die Aufnahme gestartet. Ausserdem kann an beiden Switches mittels Port-Mirroring der Datenverkehr zwischen den beiden Clients aufgezeichnet werden, womit später auch der Netzwerktraffic analysiert werden kann, sofern dies nötig ist.

Nun steht also der ganze Aufbau. Damit kann ein Anruf getätigt werden. Sobald die Verbindung steht, kann neben dem Mikrofon und dem Aufnahmegerät ein Geräusch erzeugt werden. Je kürzer und knackiger das Geräusch, desto besser lässt sich damit später der M2E-Delay messen.

Das erzeugte Geräusch wird dann sowohl vom Sender wie auch vom Aufnahmegerät aufgenommen, codiert, mittels RTP bis zum Empfänger gesendet, dort decodiert und schlussendlich auf dem Lautsprecher abgespielt. Wenn es beim Empfänger abgespielt wird, wird es auch wieder vom Audiogerät registriert. Die Aufnahme wird nun gespeichert und mittels einem Audioprogramm analysiert. Wie in Abbildung 19 ersichtlich, ist anhand zweier Peaks schön zu erkennen, wann das eigentliche Geräusch abgespielt wurde sowie wann das per VoIP übertragene Geräusch beim Empfänger abgespielt wurde. Der Abstand zwischen zwei Peaks entspricht exakt dem Mouth-To-Ear Delay zwischen den beiden Clients.

Mit diesem Versuchsaufbau können nun diverse Messungen zwischen verschie-

denen Client-System durchgeführt werden. Da es sich beim Asterisk-Server um einen normalen SIP-Server handelt, ist es damit möglich, beliebige SIP-Clients anzuhängen. Dies können Softphones wie im beschrieben Versuch sein, aber auch Hardphones (also SIP-fähige Telefone) oder auch Analog-Telefone sein, welche mittels einem Analog Telephone Adapter angeschlossen werden. Einzige Bedingung, um auch Telefone mit dem SIP-Server zu verbinden, ist eine Nummer zu hinterlegen, die nur aus Zahlen besteht. Solange man ausschliesslich mit Softphones arbeitet, können die Clients auch eine Zeichenkette als "Rufnummer" besitzen.

Schon bei diesen ersten Tests mit dem Softphone Zoiper wurde schnell klar, dass es nicht möglich ist, denn M2E Delay vollständig auf den Algorithmus zu schieben. So ist der theoretische Delay mit dem G.711 Codec rund 30 ms, wobei auf Empfangsseite noch ein Jitter Buffer vorgeschaltet wird, welcher im Bereich von 20 ms bis 100 ms liegen dürfte. Damit wäre die Verzögerung aber nur rund 50 ms bis 130 ms. Die Tests zwischen zwei Zoiper Clients ergaben aber allesamt Delays im Rahmen von 140 ms - 160 ms, wie in Tabelle 11 ersichtlich.

Sender	Empfänger	Messung M2E-Delay [ms]					
		#1	#2	#3	#4	⌀	σ
Zoiper	Zoiper	154	142	157	158	**153**	**6**
Jitsi	Jitsi	198	204	210	200	**203**	**5**

Tabelle 11: Testmessungen mit Zoiper und Jitsi (alle Werte in ms)

Messungen zwischen zwei Jitsi[26]-Clients führten zu noch grösseren M2E-Delays im Bereich von rund 200 ms, dafür sind die absoluten und relativen Unterschiede zwischen den verschiedenen Messungen kleiner. Im Worst Case Szenario gehen rund 130 ms auf den Algorithmus zurück, im Best Case gar nur 50 ms. Mit diesen Informationen fällt auf, dass ein guter Teil des entstehenden M2E-Delays nicht durch den Algorithmus erklärt werden kann. Als einzig mögliche Schlussfolgerung bleibt, dass dieser Unaccounted Delay (Abschnitt 2.1.7) im verwendeten Client entsteht, es bleibt aber die Frage, welchen Einfluss verschiedene Implementationsdetails der Clients auf diesen Delay haben und wie dieser reduziert werden könnte. Diese Frage sollte unter anderem im Abschnitt 3.4 nachgegangen werden.

4.2 Messverfahren

Bisher wurde festgestellt, dass ein guter Teil des messbaren M2E-Delays auf die verwendeten Clients zurückzuführen ist. Um aber deren Einfluss genau messen zu können, muss zuerst analysiert werden, welchen Einfluss verschiedene Netzwerkparameter auf die gemessenen Delays hat.

Zu diesem Zweck wurde der Apposite Linktropy 5500 WAN-Emulator verwendet. Mit diesem lassen sich verschiedene Netzwerkparameter, wie Bandbreite, Latenz, Jitter und Paketverluste konfigurieren, wie in Abbildung 20 ersichtlich.

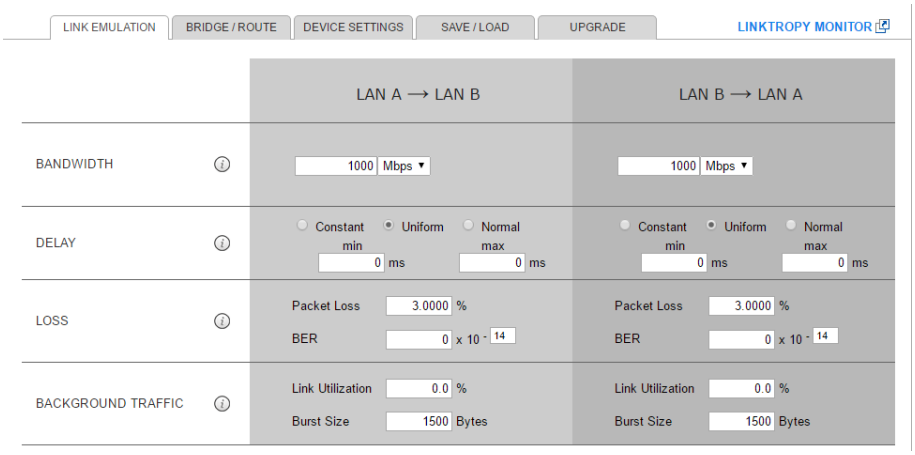


Abbildung 20: Konfigurationsmöglichkeiten des Apposite Linktropy 5500 WAN-Emulators

Es wurden deshalb verschiedene Testreihen definiert, um die Einflüsse dieser Parameter auf den M2E-Delay zu analysieren, welche in Tabelle 12 genauer beschrieben sind.

4.3 M2E-beeinflussende Parameter

In den folgenden Abschnitten wird der Einfluss verschiedener Parameter auf den M2E-Delay analysiert. Zuerst werden einige Messungen ohne weiteren Einflüsse im Netzwerk ausgewertet, um eine Referenz für weitere Messungen zu erhalten. Danach werden verschiedene Netzwerkparameter gesetzt, welche den M2E-Delay beeinflussen sollen

4.3.1 Referenzsystem

Die Auswahl des M2E-Clients ist massgeblich mitverantwortlich, in wieweit sich die M2E-Delays von den theoretisch berechneten Werten unterscheiden. In einem ersten Szenario wurden dabei unterschiedliche Softphones ohne zusätzliche Einflüsse des WAN-Emulators untersucht, also ein Referenzsystem für weitere Messungen. Die Tabelle 13 zeigt, dass es enorme Unterschiede zwischen den verschiedenen Implementierungen der VoIP Clients gibt. Die beobachteten Resultate lassen sich mit grosser Wahrscheinlichkeit auf folgende dynamisch gesetzte Parameter zurückführen und können mit den getesteten Clients nicht manuell beeinflusst werden.

Link Emulation	**Parameter**
Bandbreite	100 kbit/s
Bandbreite	200 kbit/s
Bandbreite	400 kbit/s
Bandbreite	1 Mbit/s
Bandbreite	2 Mbit/s
Bandbreite	10 Mbit/s
Bandbreite	1000 Mbit/s
Latenz	Konstant 100 ms
Latenz	Konstant 200 ms
Latenz	Normalverteilt um 100 ms (Varianz 25 ms)
Latenz	Normalverteilt um 100 ms (Varianz 50 ms)
Latenz	Normalverteilt um 200 ms (Varianz 25 ms)
Latenz	Normalverteilt um 200 ms (Varianz 50 ms)
Paketverluste	1%
Paketverluste	3%

Tabelle 12: Apposite Parameter

- Soundkarten-Buffer
- Jitter-Buffer
- Packetization Delay

Analysiert man erneut die einzelnen Bestandteile, welche für den Delay verantwortlich sein können, ergibt sich ein viel komplexeres Bild. Abschnitt 4.3.2 soll dabei die neuen Erkenntnisse aufzeigen.

Sender	Empfänger	Messung M2E-Delay [ms]					
		# 1	# 2	# 3	# 4	∅	σ
Zoiper	Zoiper	154	142	157	158	**153**	**6**
Jitsi	Jitsi	198	204	210	200	**203**	**5**
X-Lite	X-Lite	104	100	100	104	**102**	**2**
Phoner-Lite	Phoner-Lite	214	238	237	237	**233**	**10**

Tabelle 13: Testmessungen diverser SIP-Clients (alle Werte in ms)

4.3.2 Einflussfaktoren auf M2E-Delay

Abbildung 21 illustriert die Übermittlung von zwei Sprachpaketen über ein IP-Netzwerk.

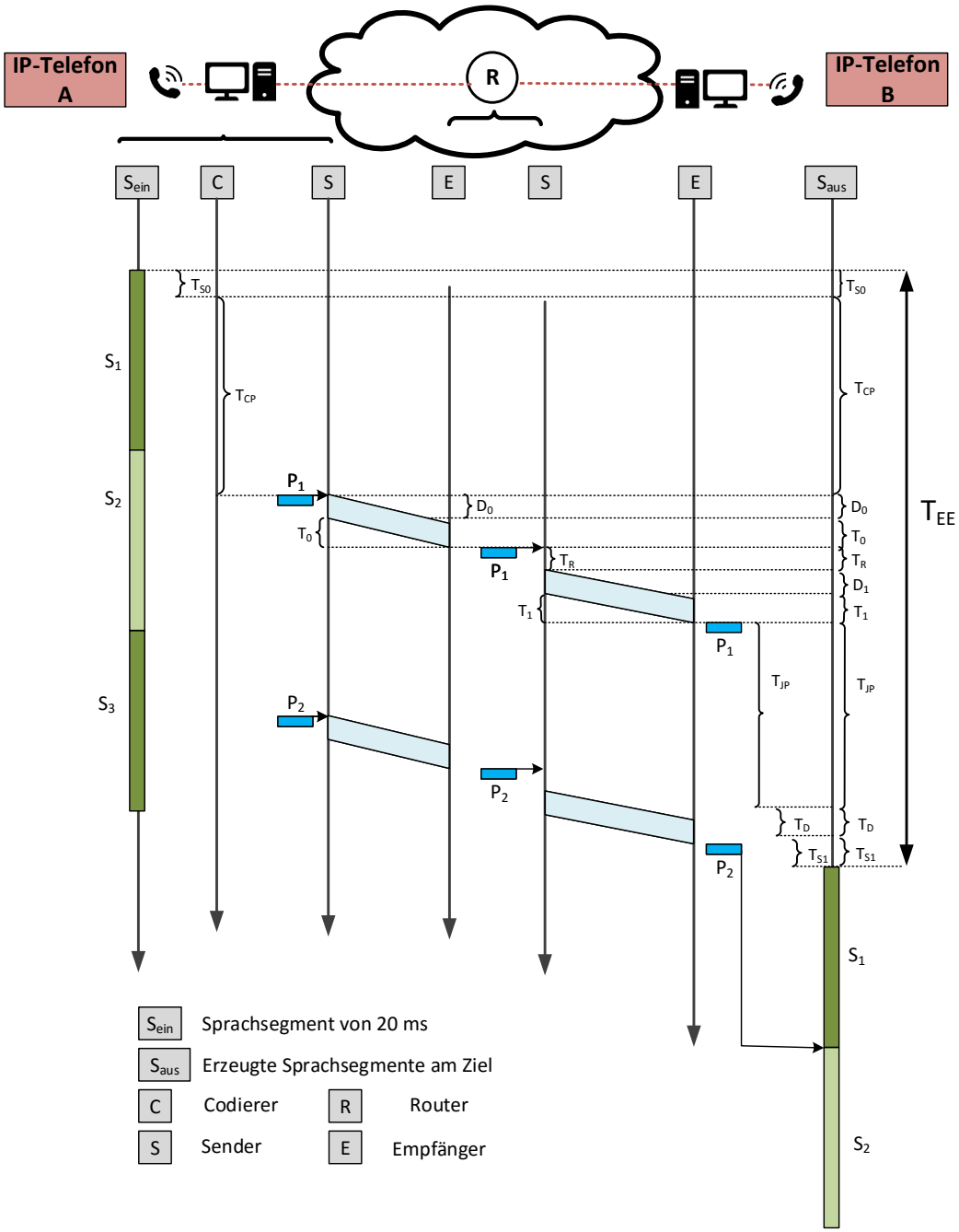


Abbildung 21: Einflussfaktoren für den M2E-Delay

Es sollen hier alle Komponenten eines M2E-Delays erläutert werden. Dabei wird angenommen, dass die Sprachcodierung nach einem segmentorientierten Verfahren stattfindet. Die Verzögerung im Decoder (T_D) wird bewusst vernachlässigt, da der Wert nicht ins Gewicht fällt.

Im Testlabor konnte mithilfe von Wireshark-Auswertungen bestimmt werden, dass es sich beim Codierungs- und Paketierungsvorgang (T_{CP}) um den theoretisch angenommenen Wert von 20 ms handelt. Dazu kann in der RTP-Analyse (Wireshark) das Delta betrachtet werden. Unbekannt bleibt weiterhin der genaue Delay der Soundkarte (T_{S0}). Ausgehend vom Gesamtdelay von ca. 104 ms (X-Lite), werden Zeiten von jeweils 0 ms - 20 ms vermutet.

Die Serialisierungsverzögerung (D_0) von 17.12 ms (100 kbit/s Leitung) konnte mit einer Messung vor und nach dem WAN-Emulator bestätigt werden. Dabei wurde die *Arrival Time* desselben Datenpakets verglichen.

Einzelne Signallaufzeiten (T_0 bzw. T_1) können vernachlässigt werden und fallen bei geringen Distanzen von wenigen Metern nicht ins Gewicht. Ansonsten wird als grobe Schätzung oftmals die Verzögerung von 1 ms pro 200 km angenommen. Was der Geschwindigkeit von $2/3$ des Lichtes entspricht.

Mit (T_R) ist eine zufällige Zeit gemeint, welche das Datenpaket innerhalb des Netzwerkgerätes verbringt. Für gewöhnlich kann dieser Wert ignoriert werden, da er keinen spürbaren Einfluss auf das Gesamtresultat hat.

Die beim Empfänger entstehenden Delays vom Jitter-Buffer (T_{JP}) und Soundkartenbuffer (T_{P1}) bleiben weiterhin unbekannt. Vermutet werden ähnliche Zeiten wie beim Sender.

Auch weitere Faktoren, die von der Implementation der eingesetzten Clients abhängen, bleiben unbekannt und können mit reinen M2E Messungen nicht analysiert werden.

4.3.3 Einfluss von unterschiedlichen Bandbreiten

Bei sehr hohen Bandbreiten hat der Serialization Delay, wie im Abschnitt 2.1.3 beschrieben, nur einen sehr geringen Einfluss auf den Gesamtdelay. Erst bei Netzwerkgeschwindigkeiten unterhalb von 2 Mbit/s, kommt man in den Millisekundenbereich. Und auch da liegt der theoretische Höchstwert, ausgegangen von einer 100 kbit/s Verbindung, bei ca. 17.12 ms. Betrachtet man nun die Testresultate, wie in der Abbildung 22 zusammengefasst, zeigt sich ein auf den ersten Blick überraschendes Bild. Die ersten zwei M2E-Delay-Resultate sind relativ hoch ausgefallen und entsprechen nicht den Erwartungen. Die Tabelle 14 zeigt, dass es in den Testresultaten zu starken Schwankungen kommt. Woher kommen also diese Delays bei 100 kbit/s bzw. 200 kbit/s? Analysiert man parallel zur Audioaufnahme die Wireshark Aufzeichnungen von den Monitoring Computern, ist ersichtlich, dass im Netzwerk nicht ausschliesslich RTP-Pakete des aktiven Telefongesprächs übertragen

werden. Es reicht demnach bereits aus, dass hin und wieder andere Protokolle, wie ARP oder SIP, das Netzwerk belasten. Dies führt einerseits dazu, dass Warteschlangen in den Netzwerkgeräten entstehen und anderseits, dass aufgrund von hohen Serialisierungszeiten Latenzschwankungen entstehen. Dieser Jitter veranlasst den VoIP Client dazu, denn Jitter-Buffer um einige 10 bis 100 ms zu erhöhen.

Sender	Empfänger	Bandbreite	Messung M2E-Delay [ms]					
			#1	#2	#3	#4	∅	σ
X-Lite	X-Lite	100 kbit/s	134	200	180	245	**190**	**40**
		200 kbit/s	142	134	138	143	**139**	**4**
		400 kbit/s	120	108	133	115	**119**	**9**
		1 Mbit/s	114	121	110	115	**115**	**4**
		2 Mbit/s	114	106	111	108	**110**	**3**
		10 Mbit/s	105	109	107	109	**108**	**2**
		1000 Mbit/s	105	107	101	104	**104**	**2**

Tabelle 14: Testmessungen mit unterschiedlichen Bandbreiten (alle Werte in ms)

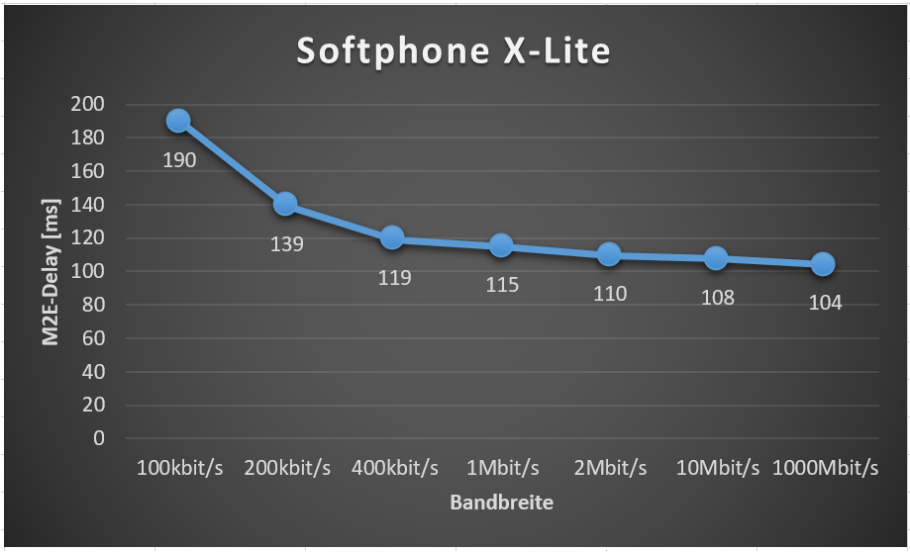


Abbildung 22: M2E-Delay mit unterschiedlichen Bandbreiten

4.3.4 Einfluss von erhöhter Latenz

Um die Auswirkungen eines konstanten Delays zu untersuchen, wurden Tests mit zwei unterschiedlichen VoIP-Clients vorgenommen. Der künstlich gesetzte Delay lag dabei bei 0 ms, 100 ms oder 200 ms. Die Tabelle 15 verdeutlicht, dass der

M2E-Delay genau um den manuell gesetzten Delay erhöht wird. Somit hat ein konstanter Delay keinerlei Auswirkung auf den Jitter Buffer.

Sender	Empfänger	Konst. Delay	Messung M2E-Delay [ms]					
			#1	#2	#3	#4	**Ø**	**σ**
Zoiper	Zoiper	0 ms	154	142	157	158	**153**	**6**
		100 ms	252	247	245	252	**249**	**3**
		200 ms	351	353	359	355	**355**	**3**
Jitsi	Jitsi	0 ms	198	204	210	200	**203**	**5**
		100 ms	317	313	300	303	**308**	**7**
		200 ms	413	412	396	412	**408**	**7**

Tabelle 15: Testmessungen mit konstantem Delay (alle Werte in ms)

Kleinere Qualitätseinbussen entstehen erst, wenn der fixe Delay wieder entfernt wird. Es kommt zu einem Burst, was aktuelle Datenpakete mit verzögerten Paketen vermischt (was sich anhand der Sequenznummer nachvollziehen lässt - Abbildung 23). Je nach Client und Grösse des Jitter-Buffers, werden die aktuellen Sprachdaten bevorzugt und die restlichen Datenpakete verworfen.

Paket	Sequenz	Delta (ms)	Jitter (ms)	Versatz	Bandbreite	Marker	Status
7895	42249	20.17	5.45	-95.93	80.00		✓
7897	42250	19.87	5.12	-95.80	81.60		✓
7901	42251	30.09	5.43	-105.89	80.00		✓
7903	42252	9.98	5.72	-95.87	81.60		✓
7904	42257	0.03	11.61	4.10	83.20		Wrong sequence number
7907	42253	19.99	17.13	-95.88	83.20		Wrong sequence number
7909	42258	10.00	21.69	-5.88	83.20		✓
7911	42259	9.84	20.97	4.28	84.80		✓
7912	42254	0.03	25.91	-95.75	86.40		Wrong sequence number
7913	42260	20.08	30.53	4.17	86.40		✓
7915	42255	10.08	35.51	-105.91	86.40		Wrong sequence number
7918	42256	9.93	33.92	-95.85	88.00		Wrong sequence number
7919	42261	0.05	38.04	4.11	89.60		✓
7922	42262	29.94	36.29	-5.83	88.00		✓

Abbildung 23: Fehlerhafte Sequenznummern nach einem Burst

4.3.5 Einfluss von Paketverlusten im Netzwerk

In einem Netzwerk können jederzeit Paketverluste auftreten und sind per se nicht ungewöhnlich. Als Ursache können überlastete Netzwerkknoten oder ein schlecht dimensionierter Jitter-Buffer in Frage kommen. Inwiefern dies den M2E-Delay beeinflusst, soll anhand von Messungen betrachtet werden.

Aus Tabelle 16 geht hervor, dass Paketverluste auf den reinen M2E-Delay keinerlei Auswirkungen haben. Fehlende Pakte werden übersprungen und der Audiostream wird beim nächsten zur Verfügung stehenden Paket weitergeführt. Hingegen werden Unterbrechungen im Telefongespräch hörbar, falls sich die Paketverluste in einer bestimmten Zeitperiode häufen.

Sender	Empfänger	Paketverlust	Messung M2E-Delay [ms]					
			#1	#2	#3	#4	⌀	σ
Zoiper	Zoiper	0%	154	142	157	158	**153**	**6**
		1%	138	148	144	128	**140**	**8**
		2%	138	145	139	135	**139**	**4**

Tabelle 16: Testmessungen bei Paketverlusten im Netzwerk (alle Werte in ms)

4.3.6 Einfluss von SRTP

Secure Real-Time Transport Protocol (SRTP) ist eine Erweiterung von RTP und wird benutzt, um die Vertraulichkeit, Integrität und Replay-Schutz eines Audio- bzw. Video-Streams sicherzustellen.[13] Dabei wurde untersucht, wie sich der M2E-Delay bei einer gesicherten VoIP-Verbindung verhält. Als Softphone Clients kamen die Lösungen aus Abbildung 9 zum Einsatz, welche das Zimmermann Real-time Transport Protocol (ZRTP) unterstützen. Die SRTP Transformationen vom Zoiper Client basierte auf einer AES-Counter-Modus-Verschlüsselung AES-CM 128 Bit und einem Authentifizierung Tag von HMAC-SHA1 32 Bit. Beim zweiten Client, Phoner-Lite, wurde AES-CM 128 Bit mit HMAC-SHA1 80 Bit verwendet.

Aus der Tabelle 17 geht hervor, dass sich der zusätzliche Delay durch die Verschlüsselung sehr tief hält. Im Durchschnitt gab es lediglich einen Delay-Anstieg von 7 ms bzw. 9 ms.

Sender	Empfänger	Protokoll	Messung M2E-Delay [ms]					
			# 1	# 2	# 3	# 4	⌀	σ
Zoiper	Zoiper	RTP	154	142	157	158	**153**	**6**
		SRTP	145	149	171	175	**160**	**13**
Phoner-Lite	Phoner-Lite	RTP	214	238	237	237	**232**	**10**
		SRTP	243	240	242	240	**241**	**1**

Tabelle 17: Testmessungen mit und ohne Verschlüsselung (alle Werte in ms)

4.3.7 Einfluss von Jitsis Jitter-Buffer

In diesem Abschnitt soll analysiert werden, welchen Einfluss der Jitter-Buffer von Jitsi auf den M2E-Delay hat. Aufgrund der geschlossenen Implementationen der anderen Clients, und auch, weil bei anderen Clients zu wenige interne Statistiken erstellt werden, wurden diese Analysen auf Jitsi eingeschränkt.

In einem ersten Schritt soll aufgezeigt werden, inwiefern sich die *Packet Queue* auf den M2E-Delay auswirkt. Dazu wird mittels WAN-Emulator der Delay kurzfristig soweit erhöht, dass sich der Client veranlasst fühlt, seine *Packet Size* anzupassen. Dies kann erreicht werden, indem ein fester Delay von beispielsweise 100 ms während rund 5 Sekunden aktiv geschaltet wird. Mit steigender *Queue Size* wird dabei ein erhöhter M2E-Delay erwartet.

Queue Size	Messung M2E-Delay [ms]					
	#1	#2	#3	#4	⌀	σ
4	225	313	225	298	**265**	**41**
6	220	233	224	219	**224**	**6**
8	240	252	263	278	**258**	**14**
10	279	312	283	278	**288**	**14**
12	299	333	303	297	**308**	**14**

Tabelle 18: Testmessungen erweitertem Jitsi-Client (alle Werte in ms)

Die ausgewerteten Messungen in Tabelle 18 stützen diese These. Der Jitter-Buffer steigt mit zunehmender *Queue Size*. Der einzige Ausreisser ist in der ersten Zeile zu finden. Obwohl zu diesem Zeitpunkt noch keinerlei Einstellungen am WAN-Emulator vorgenommen wurden, scheint die Ausgangslage sehr instabil zu sein. Genaue Rückschlüsse über dieses Verhalten konnten nicht ermittelt werden. Der verwendete Jitsi-Client befindet sich derweil noch in Entwicklung und ein Softwarebug kann nicht ausgeschlossen werden.

Für den Normalfall kann aber für die Berechnung des Jitter Buffers folgende Formel angenommen werden. Dabei würde der Delay bei einer *Queue Size* von 10 Paketen und einem G.711 Codec 100 ms betragen.

$$\frac{(\mathit{Queue\ Size} \cdot \mathit{Packetization\ Delay})}{2}$$

Dynamischer Jitter-Buffer Beobachtungen während der Messphase haben aufgezeigt, dass es sich beim Jitsi um einen dynamischen Jitter Buffer handelt. Je nachdem wie oft und für wie lange sich die Varianzen ändern, verhält sich der VoIP-Client anders. Wie sich das Verhalten manifestiert, soll mit verschiedenen

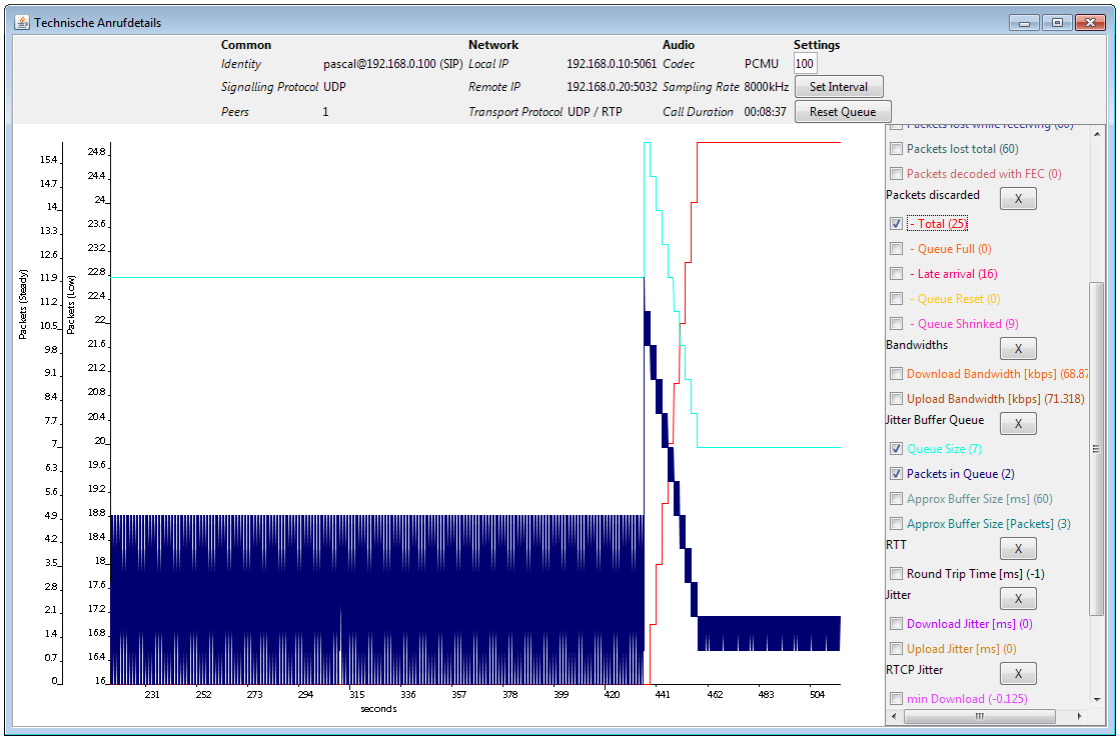


Abbildung 24: Erholung des Jitter-Buffers

Versuchen veranschaulicht werden. Dabei ist interessant zu erfahren, wie lange es dauert, bis der Jitter-Buffer wieder seinen Ursprungswert (*Queue Size* von 4) oder zumindest einen tieferen Wert erlangt.

Begonnen wird mit einem fixen Delay von 50 ms über eine Zeitdauer von 5 Sekunden. Nachdem sich eine mögliche Veränderung abzeichnet, wird darauf gewartet, bis sich der Wert normalisiert. Ist das geschehen, wird der Delay erneut aktiviert. Es gilt dabei zu erfahren, ob sich die *Queue Size* bei mehrmaligen Unregelmässigkeiten intelligent verhält.

Anders als erwartet, werden fixe Delays bis zirka 70 ms von Jitsi vollständig ignoriert. Dabei spielt es keine Rolle, ob die künstlich erzeugte Verzögerung für eine sehr kurze oder lange Zeitdauer eingeschaltet wird. Bei höheren Delay-Werten bewirkt die Verzögerung, dass sich die *Queue Size* um jeweils 2 erhöht. Jedoch ist keine Normalisierung in absehbarer Zeit (unter 5 Minuten) ersichtlich. Erst bei einer längeren stabilen Periode verringert sich der Jitter Buffer. Dabei pendelt sich der Wert bei einer *Queue Size* von 7 ein (Abbildung 24).

Weitaus dynamischer gestaltet sich der Jitter Buffer bei Unregelmässigkeiten im Delay. Der VoIP-Client regiert weitaus schneller auf neue Anforderungen als bei fixen Delays. Eine Verringerung des Jitter-Buffers hat aber zur Folge, dass einzelne

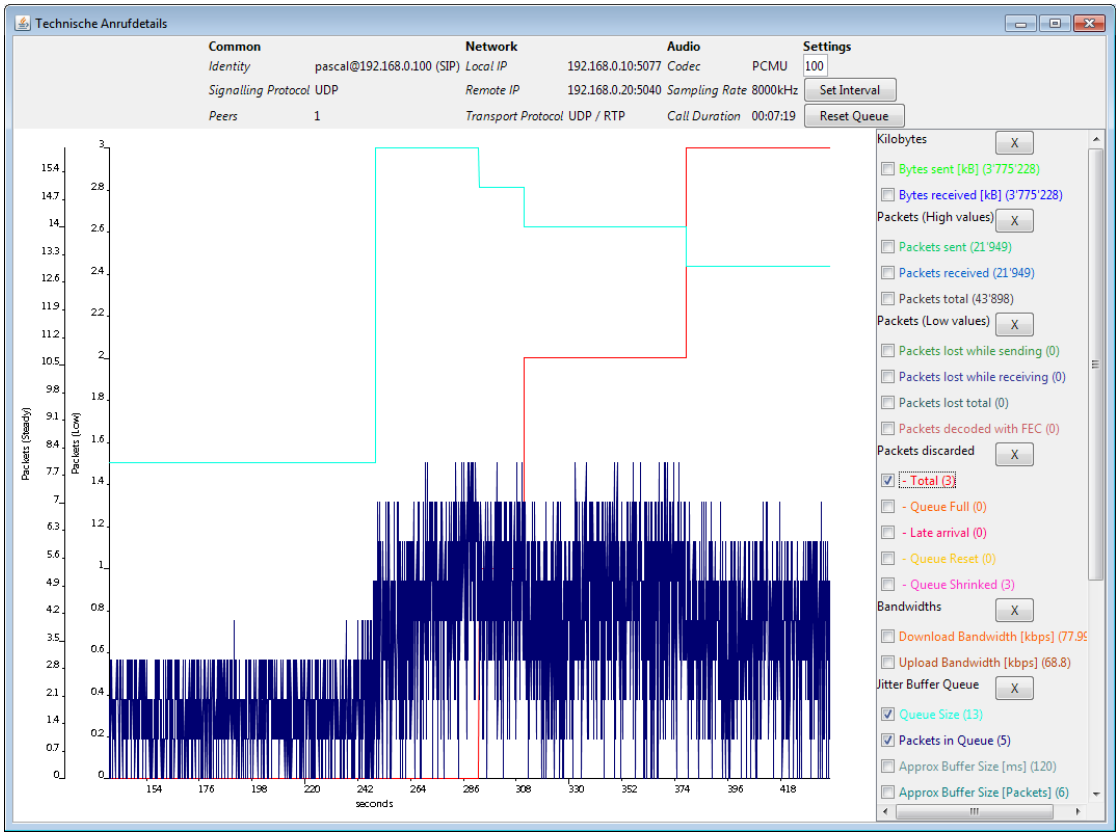


Abbildung 25: Dynamischer Jitter-Buffer

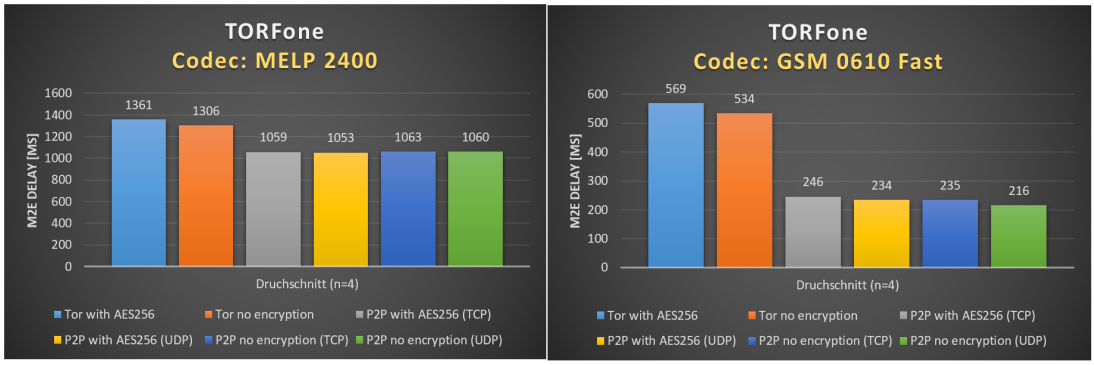
Pakete verworfen werden. Dabei verdeutlicht Abbildung 25, wie das Zusammenspiel zwischen *Packets discarded* (roter Graph) und *Queue Size* (hellblauer Graph) ausfällt. Sinkt der eine Graph mit der *Queue Size*, steigt der andere an.

4.3.8 TORFone Messungen

Standardmässig verwendet TORFone für die Sprachkomprimierung den Mixed-Excitation Linear Predictive (MELP) Codec. Dieser wurde speziell dazu entwickelt, die menschliche Stimme mit sehr niedrigen Bitraten von 600, 1200 oder 2400 Bit/s verständlich wiederzugeben.[24] Bei einer Bitrate von 2400 wird nach 22.5 ms ein Sprach-Frame erstellt. Die Abtastrate beträgt 8000 Hz und die Payload beträgt 54 Bits.

Falls der Anwender über einen Breitbandanschluss verfügt oder sich im lokalen Netzwerk befindet, wird empfohlen, den klassischen GSM 6.10 Standard zu verwenden.

Die Auswertungen der Messungen sind in Abbildung 26 ersichtlich. Dabei gibt



(a) MELP (b) GSM

Abbildung 26: M2E-Delay bei TORFone mit verschiedenen Codecs

es jeweils 6 Kategorien die miteinander verglichen werden. M2E-Messungen mit dem Tor-Netzwerk, ohne dem Tor-Netzwerk, mittels TCP oder UDP und jeweils mit und ohne Verschlüsselung.

Der Grossteil des Delays entsteht durch das Tor-Netzwerk. Die Verschlüsselung und die Wahl des Transportprotokolls beeinflussen die Werte nur geringfügig. Ausserdem ist das Verwenden eines modernen Codecs sehr ratsam.

5 Schlussfolgerungen

Im Folgenden sollen nochmals die verschiedenen Ursachen für M2E-Delays zusammengefasst, sowie Empfehlungen abgegeben werden, die bei der Inbetriebnahme eines VoIP-Netzwerks zu berücksichtigen sind.

5.1 Ursachen

Erstaunlicherweise war es in allen Tests meistens nicht das Netzwerk, welches den grössten Einfluss auf die M2E-Delays hatte, sondern der verwendete Client. Einer der verwendeten Clients hatte ohne weiteren Netzwerkeinflüsse zwar vergleichsweise tiefe Delays von nur rund 100 ms. Je nach verwendetem Client stiegen diese Delays aber auf über 300 ms an. Zum Vergleich: Geht eine VoIP-Verbindung über einen Satelliten, kommt ein *One-Way-Delay* von rund 360 ms hinzu. Wird ein suboptimaler Client verwendet, kann sich diese Laufzeit also beinahe nochmals verdoppeln.

Aber auch verschiedene Netzwerkparameter können einen messbaren Einfluss auf den gemessenen M2E-Delay haben. Der Einfluss des *Network Delays* ist soweit eindeutig und hat einen linearen Einfluss auf den M2E-Delay. Je länger es geht, ein Signal von A nach B zu transportieren, desto grösser wird der gemessene Delay.

Interessanter sind andere Einflüsse. Die Bandbreite hat, sofern genügend bemessen, nur einen geringen Einfluss auf die gemessenen Verzögerungen. Normalerweise entspricht dies einfach dem zusätzlichen *Serialization Delay*, welcher aber höchstens wenige Millisekunden beträgt. Kommt die verfügbare Bandbreite aber näher an das für den Codec benötigte Minimum, steigen die Verzögerungen stark an. Bereits bei der doppelten verfügbaren Bandbreite (in beide Richtungen) steigt der M2E-Delay merklich an.

Ebenfalls einen Einfluss hat der in der Übertragung entstehende Jitter, wobei dieser von der Implementation im Client abhängt. Die verwendeten SIP-Clients verwenden alle unterschiedliche Implementationen für den Jitter Buffer. Manche sind statisch, andere haben einen adaptiven Jitter-Buffer. Bei Clients mit einem statisch konfigurierten Buffer hat der Jitter keinen weiteren Einfluss auf die gemessenen Verzögerungen. Clients mit adaptivem Jitter-Buffer auf der anderen Seite können recht empfindlich auf Jitter reagieren, wenn der Client hohe Jitter-Anteile bemerkt. Beim Beispiel Jitsi kann der Jitter-Buffer z.B. von normalerweise 20 ms auf bis zu 80 ms ansteigen. Wenn zusätzlich das Netzwerk knapp bemessen ist oder grundsätzlich eine hoher *One-Way-Delay* vorhanden ist, kann dies einen merkbaren unterschied bedeuten.

Das verwendete System kann unter Umständen auch einen ganz kleinen Einfluss haben, dieser ist in den meisten Fällen aber vernachlässigbar. So kann der

Delay leicht ansteigen, wenn das System unter hoher Last steht. Bei modernen Systemen sind diese zusätzlichen Verzögerungen aber in der Regel im Submillisekundenbereich. Bei älteren Systemen, oder einfach schwach bemessene Embedded-Plattformen, könnte eine Überlastung des Systemes aber tatsächlich einen messbaren Einfluss haben. Da Embedded Systeme aber meistens einen bestimmten Zweck erfüllen, und keine besondere zusätzliche Software darauf läuft, ist dies auch hier in den meisten Fällen zu vernachlässigen.

5.2 Empfehlungen

Aufgrund der während der Arbeit gewonnenen Erkenntnisse, können einige Empfehlungen abgegeben werden.

Mit Abstand am wichtigsten ist die Auswahl des richtigen SIP-Clients oder des verwendeten Telefons. Kein anderer Parameter, auf den Einfluss genommen werden kann, hat einen vergleichbaren Einfluss auf die gemessenen M2E-Delays. Sind tiefe Delays gewünscht, ist es somit von grosser Wichtigkeit, vor der Inbetriebnahme verschiedene Systeme detailliert zu evaluieren und Messungen des M2E-Delays durchzuführen. An diesem Punkt sollte auf keinen Fall gespart werden.

Weiter ist wichtig, dass genügend Bandbreite auf der Leitung vorhanden ist. Bei den durchgeführten Tests hat sich herausgestellt, dass in jede Richtung etwas mehr als die doppelte Bandbreite bereitstehen sollte, um für eine stabile Verbindung zu sorgen. Auch in diesem Fall darf aber nicht zu viel Fremdverkehr in der Leitung sein. Ein gutes QoS kann dieses Problem aber zumindest reduzieren. Ist die verfügbare Bandbreite kleiner, ist damit zu rechnen, dass der M2E-Delay stark ansteigt.

Ebenfalls ist darauf zu achten, dass im Netzwerk kein grosser Jitter entsteht, besonders wenn ein Client mit adaptiven Jitter Buffer verwendet wird. Bei einem statischen Jitter Buffer hat der Jitter keinen Einfluss auf den Delay, allerdings kann die Sprachqualität bei sehr starkem Jitter abnehmen. Jitter kann in einem Netzwerk am besten reduziert werden, indem verschiedene QoS Massnahmen eingesetzt werden, sodass VoIP-Pakete mit höherer Priorität verschickt werden, und nicht durch Fremdverkehr gestört werden.

Gibt es trotz dieser Massnahmen Probleme mit dem M2E-Delay, so kann der entwickelte Jitsi-Test Client verwendet werden, um eigene Analysen durchzuführen. Damit können allfällige Probleme im Netzwerk erkannt werden, was die Behebung von Problemen vereinfacht.

5.3 Schlusswort

Es gibt viele verschiedene Ursachen für M2E-Delays in einer VoIP-Verbindung, und es ist nicht immer einfach, die genauen Ursachen herauszufinden. In dieser Arbeit

wurden einige Faktoren besprochen, die einen Einfluss auf diese Delays haben können. Es wurde auch besprochen, wie solche Faktoren reduziert werden können, sowie Mittel entwickelt, diese Faktoren zu erkennen.

Mit diesem Know-How und den entwickelten Hilfsmitteln sollte es einem Netzwerkadministrator möglich sein, die bestmöglichen Clients auszuwählen sowie das Netzwerk möglichst optimal zu konfigurieren, um eine bestmögliche VoIP-Performance zu erreichen.

Einen genaueren Blick sollte dieser Netzwerkadministrator auf die eingesetzten Clients richten. Es ist nicht immer einfach, den richtigen Client auszuwählen, und die in dieser Arbeit durchgeführten Analysen sind dahingehend bestimmt nicht abschliessend. Z.B. wurden nur Clients analysiert, die Kostenfrei verfügbar sind, wobei es gut möglich ist, dass es Lösungen gibt, die in der Hinsicht des entstehenden M2E-Delays effizienter wären, aber auch teurer.

Sollten tiefe bis keine Kosten ein wichtiger Faktor sein, kann X-Lite empfohlen werden. Dieser Client hat sich durchgehend durch eine ausgezeichnete Performance hervorgehoben und hat nur einen sehr kleinen Einfluss auf den M2E-Delay. Leider bot er aber keine Verschlüsselung an, was ihn gerade in professionellen Einsatzgebieten eher uninteressant macht.

Ist Sicherheit aber ein wichtiger Faktor, kann entweder Zoiper oder Jitsi empfohlen werden. Jitsi hat in den Tests zwar den grösseren Einfluss auf den M2E Delay als Zoiper, unterstützt aber alle heutzutage relevanten Verschlüsselungsprotokolle, inklusive ZRTP. Ist ZRTP aber nur von geringer Bedeutung, ist eher Zoiper zu empfehlen, da dieser die bessere Performance als Jitsi bietet.

Appendix

A Berechnungen

A.1 Propagation Delays

Im Folgenden soll berechnet werden, wie lange es dauert, um ein Signal von der Erde zu einem Geostationären Satelliten zu übertragen. Dabei werden folgende Parameter angenommen:

Geschwindigkeit Kommunikationssignal Vakuum: $v_v = c \approx 300^{km}/_{ms}$
Geschwindigkeit Kommunikationssignal Medium: $v_a = \frac{2}{3} v_v \approx 200^{km}/_{ms}$
Geostationärer Erdorbit: $h = 36 \cdot 10^3 km$

Da die Geschwindigkeit des Kommunikationssignals zwischen der Erdoberfläche (in der Atmosphäre) und der Position des Kommunikationssatelliten (Vakuum) stark variiert, wird für die RTT-Berechnung ausschliesslich der Wert v_a verwendet, wodurch die berechnete RTT einen Worst Case Wert darstellt.

Die Zeit t_{Signal}, die ein Kommunikationssignal von der Erde bis in den geostationären Orbit benötigt lässt sich wie folgt berechnen:

$$
\begin{aligned}
t_{Signal} &= \frac{h}{v_a} \\
t_{Signal} &= \frac{36 \cdot 10^3 km}{200^{km}/_{ms}} \\
t_{Signal} &= 180 ms.
\end{aligned}
$$

Die Latenz t_{Latenz}, also die Zeit, die es dauert, bis das Signal beim Kommunikationspartner ankommt, ist das doppelte von t_{Signal}, also

$$
\begin{aligned}
t_{Latenz} &= 2 \cdot t_{Signal} \\
t_{Latenz} &= 2 \cdot 180 ms \\
t_{Latenz} &= 360 ms,
\end{aligned}
$$

während die Zeit, bis eine Antwort vom Gegenüber erwartet werden kann, die t_{RTT} somit beim vierfachen von t_{Signal} liegt. Damit

$$t_{RTT} = 4 \cdot t_{Signal}$$
$$t_{RTT} = 4 \cdot 180ms$$
$$t_{RTT} = 720ms,$$

A.2 Interarrival Jitter

Das folgende Beispiel zeigt die Berechnung des *Interarrival Jitter* anhand einer realen Wireshark Aufzeichnung. Der verwendete Codec ist PCM mit einer Abtastrate von 8000 Hertz.

In Tabelle 19 sind die Wireshark Timestamps der verwendeten Frames aufgezeichnet. Die *Frame Arrival Time* entspricht dem von der Netzwerkkarte gemessenen Zeitstempel. Umgewandelt in Millisekunden ergibt dies den Wert für R_i. Ausserdem wurde der Wert um 1 458 481 000 000 ms gekürzt, damit die Übersichtlichkeit nicht leidet. Der *RTP Timestamp* entspricht der Position des ersten Audiosamples im RTP Paket, ist also nur eine Durchnummerierung aller Audiosamples. Da beim verwendeten Codec die Abtastrate 8000 Hertz beträgt, aber ein Wert in Millisekunden benötigt wird, muss der *RTP Timestamp* mit

$$\frac{1000}{8000} = \frac{1}{8}$$

multipliziert werden. Somit ist

$$S_i = \text{RTP Timestamp} \cdot \frac{1}{8}.$$

Auch dieser Wert wurde der Übersichtlichkeit halber um 104 600 000 ms reduziert. Da jeweils nur die Differenz zwischen S_i und S_{i+1}, bzw. R_i und R_{i+1} relevant sind, haben die Anpassungen von S_i und R_i keinen Einfluss auf das Resultat

Der Vollständigkeit halber sind hier nochmals die Gleichungen 3

$$D_{i,j} = \left(R_j - R_i\right) - \left(S_j - S_i\right)$$
$$D_{i,j} = \left(R_j - S_j\right) - (R_i - S_i)$$

und 4 hinterlegt

$$J_i = \frac{J_{i-1} + \left(\left|D_{(i-1),i}\right| - J_{i-1}\right)}{16}.$$

In Tabelle 20 sind die benötigten Schritte dargelegt, wie der *Interarrival Jitter* zwischen verschiedenen Frames berechnet wird.

Frame i	*Frame Arrival Time* R_i	*RTP Timestamp* S_i
0	Mar 20, 2016 14:37:10.645880000 30 645.880 ms	837 055 260 31 907.5 ms
1	Mar 20, 2016 14:37:10.665806000 30 665.806 ms	837 055 420 31 927.5 ms
2	Mar 20, 2016 14:37:10.686285000 30 686.285 ms	837 055 580 31 947.5 ms

Tabelle 19: Wireshark Frame Timesamps

Frame i	$D_{i-1,i}$	J_i
0	—	$J_0 = 0ms$
1	$D_{0,1} = (R_1 - R_0) - (S_1 - S_0)$ $= (30'665.806ms - 30'645.880ms)$ $\quad - (31'927.5ms - 31'907.5ms)$ $= 19.926ms - 20ms$ $= -0.074ms$	$J_1 = J_0 + \frac{\lvert D_{0,1}\rvert - J_0}{16}$ $= 0ms + \frac{\lvert -0.074ms\rvert - 0ms}{16}$ $= 0.004625ms$
2	$D_{1,2} = (R_2 - R_1) - (S_2 - S_1)$ $= (30'686.285ms - 30'665.806ms)$ $\quad - (31'947.5ms - 31'927.5ms)$ $= 20.479ms - 20ms$ $= 0.479ms$	$J_2 = J_1 + \frac{\lvert D_{1,2}\rvert - J_1}{16}$ $= 0.004625ms$ $\quad + \frac{\lvert 0.479ms\rvert - 0.004625ms}{16}$ $= 0.034273ms$

Tabelle 20: Interarrival Jitter *Berechnung*

B Jitsi

B.1 Wichtige Hinweise

Der Einsatz der aktuellen Entwicklerversion von Jitsi 2.9 führte einige Probleme mit sich. Zum zuletzt geprüften Zeitpunkt (14. Juni 2016) existierten noch immer gewisse Probleme mit RTCP, bzw. diese Pakete werden einfach nicht verschickt. Das führt dazu, dass Jitsi gewisse Statistiken nicht korrekt auswerten kann. Namentlich sind dies die Statistiken für die Round Trip Time (RTT), sowie die Jitter Stats. Sofern Jitsi im Zusammenhang mit einem anderen Client eingesetzt wird, der die RTCP Pakete korrekt verschickt, kann zumindest ein Teil dieser Statistiken korrekt berechnet werden. Vollständig werden diese Informationen aber leider erst sein, wenn in Jitsi die Bugs in Zusammenhang mit RTCP gelöst wurden. Es wird deshalb empfohlen, regelmässig die neueste Version von Jitsi zu installieren.

B.2 Inbetriebnahme

Um den mit Echtzeitanalyse-Fähigkeiten erweiterten Jitsi Client zu installieren, ist wie folgt vorzugehen:

1. Download des aktuellsten x64 Clients in Version 2.9 von
   `https://download.jitsi.org/jitsi/nightly/windows/`
   Wenn die Entwicklung des Clients abgeschlossen ist, kann der Client von hier heruntergeladen werden:
   `https://jitsi.org/Main/Download#stableline`
2. Jitsi installieren
3. Das auf der CD mitgelieferte JAR-File `/jitsi_tool/package/swing-ui.jar` identifizieren
4. Im Installationspfad das Verzeichnis `~/Jitsi/sc_bundles/` lokalisieren
5. Das File `swing-ui.jar` in das Verzeichnis `~/Jitsi/sc_bundles/` kopieren, und das bestehende Paket ersetzen
6. Damit ist die Installation abgeschlossen

B.3 Bedienung

Um mit Jitsi eine Echtzeitanalyse durchzuführen, ist wie folgt vorzugehen.

1. Jitsi starten
2. Ganz normal einen Anruf starten
3. Wenn die Verbindung steht, kann das Call Info Window geöffnet werden:

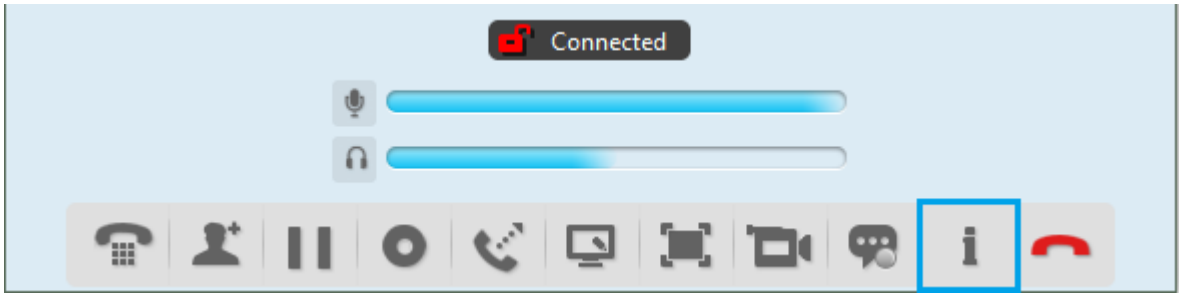


Abbildung 27: Call Info Window öffnen

4. Im in Abbildung 28 dargestellten Fenster können nun auf der rechten Seite die gewünschten Statistiken dargestellt werden

Hinweis: Statistiken werden erst gesammelt, wenn das Call Info Window das erste Mal geöffnet wurde.

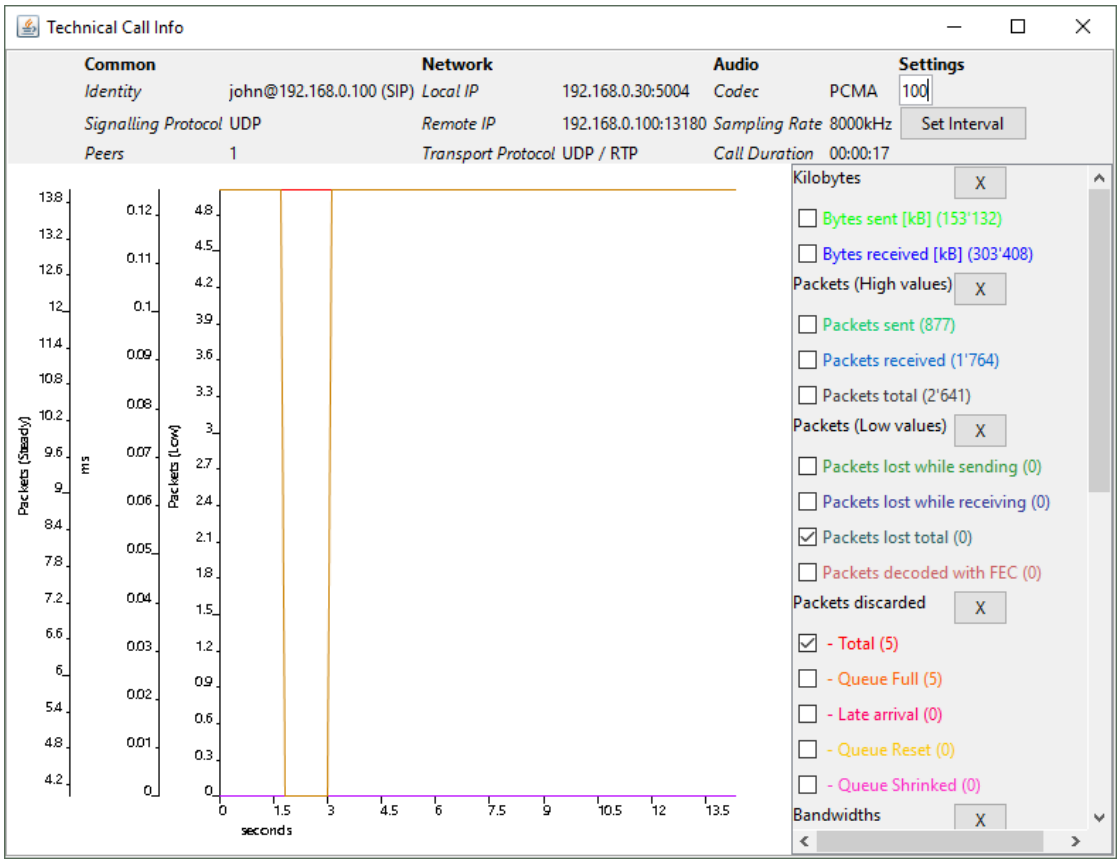


Abbildung 28: Screenshot Jitsi mit Anpassungen

Im oberen Bereich kann mit dem Textfeld im Bereich Settings das Intervall angepasst werden, indem die Statistiken aktualisert werden. Diese Angabe ist in Millisekunden.

Ein Klick auf einen X-Button in der Legende aktiviert/deaktiviert alle sich darin befindlichen Statistiken.

Je nach dargestellten Statistiken werden auf der linken Seite unterschiedliche Achsen dargestellt. Diese sind:

Packets (Steady) Für Statistken, die Pakete anzeigen, die sich über die Zeit nur wenig verändern (z.B. Grösse der Buffer Queue Size)

Packets (Low) Für Statistken, die Pakete anzeigen, die sich nur langsam erhöhen (z.B. verworfene Pakete)

Packets (High) Für Statistiken, die Pakete anzeigen, die sich sehr schnell erhöhen (z.B. gesendete Pakete)

ms Für Statistiken, die zeitliche Informationen anzeigen (z.B. Download Jitter)

kbps Für Statistiken, die Bandbreiteninformationen anzeigen

kB Für Statistiken, welche Kilobytes zählen (z.B. gesendete Bytes)

B.4 Weiterentwicklung

Sollte der es Kompatibilitätsprobleme zwischen dem installierten Jitsi Client sowie dem `swing-ui.jar` Paket geben, oder besteht die Anforderungen, die Statistiken weiter zu entwickeln, muss wie folgt vorgegangen werden.

1. Zu der Website `https://jitsi.org/Documentation/DeveloperDocumentation` navigieren
2. Der Anleitung unter *Getting Started* folgen, um eine IDE einzurichten
   Hinweis: Solange sich Jitsi 2.9 noch in der Entwicklung befindet, kann man einfach den Branch `master` klonen.
   Wenn Jitsi 2.9 fertig entwickelt ist, empfiehlt es sich, den entsprechenden Branch für die Version 2.9 von `https://github.com/jitsi/jitsi/branches/all` zu identifzieren und klonen.
3. Wenn die IDE eingerichtet ist, auf der mitgelieferten CD das Verzeichnis `/jitsi_tool/code/` identifizieren und die gesamte Ordnerstruktur ins geklonte Verzeichnis kopieren. Allfällige bereits vorhandene Files überschreiben.
4. Die Datei `build.xml` im geklonten Verzeichnis suchen und öffnen.
5. In `build.xml` den XML-Tag `<target name="bundle-swing-ui">` identifzieren.

6. Den dort zu findenden XML-Tag `<jar ...>` um die folgenden Zeilen erweitern:
```
<zipfileset src="${lib.noinst}/jchart2d-3.2.2.jar" prefix=""/>
<zipfileset src="${lib.noinst}/jide-oss-2.9.7.jar" prefix=""/>
<zipfileset src="${lib.noinst}/xmlgraphics-commons-1.3.1.jar"
prefix=""/>
```
7. Für die Weiterentwicklung sind vor allem zwei Ant-Targets von Bedeutung:
   - `rebuild`: Das Ant-Target `rebuild` kompiliert das gesamte Projekt und paketiert es entsprechend.
   - `run`: Das Ant-Target `run` basiert auf dem Target `rebuild`. Am Ende des Kompilationsprozesses wird Jitsi gestartet.
8. Nach der Ausführung eines der beiden Ant-Targets findet sich im Verzeichnis `~/sc-bundles` die Datei `swing-ui.jar`. Diese kann nun Analog Appendix B.2 benutzt werden, um einen Jitsi Client mit den Echtzeitstatistiken zu erweitern.

C Akronyme

ADPCM Adaptive Differential Pulse Code Modulation
AES Advanced Encryption Standard
APP Application-defined packet
ARP Address Resolution Protocol
ATA Analog Telephone Adapter

BAKOM Bundesamt für Kommunikation
BYE Goodbye

CS-ACELP Conjugate Structure Algebraic Code-Excited Linear Prediction

DCME Digital Circuit Multiplication Equipment
DH Diffie-Hellman
DHCP Dynamic Host Configuration Protocol

FEC Forward Error Correction
FMJ Freedom for Media in Java

GSM Global System for Mobile communications

HSR Hochschule für Technik Rapperswil

IETF The Internet Engineering Task Force
IP Internet Protocol
ISP Internet Service Provider
ITU International Telecommunication Union
ITU-T ITU Telecommunication Standardization Sector

JAR Java Archive

M2E Mouth-To-Ear
MAC Media Access Control
MELP Mixed-Excitation Linear Predictive
MOS Mean Opinion Score

OP Onion Proxy
OR Onion Router

PBX Private Branch Exchange
pcap Packet Capture
PCM Pulse Code Modulation
PESQ Perceptual evaluation of speech quality

QoS Quality of Service

RC Reception Report Count
RR Receiver Report
RTCP RTP Control Protocol
RTP Realtime Transport Protocol
RTT Round Trip Time

SDES Source Description
SIP Session Initiation Protocol
SR Sender Report
SRF Schweizer Radio und Fernsehen
SRTP Secure Real-Time Transport Protocol
SSRC Synchronization Source

TCP Transmission Control Protocol

UDP User Datagram Protocol

VoIP Voice over IP

WAN Wide Area Network
WHC Wireless Home Connection

ZRTP Zimmermann Real-time Transport Protocol

D Glossar

ITU-T G-Series Eine Familie von Standards für Übertragungssystem und -medien, digitalen Systemen und Netzwerken.[10] In dieser Familie sind unter anderem auch die verwendeten Audiocodecs beschrieben.

Jitter Jitter beschreibt die *Packet Delay Variation* und ist der Unterschied des Einweg-End-zu-End Delays von Paketen in einem Stream, wobei verlorene Pakete ignoriert werden.

Long Liner Spezielle Kunden, wie z.B. Lifttelefone oder Kunden, die nicht direkt mit dem Internet verbunden werden können. Für solche Kunden sind spezielle Verbindungsmassnahmen nötig und werden z.B. über das Mobilfunknetz oder Satellit angebunden.

Mouth-To-Ear Delay Mouth-To-Ear (M2E) Delay ist die Zeit, welche ein Sprachsignal vom Mund des ersten Kommunikationspartnes bis zum Ohr des zweiten Partners benötigt, zeigt also an, wie lange es geht, bis der Kommunikationspartner das Gesagte hört.

Private Branch Exchange Ein System, mit welchem eine private Telefonanlage betrieben werden kann.

SCSIP Steht für *S*wisscom *SIP*, das SIP System der Swisscom.

SIPCALL Ein Third-Party-Anbieter für SIP-Telefonie in der Schweiz.

Wireless Home Connection Wireless Home Connection (WHC) steht für eine Basisstation, die beim Kunden lokal installiert werden kann. Diese verstärtkt das verfügbare, schwache Mobilfunksignal, sodass der Kunde bei seinem Standort eine genügend starke Netzabdeckung hat.

E Literaturverzeichnis

[1] Bundesamt für Kommunikation (BAKOM) *Grundversorgung im Fernmeldebereich*. 05.02.2015,
http://www.bakom.admin.ch/themen/telekom/00457/index.html

[2] *Faktenblatt zur neuen IP Welt bei Swisscom*. Swisscom, 18.03.2014,
https://www.swisscom.ch/content/dam/swisscom/de/about/medien/faktencheck/documents/20140318_Faktencheck_All_IP-de.pdf

[3] Dani Müller und Maria Kressbach, *Ende des Analog-Netzes: Was Swisscom-Kunden wissen müssen*. Schweizer Radio und Fernsehen (SRF), 01.03.2016,
http://www.srf.ch/konsum/themen/multimedia/ende-des-analog-netzes-was-swisscom-kunden-wissen-muessen

[4] Manoj Bhatia, Jonathan Davidson, Satish Kalidindi, Sudipto Mukherjee, James Peters, *Voice over IP Fundamentals, 2nd Edition*. Cisco Press, 27.07.2006,
http://www.ciscopress.com/store/voice-over-ip-fundamentals-9781587052576

[5] Claude E. Shannon, *Communication in the Presence of Noise*. Proc. Institute of Radio Engineers, vol. 37, no. 1, 1949, pp. 10-21,
http://web.stanford.edu/class/ee104/shannonpaper.pdf

[6] ITU Telecommunication Standardization Sector, *Methods for subjective determination of transmission quality*. ITU-T Recommendation P.800,
http://www.itu.int/rec/T-REC-P.800-199608-I

[7] ITU Telecommunication Standardization Sector, *Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs*. ITU-T Recommendation P.862,
http://www.itu.int/rec/T-REC-P.862-200102-I

[8] *Understanding Delay in Packet Voice Networks*. Cisco, 02.02.2006
http://www.cisco.com/c/en/us/support/docs/voice/voice-quality/5125-delay-details.html#packetizationdelay

[9] Swisscom-Lernende, *M2E-Delay Tests*. 26.02.2016,
Mouth-to-Ear-Delay_new.xlsx

[10] ITU Telecommunication Standardization Sector *Transmission systems and media, digital systems and networks*. ITU-T
https://www.itu.int/rec/T-REC-G/en

[11] International Telecommunication Union (ITU) *Recommendation G.114*. ITU-T, 05.2003,
https://www.itu.int/rec/T-REC-G.114-200305-I/en

[12] International Telecommunication Union (ITU) *Recommendation G.729*. ITU-T, 06.2012,
https://www.itu.int/rec/T-REC-G.729-201206-I/en

[13] The Internet Engineering Task Force (IETF) *The Secure Real-time Transport Protocol (SRTP)*. RFC , 04.2004,
https://tools.ietf.org/html/rfc3711

[14] The Internet Engineering Task Force (IETF) *RTP: A Transport Protocol for Real-Time Applications*. RFC , 07.2003,
https://tools.ietf.org/html/rfc3550

[15] Roger Dingledine, Nick Mathewson *Tor Protocol Specification*. 04.2016,
https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt

[16] Digium Inc, *Asterisk*. 04.2016,
http://www.asterisk.org/

[17] truecrypt.ch, *Inoffizielles Download-Archiv*. 04.2016,
https://truecrypt.ch/

[18] Wireshark Wiki, *RTP statistics*. , 03.2016,
https://wiki.wireshark.org/RTP_statistics

[19] Philip R. Zimmermann, *PGPfone - Pretty Good Privacy Phone*. 04.2016,
http://www.pgpi.org/products/pgpfone/

[20] Van Gegel, *TOR Fone - p2p secure and anonymous VoIP tool*. 04.2016,
http://torfone.org/index.html

[21] Van Gegel, *TORFone User's Guide* . 05.2016,
http://torfone.org/help.html

[22] Roger Dingledine, Nick Mathewson, Paul Syverson *Tor: The Second-Generation Onion Router*. Directory Servers, Chap. 6.3, 04.2016,
http://www.onion-router.net/Publications/tor-design.pdf

[23] Roger Dingledine, Nick Mathewson, Paul Syverson *Tor: The Second-Generation Onion Router*. Constructing a circuit, Chap. 4.2, 04.2016,
http://www.onion-router.net/Publications/tor-design.pdf

[24] Victor Demjanenko, David Satterlee *RTP Payload Format for MELPe Codec*. , 10.2014,
http://www.vocal.com/wp-content/uploads/2012/05/ietf-rtp-payload-for-melpe.pdf

[25] Zoiper.com, *Zoiper*. 04.2016,
http://www.zoiper.com/

[26] Jitsi, *Jitsi*. 04.2016,
https://jitsi.org/

[27] Korolev Alexandr, *Example scenario of establish a connection with the participation of SIP Redirect Server and SIP Proxy*. 07.04.2015
https://commons.wikimedia.org/wiki/File:SIP_call_flow_between_UA,_Redirect_Server,_Proxy_and_UA.png

[28] Lumisoft.NET, *.NET Open Source Stack für SIP & RTP*. 06.04.2016
http://www.lumisoft.ee/lswww/download/downloads/Net/

[29] Lumisoft.NET UA Example, *Beispielimplementation eines SIP-Clients mit UI basierend auf Lumisoft.NET*. 06.04.2016
http://www.lumisoft.ee/lswww/download/downloads/examples/SIP_UA.zip

[30] Tool für PC Stresstests, *Stress test for your PC. Let him sweat*. 25.05.2016
http://www.softwareok.com/?Download=StressMyPC

[31] Beschreibung von a-Law und *mu*-Law, *Waveform Coding Techniques - A-law and u-law Companding*. 28.04.2016
http://www.cisco.com/c/en/us/support/docs/voice/h323/8123-waveform-coding.html#t7

F Projektmanagement

Für die Projektplanung wurde auf kein typisches Vorgehensmodell, wie es aus der Softwareentwicklung bekannt ist, zurückgegriffen. Zu Beginn wurde ein Grobplan erstellt und laufend an die neuen Gegebenheiten angepasst. Dabei wurden neue Beschlüsse und Vertiefungsarbeiten mit dem Betreuer abgesprochen.

F.1 Projektplan

Der Projektplan hält die Ressourcen und groben Aktivitäten der Projektmitarbeiter fest. Dabei handelt es sich um eine IST-Aufzeichnung.

	Datum	22.02.	29.02.	07.03.	14.03.	21.03.	28.03.	04.04.	11.04.	18.04.	25.04.	02.05.	09.05.	16.05.	23.05.	30.05.	06.06.	13.06.	
	Woche	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	**Total**
Besprechung, Planung, Sitzung, Administrative Arbeit	pmeier	3		6	3	5	5	4	7	1	3	2	3	3	4	4	8	11	**72**
	mobrist	2		3	4	4	2	2	3	1	3		3	3	3	4	2	9	**48**
Analyse, Verfeinerung Aufgabenstellung, Arbeitsumgebung einrichten	pmeier	3				1			3	4	5		1	2			1	2	**22**
	mobrist	6	6																**12**
Einarbeitung, Vertiefung der Theorie	pmeier	6	10	6		3			5	7	2	5	3	1					**48**
	mobrist		8	9		4													**21**
Technologiestudie: Installation, Inbetriebnahme VoIP-Labor	pmeier		5	2	4	11	2	2	2	1	1								**30**
	mobrist			2	5	5													**12**
Testen: Delay, Qualitätsmessungen	pmeier			7	12		14	7		1	3	8	6	7	6	3			**74**
	mobrist			2															**2**
Entwicklung SIP Test Client	pmeier								2						3				**5**
	mobrist						4	20	16	15	23	11	25	17	17	5			**153**
Dokumentation	pmeier		5	1	2		9	11	1	9	6	2	8	7	8	17	19	10	**115**
	mobrist	2	4	7	9	13	10	2				2			2	13	40	11	**115**
	Total pmeier	12	20	22	21	20	30	24	20	23	20	17	21	20	21	24	28	23	**366**
	Total mobrist	10	18	23	18	26	16	24	19	16	26	13	28	20	22	22	42	20	**363**
	Total	**22**	**38**	**45**	**39**	**46**	**46**	**48**	**39**	**39**	**46**	**30**	**49**	**40**	**43**	**46**	**70**	**43**	**729**

Abbildung 29: Projektplan

F.2 Projektorganisation

F.F.2.1 Teammitglieder

Das Projektteam besteht aus zwei gleichberechtigten Personen:

- Pascal Meier
- Max Obrist

F.F.2.2 Betreuer

Die Betreuung dieser Bachelorarbeit wird übernommen von:

- Prof. Dr. sc. techn. Peter Heinzmann

F.3 Zeitaufwand

Mit dem Startstuss am 22.02.2016 stehen dem Kandidaten 17 Wochen zur Verfügung. Dabei soll der Aufwand von mind. 360 Stunden nicht unterschritten werden. Die Arbeit findet parallel zum Unterricht der HSR statt und kann frei eingeteilt werden.

F.4 Zeitauswertung

Abbildung 30 gibt einen Überblick über die aufgewendeten Arbeitsstunden pro Person und Semesterwoche. Geplant wurde mit einem Projektaufwand von 360 Stunden. Die Ist-Zeit übersteigt das Zeitbudget nur geringfügig, die Vorgabe konnte somit eingehalten werden.

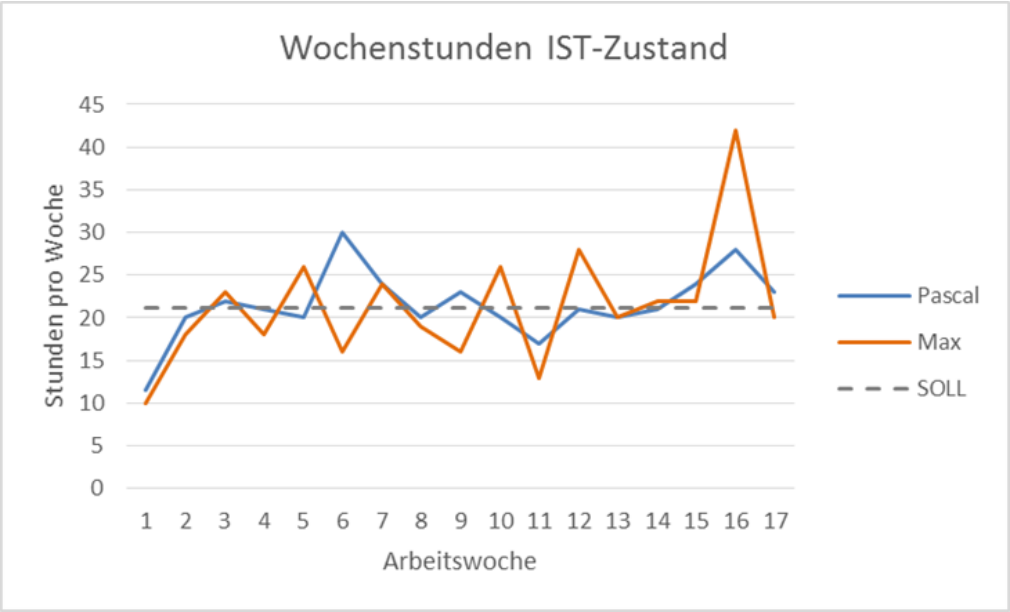


Abbildung 30: Zeitauswertung

F.5 Besprechungen

Es wird eine wöchentliche Sitzung mit dem Betreuer angestrebt, um den aktuellen Stand zu besprechen. Dabei kann der genaue Wochentag variieren. Alle wichtigen Ereignisse und Entscheidungen werden in einem Sitzungsprotokoll festgehalten.

F.6 Infrastruktur

F.F.6.1 Hardware

- Intel Xeon CPU @ 3.40GHz, 16GB RAM
- Apposite Linktropy 5500 WAN Emulator
- HP 1810-8G - Switch
- Microsoft Lumia 950 - Aufnahmegerät

F.F.6.2 Software

- Allgemein
  - Windows 7, SP1 64bit
  - Windows 10, 64bit
- VoIP Infrastruktur
  - Ubuntu, 15.10
  - Asterisk, 11.21.2
  - Jitsi, 2.9.5518
  - X-Lite, 4.9.3 79961
  - Zoiper, 3.9.32133 32bit
  - TorChat, 0.9.9.553
  - TORFone, 1.1b
  - GoldWave, 6.20
  - Wireshark, 2.0.3
- Dokumentation
  - Microsoft Office 2013
  - MiKTeK 2.9
  - LaTeX
- Entwicklung Lumisoft
  - Microsoft Visual Studio 2015 Professional
  - .NET / C#
- Entwicklung Jitsi
  - IntelliJ IDEA 2016
  - Java 8

F.7 Risikomanagement

Abbildung 31 zeigt eine Analyse der während der Arbeit erwarteten Risiken. Diese werden in die Eintrittswahrscheinlichkeiten niedrig, mittel und hoch klassifiziert. Anhand einer Auswertung der möglichen Auswirkungen sowie der Eintrittswahrscheinlichkeit empfiehlt sich eine frühzeitige Behandlung der Risiken.

ID	Risiko	Auswirkung	Massnahme	Eintrittswahrscheinlichkeit
R01	Ausfall durch Krankheit	Verzögerung des Projektes	-	Niedrig
R02	Projektumfang lässt sich nicht genau abschätzen	Planung des Projektes erschwert. Zeitplanung kann nicht vollständig gemacht werden.	Regelmässiges Feedback einholen. Angepasste Aufgabenstellung anfordern	Hoch
R03	Datenverlust	Teile der Arbeit gehen verloren (Code, Messresultate, Code, ect.)	Einsatz eines Versionsverwaltungssystem, zusätzliche Sicherungen auf andere virtuelle Server	Niedrig
R04	Hardware/Software kann nicht wie geplant beschaffen werden	Zeitliche Verzögerung. Unter Umständen muss eine Alternative beschaffen werden.	Abklärungen und Zugeständnisse rechtzeitig einholen	Niedrig - Mittel
R05	Eigner SIP-Client kann nicht in angemessener Zeit entwickelt werden, um den Jitter-Buffer manuell zu setzen	Delays können nicht vollständig lokalisiert werden und mindern Testresultate.	Verschiedene SDKs evaluieren und ausprobieren	Mittel - Hoch
R06	Tool zur Qualitätsmessung kann nicht vollständig umgesetzt werden.	Minderung des Gesamtresultats	Frühzeitig die genauen Anforderungen aufnehmen und genügend Zeit einplanen	Mittel
R07	Benutzeroberfläche des Tools entspricht nicht den Anforderungen des Endbenutzers	Endbenutzer finden sich nicht vollständig zurecht	Besprechung der Designideen bevor Oberfläche entwickelt wird	Mittel

Abbildung 31: Risiken

F.8 Meilensteine

MS1 (09.03.2016) – Aufbau Testlabor Demonstration des Testaufbaus

MS2 (30.03.2016) – Einarbeitung in die Thematik Theorie / Einführung abgeschlossen

MS3 (11.05.2016) – Auswertung eigener Client Lumisoft.NET Client fertig entwickelt und Untersuchungen abgeschlossen

MS4 (20.05.2016) – Erweiterung Jitsi abgeschlossen Bestehender Client wurde weiterentwickelt um erweiterte Echtzeitstatistiken anzuzeigen

MS5 (17.06.2016) – Abgabe Bachelor-Arbeit Vollständige Dokumentation inklusive Plakat und abzugebende Elemente

MS6 (23.08.2016) – Präsentation Bachelor-Arbeit Präsentation der Arbeit mit anschliessender Fragerunde

G Persönliche Berichte

G.1 Max Obrist

Durch meine tägliche Arbeit als Software Engineer in der CREALOGIX AG sammle ich bereits jetzt viel praktische Erfahrung in der Software Entwicklung. Wie schon bei meiner Studienarbeit wollte ich auch bei meiner Bachelorarbeit deshalb ein Thema bearbeiten, bei welchem mir die praktische Erfahrung eher etwas fehlt. Wenn möglich sollte die Arbeit aber trotzdem einige Möglichkeiten zur Softwareentwicklung bieten. Mein Partner in der Bachelorarbeit, Pascal Meier, kam gleichzeitig aus einem Bereich, in dem er vor allem mit diversen Netzwerktechnologien in Berührung kam. Damit empfahl sich ein Thema, welches diese beiden Bereiche miteinander kombiniert. Die Arbeit von Prof. Dr. sc. techn. Peter Heinzmannempfahl sich deshalb besonders für uns, auch wenn ich persönlich mit dem Thema VoIP bisher nur sehr wenige Berührungspunkte hatte.

In den ersten Wochen ging es deshalb vor allem um die Einarbeitung in dieses Themenfeld und das Lesen diverser theoretischer Quellen. Die Thematik stellte sich dabei in diesem ersten Schritt als interessanter als erwartet dar, hauptsächlich, weil sie um einiges breiter als angenommen war.

Nach diesen initialen theoretischen Analysen ging es darum, einen Versuchsaufbau zu erstellen, mit dem wir während dem Rest der Arbeit möglichst vielfältige Tests durchführen konnten. Schnell stellte sich heraus, dass unsere ursprünglichen Annahmen für die Hauptquellen der Verzögerungen ziemlich an der falschen Stelle ansetzten und die Hauptgründe für die Verzögerungen in den Clients, und nicht im Netzwerk lag.

Dadurch viel der Entschluss, einen eigenen kleinen SIP-Client zu schreiben – auf Basis von Lumisoft – um zu analysieren, was denn eigentlich innerhalb eines Clients alles geschieht. Ab diesem Zeitpunkt wurde die Arbeit für mich besonders interessant, da ich die gesamte Problematik rund um SIP nun einmal aus der Perspektive eines Softwareentwicklers ansehen konnte. Es stellte sich heraus, dass bei der Implementation des SIP-Client viele Probleme auftauchen können – Eigenheiten der Protokolle, unerwartetes Verhalten der Clients und in meinem Fall vor allem der leider nicht ganz optimale Lumisoft Software Stack.

Ab diesem Zeitpunkt hat sich der Fokus von Pascal Meier und mir etwas getrennt. Während er sich weiterhin um das Messen und Erklären des Einflusses verschiedener Parameter konzentrierte, lag mein Fokus nun weithin im Entwickeln von Software. Als es darum ging, mithilfe eines selbst entwickelten Clients detaillierte Statistiken darzustellen, merkte ich aber, dass die Basis mit Lumisoft zu instabil ist, und ich musste mich entscheiden, die weiteren Entwicklungen auf Basis von Jitsi zu erstellen. Damit war der eigentliche VoIP Teil der Entwicklung abgeschlossen, es ging nun hauptsächlich darum, bestehende Teile von Jitsi zu erweitern, um die

gesammelten Statistiken schön darzustellen. Das finale Produkt wird zwar bestimmt keine Schönheitspreise gewinnen, aber ich denke, dass es einem doch helfen wird, VoIP Verbindungen zu analysieren und allfällige Probleme zu beheben.

Ich muss allerdings zugeben, dass mich die Thematik, trotz einiger sehr interessanten Aspekten, nie richtig gepackt hat. Ich denke, das liegt daran, dass ich schlicht und einfach ein Entwickler bin. Da gehört es natürlich manchmal auch dazu, sich um die kleinen Details diverser Protokolle zu kümmern, allerdings nicht auf einem Level, wie es diese Arbeit erfordert hat. Das kann zwar kurzzeitig interessant sein, aber für mich fehlte ein wenig die Kreativität, die man der Software Entwicklung doch immer wieder benötigt. Deshalb empfand ich die Arbeit teilweise als etwas harzig. Ich bin wohl einfach ein Vollblut-Entwickler, wie ich das manchmal ausdrücke.

Die Zusammenarbeit mit Pascal Meier hat dafür immer sehr gut funktioniert. Es war wohl auch ein Vorteil, dass wir uns um verschiedene Bereiche der Arbeit gekümmert haben, wodurch man ein wenig aneinander vorbeiarbeiten konnte und meist unterschiedliche Bereiche bearbeitete, was die Zusammenarbeit doch erheblich vereinfacht. Ich bin aber überzeugt, dass die Zusammenarbeit auch bei einem Projekt mit mehr Reibungsfläche wunderbar funktioniert hätte.

Auch die Betreuung durch Prof. Dr. sc. techn. Peter Heinzmannwar meistens sehr ergiebig. Er gab uns immer wieder Inputs, auch was für Bereiche man sich noch zusätzlich fokussieren könnte oder eine detailliertere Betrachtung benötigten. Ich persönliche hätte mir aber Teils etwas mehr Feedback gewünscht. Ich hatte häufig das Gefühl, nicht genau zu wissen, wie wir eigentlich mit unserer Arbeit stehen, und ob sie dem erwarteten Niveau entsprach. Wenn es aber wichtig war, erhielten wir immer die entsprechende Unterstützung.

G.2 Pascal Meier

Für mich stand von Anfang an fest, dass ich eine Arbeit schreiben wollte, bei der die Softwareentwicklung nicht im Vordergrund stand. Beruflich war ich viele Jahre im Telekommunikationsbereich als Netzwerkingenieur tätig und wollte diesem Berufszweig treu bleiben. Die Thematik Voice over IP interessierte mich von Anfang und ich konnte mich dafür begeistern.

Nachdem ich bereits meine Studienarbeit im Alleingang absolvierte hatte, wollte ich dies bei meiner Bachelorarbeit ändern. Mit Max Obrist habe ich einen idealen Arbeitspartner gefunden, der mich bestens ergänzt. Da sein Schwerpunkt in der Softwareentwicklung liegt, konnten wir unsere Stärken ergänzen.

Die Arbeit selbst verlief ein wenig anders als erwartet. Nachdem in den ersten paar Wochen der Industriepartner noch ziemlich stark involviert war, zeichnete sich später ein anderes Bild ab. Da sich der Fokus der Arbeit fortlaufend veränderte,

erschwerte sich die Projektplanung erheblich. Der Schwerpunkt wurde auf die genaue Analyse des M2E-Delay gelegt. Das lag hauptsächlich daran, dass sich die gewonnenen Messresultate zwischen den verschiedenen VoIP-Clients sehr stark Unterschieden haben. Das wiederum führte dazu, dass wir der Ursache auf den Grund gehen mussten. Es stellte sich heraus, dass die Theorie nicht mit der Praxis übereinstimmte. Es gab innerhalb des Clients zu viele unbekannte Komponenten, die eine präzise Schlussfolgerung verunmöglichte. Daher haben wir den Entschluss gefasst, einige Teile der Clients selbst zu erstellen bzw. weiterzuentwickeln. Erst dadurch konnten einzelne Verhaltensmuster weitgehend geklärt werden.

Rückblickend schaue ich auf 17 spannende Wochen zurück, in denen ich vieles gelernt habe. Es hat sich gezeigt, dass man stets flexibel und auf neue Gegebenheiten gefasst sein muss. Nur so ist es möglich, ein optimales Ergebnis zu erreichen.

H Inhaltsverzeichnis CD

CD	
+ doc	
| - admin	Enthält administrative Dokumentationen
| - diagrams	Diagramme und Grundlagen für deren Erzeugung
| + documentation	Enthält den LaTeXSourcecode für diese Dokumentation
| | - images	Enthält die Bilder der Dokumentation
| - protocol	Sitzungsprotokolle aller Sitzungen
| - stats_measurements	Rohdaten der Messungen und ausgewertete Statistiken
+ jitsi_tool	Enthält alle für Jitsi relevanten Dateien
| - code	Enthält den Sourcecode der Anpassungen von Jitsi
| - package	Enthält das Package `swing-ui.jar`
+ SIPTestClient	Enthält den Sourcecode des Lumisoft Clients

I Poster

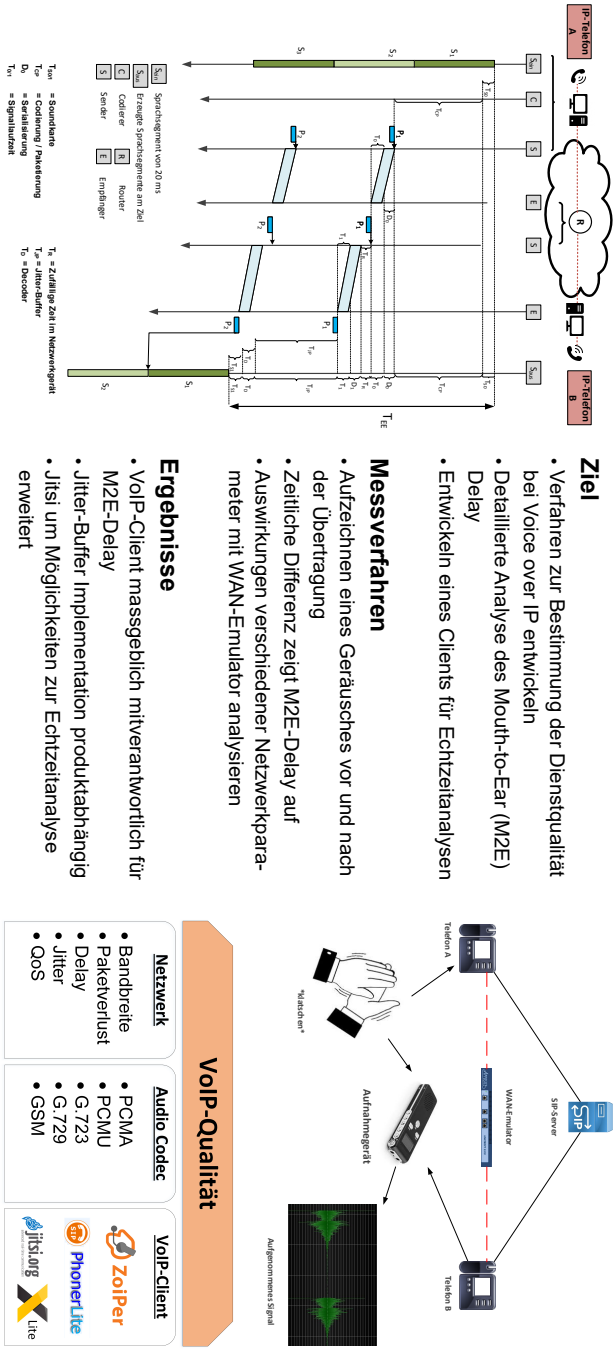


Abbildung 32: BA Poster

J Eigenständigkeitserklärung

Rapperswil, 17.06.2016

Titel der Arbeit:	**VoIP-Qualitätsparameter – Mouth-to-Ear Delay**
Team:	**Max Obrist, Pascal Meier**
Betreuer:	**Prof. Dr. sc. techn. Peter Heinzmann**
Co-Referent:	**Prof. Oliver Augenstein**
Experte:	**Dr. Th. Siegenthaler, CSI Consulting AG**
Industriepartner:	**Swisscom**

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habne, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben,
- dass wir keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt haben.

Max Obrist
Student

_____

Pascal Meier
Student

_____

K Vereinbarung zur Verwendung und Weiterentwicklung der Arbeit

Titel der Arbeit:	**VoIP-Qualitätsparameter – Mouth-to-Ear Delay**
Team:	**Max Obrist, Pascal Meier**
Betreuer:	**Prof. Dr. sc. techn. Peter Heinzmann**
Co-Referent:	**Prof. Oliver Augenstein**
Experte:	**Dr. Th. Siegenthaler, CSI Consulting AG**
Industriepartner:	**Swisscom**

Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der *Bachelorarbeit „VoIP-Qualitätsparameter – Mouth-to-Ear Delay*“ von *Max Obrist* und *Pascal Meier* unter der Betreuung von Prof. Dr. P. Heinzmann (für die Arbeit verantwortlicher Professor) geregelt.

Urheberrecht

Die Urheberrechte stehen den Studenten Max Obrist und Pascal Meier zu.

Verwendung

Die Ergebnisse der Arbeit dürfen sowohl von allen an der Arbeit beteiligten Parteien, d.h. von den Studenten, welche die Arbeit verfasst haben, vom verantwortlichen Professor sowie vom Industriepartner verwendet und weiterentwickelt werden. Die Namensnennung der beteiligten Parteien ist bei der Weiterverwendung erwünscht, aber nicht Pflicht.

Max Obrist *Student*	Ort, Datum	Ort, Datum	Pascal Meier *Student*
_____	_____	_____	_____
Peter Heinzmann *Verantwortlicher Professor*	Ort, Datum	Ort, Datum	Industriepartner
_____	_____	_____	_____

L Sitzungsprotokolle

In diesem Abschnitt sind alle während der Arbeit erstellten Sitzungsprotokolle hinterlegt.

L.1 Sitzung 25. Februar 2016

Sitzungsprotokoll

Projekt:	**Peer-to-Peer Anwendungen und VoIP**
Woche:	**KW 08**
Ort:	**CNLab, Obere Bahnhofstrasse 32B, 8640 Rapperswil**
Datum / Zeit:	**25.02.2016 16.00 – 17.15**

Sitzungsteilnehmer / Kürzel

Pascal Meier	pmeier@hsr.ch
Max Obrist	mobrist@hsr.ch
Peter Heinzmann	peter.heinzmann@cnlab.ch
Patrick Eichler	peichler@hsr.ch
Reto Hauri	reto.hauri@swisscom.com

Traktanden:

- Einführung Problemstellung (Swisscom)
- Organisatorisches
  - Termine
  - Zeiterfassung
  - Protokolle
  - Projektplan

Beschlüsse (Diskussion):

- Swisscom: Umstellung auf „All IP“ https://www.swisscom.ch/en/residential/internet/offers/grundversorgung.html
  - Leute, die nur ein Analogtelefon haben, erhalten einen Router mit SIP Client und können so ihr Telefon über den ATA (Analog Telephone Adapter für die Wandlung zu Analog) anschliessen
  - Es gibt Telefone, mit Anforderungen an die Stromversorgung über die Analogleitung, welche vom ATA-Adapter nicht erfüllt werden können (z.B. Lifttelefone, Alarmanlagen, Telefonzentralen). Vgl. Kassensturz nächste Woche?
  - 7‘000 bis 10'000 Kunden können nicht mit der geforderten minimalen Datenrate angeschlossen werden (Longliner)
    - Input hei: Im Rahmen dieser Arbeit soll aufgezeigt werden, was die minimal erforderliche Datenrate für gute VoIP-Qualität. Swisscom könnte theoretisch auch spezielle Modem-Konfigurationen mit 600kb/s symmetrisch einführen, was die Anzahl der Longliner reduzieren würde.
    - von den Longlinern sollen 2/3 via Wireless Home Connection (WHC) Router angeschlossen werden
    - die restlichen 1/3 sollen soll via Satellit angeschlossen werden. Die verwendeten geostationären Satelliten sind aber auf 36‘000km Höhe, was Delays von 240ms für einen Weg zu Folge hat, Signallaufzeit vom Sender zum Empfänger 2*36‘000km/300km/ms = 240ms); die resultierenden rund 650ms Round-Trip-Delay (RTT) beeinträchtigen die Kommunikationsqualität
- Swisscom und cnlab führten Mouth To Ear (M2E) Delay Tests für verschiedene Voice-Verbindungen durch.

  - Nur schon aufgrund von verschiedenen Digital-Analog-Wandlungen und wegen der Kompression kommen Signalverzögerungen von 100ms bis 300ms zustande (siehe Dropbox)
  - Man beobachtet grosse Unterschiede bei den Mouth-to-Ear-Delays zwischen SCSIP (Swisscom SIP) und SIPCALL (Schweizer Third Party Anbieter). In der ersten Phase der Arbeit geht es darum, zu verstehen, wie die Verzögerungen bei den verschiedenen Systemen zustande kommen (Digitalisierung, Kompression, Netzwerklaufzeit etc.). Basierend auf diesem Wissen sollten dann auch Unterschiede erklärt werden können.
- Analyse der verschiedenen Schritte (und Algorithmen) von Telefon zu Telefon -> Finden der technisch bedingten Latenzen.
  - Siehe Dropbox -> Referenzen -> VoIP Kurs
  - Allenfalls weitere Informationen bei: SIPCALL https://www.sipcall.ch/sipcall-de/ueber-uns/unsere-adresse.php
  - Oder bei einer der Firmen von Prof. Stettler (https://www.e-fon.ch)

Entscheide:

Weiteres Vorgehen während den ersten zwei Arbeitswochen

1) Theorie einarbeiten (parallel zur Erstellung des Versuchsaufbaus)
2) Einfaches System aufbauen für Messungen von M2E Delay
   a) Im ersten Schritt wollen wir möglichst alle Komponenten selbst unter Kontrolle haben (z.B. mit eigner Asterisk Lösung http://www.asterisk.org)
      i) Versuchsaufbau: Asterisk () auf Studienrechner. SIPClients auf Laptops. M2E Delay zwischen Laptops Messen
      ii) Swisscom WHC-Lösung in Betrieb nehmen (Material bei Swisscom abholen) und dazu alle Komponenten und Delayeinflüsse erklären
3) Stand der Arbeit bzw. des Messaufbaus anhand einiger Powerpoint-Slides beim nächsten Meeting präsentieren
4) Groben Zeitplan erstellen mit Meilensteinen

- Allgemeines Vorgehen für Arbeit: Zuerst Delay messen und analysieren, dann eigentliche Sprachqualität.

Pendenzen:

Was	Verantwortlichkeit
Sofern vorhanden: Weitere Unterlagen zu den Messungen	Reto Hauri
Besuch bei Swisscom und Material abholen	Studenten / Reto Hauri
Einarbeitung Theorie	Studenten
Verschicken Kontaktdaten	Studenten

Materialliste:

Datum Abgabe	Gegenstand (an Studenten abgegebenes Material, Bücher, etc.)	Datum Rückgabe
25.02.2016	Apposite, Serie Nr. ??? HSR Inventarnummer: ??? mit Power- und Serial-Kabel	

L.2 Sitzung 09. März 2016

Sitzungsprotokoll

Projekt:	**Peer-to-Peer Anwendungen und VoIP**
Woche:	**KW 10**
Ort:	**HSR, Raum 1.262**
Datum / Zeit:	**09.03.2016 10.30 – 12.00**

Sitzungsteilnehmer

Pascal Meier	pmeier@hsr.ch
Max Obrist	mobrist@hsr.ch
Peter Heinzmann	peter.heinzmann@cnlab.ch

Traktanden:

- Analyse Versuchsaufbau
- Grobplanung
- Weiteres Vorgehen

Beschlüsse (Diskussion):

- Grosse Delays trotz sehr einfachem Versuchsaufbau (Asterisk PBX mit 2 Softphones am selben Switch → 150ms bis 175ms)
- Entstehung von Delays kurz angeschaut mit Peter Heinzmann
  - Je nach Codierer werden beispielsweise 160 Bytes (das entsprich 160 Abtastwerten) in einem RTP-Paket übertragen. Das Sammeln dieser 160-Abtastwerte benötigt 20ms. Wo genau die restlichen 130ms zustande kommen ist noch nicht klar. Es könnte sein, dass die Empfangspuffer mehrere Datenpakete sammeln, um allfälligen Jitter-Problemen entgegen zu wirken.
- Weitere Versuche: WHC, Satellit, Softphones, Analog Telefon, Mobilfunk…

Entscheide:

Weiteres Vorgehen

- Analyse G.711 (PCM) sowie GSM
  - Was sind technisch bedingte Delays?
  - Woher kommen Unterschiede?
- Material für weitere Versuche bei Swisscom abholen

Pendenzen:

Was	Verantwortlichkeit
Technisch bedingte Delays G.711 analysieren	Studenten
Projektplan erstellen	Studenten
Material für Versuchsaufbau organisieren	Studenten/Reto Hauri

Nächster Termin:

Datum:	Mi, 17.03.2016
Zeit:	11.00
Ort:	HSR, 1.262
Dauer:	1 Stunde

L.3 Sitzung 10. März 2016

Sitzungsprotokoll

Projekt:	**Peer-to-Peer Anwendungen und VoIP**
Woche:	**KW 10**
Ort:	**Swisscom, Hardturmstrasse 3, 8005 Zürich**
Datum / Zeit:	**10.03.2016 14:00 – 14:30**

Sitzungsteilnehmer

Pascal Meier	pmeier@hsr.ch
Max Obrist	mobrist@hsr.ch
Reto Hauri	reto.hauri@swisscom.com

Traktanden

- Vorstellung Versuchsaufbau M2E Messungen bei Swisscom
- Swisscom Infrastruktur
- Material für weitere Versuche mitnehmen

Beschlüsse (Diskussion)

- Versuchsaufbau sehr ähnlich wie bei uns (Tonaufnahme mit externem Gerät, Audiofile mit Audacity analysiert, Delays extrahiert)
- Zusätzliche getestete Verbindungen (WHC, Satellit, SCSIP, SIPCALL, Analog Telefon over ATA, SIP-Phone…)
- Leitungsspeed am Satellit
    - Ungedrosselt: 3Mbit/s Downstream /300kbit/s Upstream
    - Gedrosselt (nach 16GB/Monat): 400kbit/s/200kbit/s
- Codec: G.711. Weitere Informationen zu Leitung/Codecs und Infrastruktur im Dropbox unter Swisscom Tests

Entscheide

Weiteres Vorgehen

- WHC Tests durchführen
- Material von Swisscom erhalten (SIP-Phone, Analog Telefon)
- Weitere Hardware kommt später

Pendenzen:

Was	Verantwortlichkeit
Technisch bedingte Delays G.711 analysieren	Studenten
Projektplan erstellen	Studenten
Material für Versuchsaufbau organisieren	Studenten/Reto Hauri

L.4 Sitzung 16. März 2016

Sitzungsprotokoll

Projekt:	**Peer-to-Peer Anwendungen und VoIP**
Woche:	**KW 12**
Ort:	**cnlab, Sitzungszimmer**
Datum / Zeit:	**16.03.2016 11:00 – 12:20**

Sitzungsteilnehmer

Pascal Meier	pmeier@hsr.ch
Max Obrist	mobrist@hsr.ch
Patrick Eichler	peichler@hsr.ch
Peter Heinzmann	peter.heinzmann@cnlab.ch

Traktanden

- Aktueller Stand besprechen (unerklärliche Delays)
- Delay-Analyse
- Materialliste besprechen
    - Satellit?

Diskussion

- Analyse der bisherigen Delays
    - Die mit verschiedenen Clients gemessenen Delays (Client A = ???, Client B = ???) lassen vermuten, dass die Delays von den Clients selbst kommen. Peter sagt, dass die Empfangspuffer wahrscheinlich mehr Delay verursachen, als wir das bisher angenommen haben. Es sollen nun Messungen mit unterschiedlichen Clients durchgeführt und die Ergebnisse tabellarisch zusammengefasst werden:
      Sender – Empfänger:
      Client A – Client A
      Client B – Client B
      Client A – Client B
      Client B – Cleint A
    - Delays Messung zwischen Mikrofon Aufnahme und Serialisierung auf Netzwerk Interface
- Messsetup
    - Zwei Switches am Apposite, Clients je am Switch
    - Weitere Messungen zwischen Hardphone, Zoiper, Jitsi (alle mit allem)
    - Mit Apposite diverse weitere Einstellungen messen (Linkspeed, Jitter, Latency, Packet Drops)
- Coreferent: entweder Farhad Metha oder Oliver Augenstein (allenfalls auch Stettler)
- Satellit: Messung möglich bei cnlab

Entscheide

Weiteres Vorgehen

- GSM: Delay Analyse zwischen Jitsi und Zoiper
- Bei IT Desk 2 Cisco Phones abholen → Delays messen
    - IT Desk: Keine
    - Cisco Phones an HSR verwenden proprietäres Cisco Protokoll, Firmware Upgrade nötig…
- Dokumentation: Theorie und Messungen kombinieren
- Ein weiteren Monitoring-Switch

Zu bestellendes Material: 1 WHC Router, 1 Analog Telefon

Pendenzen:

Was	Verantwortlichkeit
Schindelleggi SIPCALL anfragen	Peter Heinzmann
2 SIP Phones von HSR IT Diensten	Studenten
SIP/RTP Portfreigabe durch HSR IT Dienste	Studenten
Oszilloskop Ethernet Kasten abholen	Studenten/Patrick Eichlers
Material bei Swisscom bestellen	Studenten

Materialliste:

Datum Abgabe	Gegenstand (an Studenten abgegebenes Material, Bücher, etc.)	Datum Rückgabe
25.02.2016	Apposite, Serie Nr. LR201311009378 HSR Inventarnummer: 118811 mit Power- und Serial-Kabel	
04.03.2016	4x LAN Kabel gründ, HP Switch 8 Port	
04.03.2016	DSO + Netzteil + USB-Verbindung, Audiomessbox, 2 Kopfhörer, 2x Koaxkabel	
10.03.2016	WHC Router, analog Telefon	

Nächster Termin:

Datum: Do, 24.03.2016
Zeit: 13:30 Uhr
Ort: HSR, 1.262
Dauer: 1 Stunde

L.5 Sitzung 24. März 2016

Sitzungsprotokoll

Projekt:	**Peer-to-Peer Anwendungen und VoIP**
Woche:	**KW 12**
Ort:	**1.262**
Datum / Zeit:	**24.03.2016 13:30 – 14:30**

Sitzungsteilnehmer

Pascal Meier	pmeier@hsr.ch
Max Obrist	mobrist@hsr.ch
Patrick Eichler	peichler@hsr.ch
Peter Heinzmann	peter.heinzmann@cnlab.ch

Traktanden

- Aktueller Stand besprechen
- Delay-Analyse

Diskussion

- Bandbreitentests ohne „HSR-Netz2
  - fixe IPs
  - Netzwerk auf beiden Seiten des Apposite sniffen
- Messerwerte auch relativ zum Durchschnitt angeben
  - viel bessere Übersicht
- Jitsi Messung mit und ohne Verschlüsselung
  - Grosse Differenz?
- Ergebnisse mit „Charts“ festhalten
  - Detaillierter Bericht folgt später
- RTP Daten nochmals einspielen um den Ausgangsbuffer zu umgehen
  - Colasoft Packet Player
  - Machbarkeitsstudie

Entscheide

Weiteres Vorgehen

- Neue Messungen vornehmen und auswerten

Materialliste:

Datum Abgabe	Gegenstand (an Studenten abgegebenes Material, Bücher, etc.)	Datum Rückgabe
25.02.2016	Apposite, Serie Nr. LR201311009378 HSR Inventarnummer: 118811 mit Power- und Serial-Kabel	
04.03.2016	4x LAN Kabel gründ, HP Switch 8 Port	
04.03.2016	DSO + Netzteil + USB-Verbindung, Audiomessbox, 2 Kopfhörer, 2x Koaxkabel	
10.03.2016	WHC Router, analog Telefon	

L.6 Sitzung 31. März 2016

Sitzungsprotokoll

Projekt:	**Peer-to-Peer Anwendungen und VoIP**
Woche:	**KW 13**
Ort:	**HSR, 1.262**
Datum / Zeit:	**31.03.2016 13:00 – 14:15**

Sitzungsteilnehmer

Pascal Meier	pmeier@hsr.ch
Patrick Eichler	peichler@hsr.ch
Peter Heinzmann	peter.heinzmann@cnlab.ch

Traktanden

- Aktueller Stand besprechen

Diskussion

- Phoner-Lite hat neue Erkenntnisse gezeigt.
    - Zusätzlicher Delay entsteht auch in der Soundkarte und im Ausgabebuffer.

Entscheide

Weiteres Vorgehen

- Erster Teil der Doku fertigstellen und Peter zukommen lassen
- Zwischenpräsentation erstellen

Pendenzen:

Was	Verantwortlichkeit
Theorie Teil an Peter versenden (So)	Max Obrist
Messbericht an Peter versenden (Di)	Pascal Meer
Zwischenpräsentation vorbereiten	Studenten

Materialliste:

Datum Abgabe	Gegenstand (an Studenten abgegebenes Material, Bücher, etc.)	Datum Rückgabe
25.02.2016	Apposite, Serie Nr. LR201311009378 HSR Inventarnummer: 118811 mit Power- und Serial-Kabel	
04.03.2016	4x LAN Kabel gründ, HP Switch 8 Port	
04.03.2016	DSO + Netzteil + USB-Verbindung, Audiomessbox, 2 Kopfhörer, 2x Koaxkabel	
10.03.2016	WHC Router, analog Telefon	

Nächster Termin:

Datum:	Mi, 06.04.2016
Zeit:	13:30 Uhr
Ort:	HSR, 1.262
Dauer:	1 Stunde

L.7 Sitzung 06. April 2016

Sitzungsprotokoll

Projekt:	**Peer-to-Peer Anwendungen und VoIP**
Woche:	**KW 14**
Ort:	**HSR, 1.262**
Datum / Zeit:	**06.04.2016 13:30 – 14:30**

Sitzungsteilnehmer

Pascal Meier	pmeier@hsr.ch
Max Obrist	mobrist@hsr.ch
Patrick Eichler	peichler@hsr.ch
Peter Heinzmann	peter.heinzmann@cnlab.ch

Traktanden

- Präsentation (Draft) besprechen

Beschlüsse (Diskussion):

- Fehlende Punkte in der Präsentation
  - Zeitplan
  - Risikoanalyse
  - Relative Werte zu den M2E Messungen
  - Client Charakteristik
    - Codecs
    - Verschlüsselung
  - Ziele des eigenen Clients
  - Ziel des weiteren Projektes aufzeigen (Qualitätsmessung)
- Zeitplan anpassen
- Projektmanagementelemente in die Doku einbauen
  - Risikoanalyse
- Versuch „RTP Pakete neu zu versenden“ dokumentieren
- „SRTP“ Versuch dokumentieren
- Thesen zu den hohen Delays umformulieren. Keine Annahmen als Bullet setzen
- Swisscom führt anhand unserer „findings“ neue Messungen durch
- Machtbarkeit eines eigenen Clients abschätzen

Pendenzen:

Was	Verantwortlichkeit
Präsentation überarbeiten	Studenten
Dokumentation erweitern	Studenten
SIP Client entwickeln	Max Obrist

Materialliste:

Datum Abgabe	Gegenstand (an Studenten abgegebenes Material, Bücher, etc.)	Datum Rückgabe
25.02.2016	Apposite, Serie Nr. LR201311009378 HSR Inventarnummer: 118811 mit Power- und Serial-Kabel	
04.03.2016	4x LAN Kabel gründ, HP Switch 8 Port	
04.03.2016	DSO + Netzteil + USB-Verbindung, Audiomessbox, 2 Kopfhörer, 2x Koaxkabel	
10.03.2016	WHC Router, analog Telefon	

Nächster Termin:

Datum: Mi, 13.04.2016
Zeit: 08:00 Uhr
Ort: HSR, 1.262
Dauer: 1 Stunde

L.8 Zwischenpräsentation 13. April 2016

Präsentationsprotokoll

Projekt:	**Peer-to-Peer Anwendungen und VoIP**
Woche:	**KW 15**
Ort:	**HSR, 1.223**
Datum / Zeit:	**13.04.2016 08:00 – 09:30**

Sitzungsteilnehmer

Pascal Meier	pmeier@hsr.ch
Max Obrist	mobrist@hsr.ch
Patrick Eichler	peichler@hsr.ch
Peter Heinzmann	peter.heinzmann@cnlab.ch
Reto Hauri	reto.hauri@swisscom.com
Oliver Augenstein	oliver.augenstein@hsr.ch

Traktanden

- Präsentation halten

Beschlüsse (Diskussion):

- Folie 8: Aussage *Audiosamples à 10ms* anpassen. Ausser in der Cisco Literatur ist diese Aussage nicht anzutreffen.
  Normalerweise 2 Samples pro Frame, falsche Aussage. Es sind 160 Samples pro Frame (G.711 8000Hz)
- Folie 9: Abbildung der detaillierten Delayanalyse überarbeiten. Relationen stimmen nicht. Details noch anpassen. Peter gibt noch seinen Input.
- Folie 13: Messmatrix sollte auch die Anzahl Messungen sowie Min-, Max und Standardabweichung beinhalten.
- Folie 14: Statistik über den M2E Delay bei unterschiedlichen Bandbreiten besser darstellen. Grundsätzlich soll klarer werden, was mit „Verhalten sich Erwartungsgemäss“ gemeint ist.
- Auf RTCP bessern eingehen.

Materialliste:

Datum Abgabe	Gegenstand (an Studenten abgegebenes Material, Bücher, etc.)	Datum Rückgabe
25.02.2016	Apposite, Serie Nr. LR201311009378 HSR Inventarnummer: 118811 mit Power- und Serial-Kabel	
04.03.2016	4x LAN Kabel grün, HP Switch 8 Port	
04.03.2016	DSO + Netzteil + USB-Verbindung, Audiomessbox, 2 Kopfhörer, 2x Koaxkabel	
10.03.2016	WHC Router, analog Telefon	

Nächster Termin:

Datum:	Do, 14.04.2016
Zeit:	13:05 Uhr
Ort:	cnlab, Obere Bahnhofstrasse 32B, 8640 Rapperswil
Dauer:	45 Minuten

L.9 Sitzung 14. April 2016

Sitzungsprotokoll

Projekt:	**Peer-to-Peer Anwendungen und VoIP**
Woche:	**KW 15**
Ort:	**cnlab, Obere Bahnhofstrasse 32B, 8640 Rapperswil**
Datum / Zeit:	**14.04.2016 13:15 – 14:00**

Sitzungsteilnehmer

Pascal Meier	pmeier@hsr.ch
Max Obrist	mobrist@hsr.ch
Patrick Eichler	peichler@hsr.ch
Peter Heinzmann	peter.heinzmann@cnlab.ch

Traktanden

- Feedback Doku (Draft)
- Die nächsten Schritte besprechen

Beschlüsse (Diskussion):

- Nach wie vor ein besseres Verständnis zur Qualität von VoIP mit Hauptfokus auf die Identifikation verschiedener Einflüsse auf den M2E-Delay.
- Jitsi wird von Peter und Patrick genauer betrachtet.
- Konkrete Tipps für die Illustration von Messdaten wird Peter uns noch mitteilen.
- Fokus Max
    - Eigener VoIP-Client weiterentwickeln. Sollte bis zum nächsten Meeting funktionsfähig sein.
    - Der Schwerpunkt liegt dabei auf dem Jitter-Buffer und dem Verständnis, was mit den RTP Paketen vor bzw. nach dem Versand passiert.
- Fokus Pascal
    - Messungen/Analyse TORFone
    - Versuche mit IPv4 Multicast im VoIP Umfeld. Genaue Messverfahren sollten aber noch genau erläutert werden.
    - Erklärung der Details vom RTP/RTCP inkl. Wireshark RTP Stream Details
    - Später erneute Messungen in Zusammenarbeit mit der Swisscom

Pendenzen:

Was	Verantwortlichkeit
Client fertig entwickeln	Max Obrist
Alle offenen Protokolle fertigstellen und Peter zukommen lassen	Pascal Meier
Überarbeitete Doku (Teil Pascal) Peter zusenden	Pascal Meier

Materialliste:

Datum Abgabe	Gegenstand (an Studenten abgegebenes Material, Bücher, etc.)	Datum Rückgabe
25.02.2016	Apposite, Serie Nr. LR201311009378 HSR Inventarnummer: 118811 mit Power- und Serial-Kabel	
04.03.2016	4x LAN Kabel gründ, HP Switch 8 Port	
04.03.2016	DSO + Netzteil + USB-Verbindung, Audiomessbox, 2 Kopfhörer, 2x Koaxkabel	
10.03.2016	WHC Router, analog Telefon	

Nächster Termin:

Datum:	Mi, 27.04.2016
Zeit:	10:00 Uhr
Ort:	cnlab, Obere Bahnhofstrasse 32B, 8640 Rapperswil
Dauer:	1 Stunde

L.10 Sitzung 27. April 2016

Sitzungsprotokoll

Projekt:	**Peer-to-Peer Anwendungen und VoIP**
Woche:	**KW 16**
Ort:	**cnlab, Obere Bahnhofstrasse 32B, 8640 Rapperswil**
Datum / Zeit:	**27.04.2016 10:00 – 10:45**

Sitzungsteilnehmer

Pascal Meier	pmeier@hsr.ch
Max Obrist	mobrist@hsr.ch
Patrick Eichler	peichler@hsr.ch
Peter Heinzmann	peter.heinzmann@cnlab.ch

Traktanden

- Die nächsten Schritte besprechen

Beschlüsse (Diskussion):

- Möglicherweise Soundfile direkt in den VoIP-Client einspielen
  - Mehrwert nicht so gross. Idee eher verworfen
- Einflüsse von anderen Sound In-/Outputs auf Soundkarte
  - Einfluss Playout Buffer?
- Fokus Max
  - Im eigenen Client: RTP/RTCP Statistik
  - Doku nachführen (Eigener Client, …)
- Fokus Pascal
  - Delay-Messungen zwischen bestehenden Softphones und „eigenem Client“.
  - Nochmals Audio Multicast recherchieren (Skype, Asterisk)
  - TorFone dokumentieren
  - Delay Grafik überarbeiten (Verhältnisse, Sampling, Quantisierung)
  - Doku nachführen

Pendenzen:

Was	Verantwortlichkeit
Client weiterentwickeln	Max Obrist
Neue Messungen auswerten	Pascal Meier
Doku schreiben	Studenten

Materialliste:

Datum Abgabe	Gegenstand (an Studenten abgegebenes Material, Bücher, etc.)	Datum Rückgabe
25.02.2016	Apposite, Serie Nr. LR201311009378 HSR Inventarnummer: 118811 mit Power- und Serial-Kabel	
04.03.2016	4x LAN Kabel gründ, HP Switch 8 Port	
04.03.2016	DSO + Netzteil + USB-Verbindung, Audiomessbox, 2 Kopfhörer, 2x Koaxkabel	
10.03.2016	WHC Router, analog Telefon	

Nächster Termin:

Datum:	Mi, 11.05.2016
Zeit:	14:00 Uhr
Ort:	HSR, 1.262
Dauer:	1 Stunde

L.11 Sitzung 11. Mai 2016

Sitzungsprotokoll

Projekt:	**Peer-to-Peer Anwendungen und VoIP**
Woche:	**KW 19**
Ort:	**HSR, 1.262**
Datum / Zeit:	**11.05.2016 14:30 – 15:15**

Sitzungsteilnehmer

Pascal Meier	pmeier@hsr.ch
Max Obrist	mobrist@hsr.ch
Peter Heinzmann	peter.heinzmann@cnlab.ch

Traktanden

- Die nächsten Schritte besprechen

Beschlüsse (Diskussion):

- Delay der Soundkarte nochmals analysieren
  - Thread Switch mittels CPU Stresstest provozieren
- Lumisoft Client wird nicht mehr weiterentwickelt
  - Neu wird mit Jitsi entwickelt
- Fokus Max
  - Statistik ausbauen und grafisch darstellen
- Fokus Pascal
  - Verbesserungsvorschläge von Peter in die Doku einfliessen lassen
  - Wireshark Werte aus der RTP Stream Analyse in die Doku einbauen
  - Jitsi Funktionen testen für Max

Pendenzen:

Was	Verantwortlichkeit
Client weiterentwickeln	Max Obrist
Doku schreiben	Studenten

Materialliste:

Datum Abgabe	Gegenstand (an Studenten abgegebenes Material, Bücher, etc.)	Datum Rückgabe
25.02.2016	Apposite, Serie Nr. LR201311009378 HSR Inventarnummer: 118811 mit Power- und Serial-Kabel	
04.03.2016	4x LAN Kabel gründ, HP Switch 8 Port	
04.03.2016	DSO + Netzteil + USB-Verbindung, Audiomessbox, 2 Kopfhörer, 2x Koaxkabel	
10.03.2016	WHC Router, analog Telefon	

Nächster Termin:

Datum:	Do, 19.05.2016
Zeit:	15:00 Uhr
Ort:	cnlab, Obere Bahnhofstrasse 32B, 8640 Rapperswil
Dauer:	1 Stunde

L.12 Sitzung 19. Mai 2016

Sitzungsprotokoll

Projekt:	**Peer-to-Peer Anwendungen und VoIP**
Woche:	**KW 20**
Ort:	**cnlab, Obere Bahnhofstrasse 32B, 8640 Rapperswil**
Datum / Zeit:	**19.05.2016 15:00 – 16:00**

Sitzungsteilnehmer

Pascal Meier	pmeier@hsr.ch
Max Obrist	mobrist@hsr.ch
Peter Heinzmann	peter.heinzmann@cnlab.ch
Patrick Eichler	peichler@hsr.ch

Traktanden

- Konkrete Ziele bis zur Abgabe besprechen. Was muss alles noch von Max und Pascal gemacht werden.
- Im Bericht im Teil „Einleitung“ wurde bei den Verbesserungsvorschlägen von Peter von Messungen Seitens Swisscom gesprochen. Was genau war da gemeint? Und auf welche Folien sollte referenziert werden? (*Swisscom und das cnlab haben die Verzögerung von Mikrophon…*)
- Doku vom Dozenten nochmals überfliegen, Feedback geben

Beschlüsse (Diskussion):

- Verbesserungsvorschläge bezüglich Jitsi-Statistik:
  - Tab Namen besser wählen
  - Dimensionen beachten (unterschiedliche Einheiten)
  - Mehrere Skalen machen
  - Zusammenhängende Charts müssen vergleichbar sein
- CPU Stresstest nachholen (Audio Delay)
- Weitere Messungen seitens Swisscom im Moment nicht notwendig
- Neue Jitsi-Statistik auf Korrektheit überprüfen
- Jitter Buffer/Size/Queue richtig unterscheiden können
- Jitter-Buffer mit einer längeren Szenario testen (Peak provozieren und abwarten ob sich der Jitter-Buffer wieder beruhigt)
- Dokumentationsentwurf bis spätestens am 26.05 versenden damit Peter die Doku anschauen kann

Pendenzen:

Was	Verantwortlichkeit
Doku schreiben	Studenten
Jitsi-Statistik auf korrektheit überprüfen	Pascal
Client weiterentwickeln	Max

Materialliste:

Datum Abgabe	Gegenstand (an Studenten abgegebenes Material, Bücher, etc.)	Datum Rückgabe
25.02.2016	Apposite, Serie Nr. LR201311009378 HSR Inventarnummer: 118811 mit Power- und Serial-Kabel	
04.03.2016	4x LAN Kabel gründ, HP Switch 8 Port	
04.03.2016	DSO + Netzteil + USB-Verbindung, Audiomessbox, 2 Kopfhörer, 2x Koaxkabel	
10.03.2016	WHC Router, analog Telefon	

Nächster Termin:

Datum:	Mi, 01.06.2016
Zeit:	10:00 Uhr
Ort:	cnlab, Obere Bahnhofstrasse 32B, 8640 Rapperswil
Dauer:	1 Stunde

L.13 Sitzung 01. Juni 2016

Sitzungsprotokoll

Projekt:	**Peer-to-Peer Anwendungen und VoIP**
Woche:	**KW 22**
Ort:	**cnlab, Obere Bahnhofstrasse 32B, 8640 Rapperswil**
Datum / Zeit:	**01.06.2016 10:00 – 11:00**

Sitzungsteilnehmer

Pascal Meier	pmeier@hsr.ch
Max Obrist	mobrist@hsr.ch
Peter Heinzmann	peter.heinzmann@cnlab.ch
Patrick Eichler	peichler@hsr.ch

Traktanden

- An der Präsentation soll ebenfalls eine Demo stattfinden. Wie soll das gehen? Der Raum und die Hardware werden dann nicht mehr zur Verfügung gestellt.
- Was genau hat es auf sich mit der Präsentation am 17.06.2016?
- Weiteres Vorgehen besprechen

Beschlüsse (Diskussion):

- Experte braucht ebenfalls ein Exemplar des Berichts. Zudem muss er namentlich auf dem Titelblatt erwähnt werden
- Zur Erinnerung: Es braucht 3 Berichte inkl. CD und einmal nur CD für die HSR
- Abstract bis zum 08.06 fertigstellen. Ab besten vorgängig eine .docx Version erstellen und Peter zum gegenlesen geben
- Abstract sollte in der Doku keine Untertitel enthalten
- Einleitung fällt noch zu knapp aus. Schwerpunkt sollte die Ausgangslage und das Ziele der Arbeit sein
- Abbildungen sollten nicht zu knapp beschrieben werden. Als Beispiel sei die Erklärung zur MOS-Abbildung erwähnt. Es sollte genauer auf das Bild eingegangen werden. (Was macht der Graph?)
- Abbildungen brauchen auch Quellenangaben
- Strukturierung der Dokumentation sollte angepasst werden.
    - Alle Client Themen in ein einzelnes Kapitel verschieben.
    - Messresultate gehören auch in das Kapitel Messungen
    - Kapitel Projektmanagement gehört in den Anhang
    - Schlusswort vor den Anhang

Pendenzen:

Was	Verantwortlichkeit
Abklärung in welcher Form Herrn Augenstein den Bericht erhalten möchte.	Studenten
Dokumentationen ausbauen	Studenten
Abstract verfassen	Studenten
Material zur Rückgabe vorbereiten	Studenten

Materialliste:

Datum Abgabe	Gegenstand (an Studenten abgegebenes Material, Bücher, etc.)	Datum Rückgabe
25.02.2016	Apposite, Serie Nr. LR201311009378 HSR Inventarnummer: 118811 mit Power- und Serial-Kabel	
04.03.2016	4x LAN Kabel gründ, HP Switch 8 Port	
04.03.2016	DSO + Netzteil + USB-Verbindung, Audiomessbox, 2 Kopfhörer, 2x Koaxkabel	
10.03.2016	WHC Router, analog Telefon	

Nächster Termin:

Datum:	Mo, 13.06.2016
Zeit:	10:30 Uhr
Ort:	cnlab, Obere Bahnhofstrasse 32B, 8640 Rapperswil
Dauer:	1 Stunde

L.14 Sitzung 13. Juni 2016

Sitzungsprotokoll

Projekt:	**Peer-to-Peer Anwendungen und VoIP**
Woche:	**KW 24**
Ort:	**cnlab, Obere Bahnhofstrasse 32B, 8640 Rapperswil**
Datum / Zeit:	**13.06.2016 10:30 – 11:30**

Sitzungsteilnehmer

Pascal Meier	pmeier@hsr.ch
Peter Heinzmann	peter.heinzmann@cnlab.ch

Traktanden

- Wo sollen sich folgende Dokumente befinden: Management Summary, Aufgabenstellung, div. Erklärungen
- Aufgabenstellung unterschreiben
- Weiterverwendungsformular unterschreiben
- Besprechung Poster
- Materialrückgabe
- Wie heisst die Kategorie der BA? Ist es „Technologies / Communication“?

Beschlüsse (Diskussion):

- Kapitel 2.3 SIP&RTP und 2.4 RTCP sollten lieber in ein Kapitel mit der Überschrift „VoIP-Protokolle“. Das führt dazu, dass sich alles um eine Hierarchie nach unten verschiebt
- Kapitel 3 SIP-Clients lieber in „VoIP-Clients“ umbenennen, da TORFone kein SIP-Client ist
- Kapitel 3.2 Tor-Client eher in „Anonymisierung“ umbenennen. 3.3 Tor-Netzwerk wäre dann ein Unterkapitel von Anonymisierung. Das Unterkapitel Funktionsweise wäre dann kein Unterkapitel mehr.
- Kapitel 3.5 Client für interne Analysen eher in „Realisierung eigner VoIP-Client (Lumisoft)“ umbenennen.
- Kapitel 3.6 Client für live Analyse eher „Erweiterung Jitsi-Client“ umbenennen.
- Kapitel 4.3 Verhalten eher in „Use Cases“ umbenennen.
- Die ganzen Kapitel 4.3.x vielleicht anpassen, damit nicht immer M2E steht. Oberkapitel könnte einmalig M2E Delay im Namen haben.
- Nach Möglichkeit 4.4.1 und 4.4.2 in 4.3 einbauen. Das hätte zur Folge, dass das Oberkapitel 4.4 „Erkenntnisse Jitsi“ entfallen würde.
- Schlusswort könnte konkretere Aussagen vertragen. Was wäre z.B. bestmöglicher Client.
- Kapitel 4.3.2 Detaillierter M2E Delay eher als erstes Unterkapitel nehmen und in „M2E Einflussfaktoren“ umbenennen. Stellt nachher die Grundlage für die anderen Unterkapitel.
- Kapitel 4.3.1 eher in „M2E Delay bei Referenzsystem“ umbenennen
- Für die Erklärung zur Urheberschaft sollte die Vorlage von Peter verendet werden. Es sollte stehen, wer was gemacht hat.
    - Max: Lumisoft-Client, Jitsi-Erweiterung
    - Pascal: TORFone, Messungen
    - Gemeinsam: Dokumentation
- Im Anhang sollen noch das Inhaltsverzeichnis der CD und die Kontaktdaten der Studenten abgebildet werden.

- Das Plakat mit weniger Inhalt wurde von Peter bevorzug.
  Feedback Version 1 (wenig Text):
  - Die Box „Ziele“ sollte zwei Bullet-Points haben und nicht Fliesstext
  - Box „Technologien“ durch etwas anderes anpassen.

  Feedback Version 2 (viel Text):
  - Teil Messverfahren liest sich ein wenig holperig
  - 3. Kapitel eher in VoIP-Qualität umbenennen
  - 3. Grafik anpassen. Audio Codecs in die Säule VoIP-Client einfliessen lassen. Wobei Pascal es bevorzugen würde die Grafik so zu belassen aber im Teil VoIP-Client die Begriffe „Grösse Jitter-Buffer“ und „Verhalten Jitter-Buffer“ einfliessen zu lassen
  - Grafik oder Text zu Ergebnisse wünschenswert. Vielleicht eine Grafik mit den unterschiedlichen M2E Delays als Balkendiagramm.
- Material am Mittwoch zurückgeben und die Thematik mit Patrick anschauen.

Pendenzen:

Was	Verantwortlichkeit
Materialrückgabe	Studenten
Dokumentation abschliessen	Studenten

Materialliste:

Datum Abgabe	Gegenstand (an Studenten abgegebenes Material, Bücher, etc.)	Datum Rückgabe
25.02.2016	Apposite, Serie Nr. LR201311009378 HSR Inventarnummer: 118811 mit Power- und Serial-Kabel	
04.03.2016	4x LAN Kabel gründ, HP Switch 8 Port	
04.03.2016	DSO + Netzteil + USB-Verbindung, Audiomessbox, 2 Kopfhörer, 2x Koaxkabel	
10.03.2016	WHC Router, analog Telefon	