VERHALTENSANALYSE VON EINPLATINENCOMPUTERN BEIM TRANSCODING VON ECHTZEIT-AUDIODATEN

Martin Meszaros, Michael Maruschke

Hochschule für Telekommunikation Leipzig (HfTL)
{*meszaros; maruschke*}*@hft-leipzig.de*

Kurzfassung: Zur Erreichung der Interoperabilität von Endnutzern, welche unterschiedliche Audiocodecs nutzen, wird eine Umwandlung (Transcoding) zwischen den jeweils zum Einsatz kommenden Codecs durchgeführt. Dieser Beitrag beschäftigt sich mit der Frage, welchen Einfluss die durch das Transcoding entstehende Verzögerung auf das Ende-zu-Ende-Delay der Kommunikation hat. Getestet wurden hier insbesondere die HD-Voice fähigen Audiocodecs Opus und G.722 beim Einsatz eines Einplatinencomputers als Transcoding-Einheit. Weiterhin wird mittels POLQA-Testverfahren untersucht, welche Codec-Kombinationen den geringsten Qualitätsverlust beim Transcoding erreichen.

1 Einleitung

Mit Einführung der IP-basierten, öffentlichen Telefonie ist es erstmals möglich, sogenannte High-Definition (HD)-fähige Audiocodecs einzusetzen. Diese ermöglichen eine bessere Sprachqualität durch Ausnutzung eines größeren Frequenzbandes als die bisher im traditionellen Festnetz eingesetzten Sprachcodecs. Während der ausschließlich Narrowband-Telefonie unterstützende Codec G.711 ein Frequenzband von 0,3 - 3,4 kHz nutzt, können die HD-Voice-fähigen Codecs (z. B. G.722, Opus) einen Frequenzbereich von 50 Hz - 7 kHz ausschöpfen (Wideband-Telefonie).
Der Opus-Codec ist als sehr flexibler Sprach- und Audiocodec für Frequenzbereiche von 50 Hz bis maximal 20 kHz (Fullband) von der Internet Engineering Task Force (IETF) standardisiert wurden [1]. Mit Web Real-Time Communication (WebRTC) existiert eine standardisierte Technologie, welche es ermöglicht, unter Zuhilfenahme eines modernen Webbrowsers in Echtzeit miteinander zu kommunizieren [2]. Dazu ist es nicht notwendig, zusätzliche Plugins zu installieren. Die Application Programming Interface (API) wird unter anderem von Google, Mozilla und Opera entwickelt [3] und ist schon seit einiger Zeit in deren Browsern integriert. WebRTC nutzt für die Echtzeittelefonie vorzugsweise den Opus-Codec. Damit steht allen Webbrowser-basierten Endgeräten (z. B. Laptop, Tablet, Smartphone) ein sehr leistungsfähiger Sprach- und Audiocodec zur Verfügung, welcher in CD-Qualität arbeiten kann.
In [4] wird eine Möglichkeit vorgestellt, mithilfe eines WebRTC-fähigen Browsers, Telefonanrufe in das öffentliche IP-basierte Festnetz in HD-Voice-Qualität zu tätigen. Das Transcoding der jeweils zum Einsatz kommenden unterschiedlichen Codecs ist dabei zu berücksichtigen. Typischerweise wird für HD-Voice Telefonie im Festnetz der Widebandcodec G.722 verwendet, für alle anderen Festnetz-Telefonate der bekannte Narrowbandcodec G.711. Der Endnutzer der browser-basierten WebRTC-Telefonie nutzt hingegen standardmäßig den Opus-Codec. Im Google Chrome-Browser kann der Entwickler der Web-App weitere alternative Sprach-Codecs wie G.722, G.711 oder Speex für die Kommunikation auswählen (vergleiche dazu auch [5]).
Um die Umwandlung von WebRTC-Codecs in Festnetz-Codecs durchzuführen, muss ein Gateway eingesetzt werden, welches neben dem notwendigen Transcoding auch eine Anpassung

der unterschiedlichen zum Einsatz kommenden Signalisierungsprotokolle vornimmt. Zur praktischen Realisierung verwenden die Autoren in Anlehnung an [4] den Einplatinencomputer Raspberry Pi 2 Modell B [6] mit der Gateway Software *webrtc2sip* [7], welche in der Lage ist, obige Anpassungen durchzuführen. Diese Kombination stellt ein kostengünstiges WebRTC-zu-VoIP-Gateway dar und kann somit im Heimgebrauch oder in kleinen Unternehmen eingesetzt werden, um mit WebRTC-fähigen, IP-basierten Endgeräten in das Festnetz zu telefonieren.
Aufbauend auf der Lösung von [4] beschäftigt sich dieser Beitrag mit folgenden Fragen:

- Welchen Einfluss hat das Transcoding auf die Verzögerung bei der Echtzeit-Audio-Kommunikation?
- Inwiefern wirkt sich das Transcoding auf die Sprachqualität aus?
- Ist ein Einplatinencomputer für den Einsatz als Transcoding-Einheit geeignet?

In Abschnitt 2 wird der Einfluss der Verzögerung bei Echtzeitkommunikation auf die Sprachqualität skizziert. Während in Abschnitt 3 die Autoren ihr Konzept zur Bestimmung der beim Transcoding-Prozess auftretenden Verzögerung vorstellen, erfolgt im nachfolgenden Abschnitt 4 die Auswertung der erzielten Ergebnisse. Dabei werden insbesondere die am Next Generation Network (NGN)-Festnetz der Deutschen Telekom AG eingesetzten Sprachcodecs G.722, G.711 sowie der für WebRTC typische Opus-Codec betrachtet.

2 Verzögerung bei Echtzeit-Audiokommunikation

Ein wichtiger Qualitätsfaktor bei Echtzeitkommunikation ist das sogenannte One-Way-Delay, oft auch Mund-zu-Ohr-Verzögerung genannt. Dabei handelt es sich um die Zeit, die benötigt wird, um die Sprache vom Mikrofon von Partei A zum Ohr von Partei B zu transportieren. Dies schließt unter anderem die Zeit-Dauer folgender Teilprozesse[1] ein:

- Analog-Digital Wandlung,
- Encoding-Vorgang auf Senderseite,
- Transport der Sprachdaten im Netzwerk,
- Decoding-Vorgang auf Empfängerseite,
- Digital-Analog Wandlung.

Die ITU-T Spezifikation G.114 [8] empfiehlt, einen maximalen One-Way-Delay-Wert in Höhe von 400 ms nicht zu überschreiten, da ansonsten die Sprachqualität signifikant vermindert wird. Weiterhin wird angegeben, dass bei einer Mund-zu-Ohr-Verzögerung von weniger als 150 ms keinerlei negative Auswirkungen auf die Sprachqualität festzustellen sind. Moderne Audio-Codecs arbeiten Frame-basiert. Das heißt, sie sammeln eine gewisse Anzahl an Signalproben und komprimieren diese in einen Frame. Hierbei kann dieser Frame erst dann generiert werden, wenn alle benötigten Signalproben vom Encoder gesammelt wurden. Um die Kompressionseffizienz zu erhöhen, betrachten moderne Audio-Codecs zusätzlich einen Teil des nachfolgenden Frames. Die Dauer für diesen Vorgang wird als *Look-Ahead* bezeichnet.

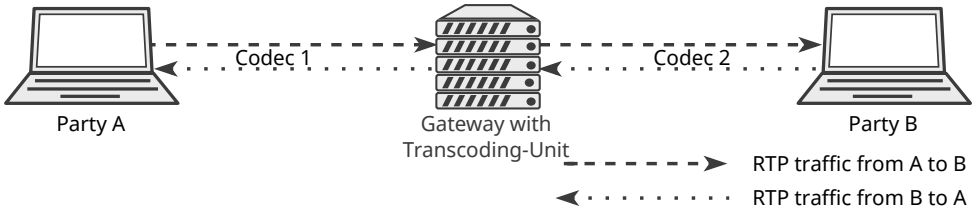

Abbildung 1 - Nutzdatenströme zwischen Clients und einer Transcoding-Unit

3 Messung des Transcoding Delays

3.1 Messverfahren

Die Autoren entwickelten ein Verfahren zur Messung des Transcoding-Delays in Voice over IP (VoIP) Anwendungen. Hierbei wird der ein- und ausgehende Datenverkehr der Transcoding-Unit mittels *Wireshark* aufgezeichnet. Die encodierten Sprachdaten liegen eingebettet in Real-Time Transport Protocol (RTP)-Paketen vor. Wie in Abbildung 1 sichtbar, sind bei der Kommunikation von zwei Gesprächspartnern vier RTP-Ströme beteiligt:

- Party A zur Transcoding-Unit in Codec 1,
- Transcoding-Unit zu Party B in Codec 2,
- Party B zu zur Transcoding-Unit in Codec 2,
- Transcoding-Unit zu Party A in Codec 1.

Die *Wireshark*-Aufzeichnung enthält außerdem die relative Ankunftszeit eines jeden Paketes, seit Beginn der Aufzeichnung. Zur Bestimmung der Transcoding-Dauer wird nun die Differenz der Zeitpunkte gebildet, zu dem ein Paket in Codec 1 die Transcoding-Unit erreicht (Zeitpunkt 1) und zu dem das korrespondierende Paket in Codec 2 diese verlässt (Zeitpunkt 2). Als Herausforderung stellte sich das Auffinden der korrespondierenden Pakete in den verschiedenen Codecs dar. Aufgrund von Initialisierungsvorgängen in der Transcoding-Unit werden eingehende Pakete bei Kommunikationsbeginn von dieser verworfen, so dass eine unmittelbare und exakte zeit-korrelierte Zuordnung zwischen den RTP-Paketen der Ein- und Ausgangsseite nicht möglich ist.
Um eine solche Einordnung der korrespondierenden Nutzdatenpakete zu ermöglichen, wurde das decodierte Audiosignal an sich analysiert. Hierfür werden in einem ersten Schritt die Codec-Nutzdaten aus den RTP-Paketen extrahiert. Für sample-basierte Codecs wie G.711 oder G.722 bietet *Wireshark* prinzipiell eine Möglichkeit, die reinen Nutzdaten als Rohdaten zu speichern. Hierbei haben alle RTP-Pakete die gleiche Größe. Frame-basierte Codecs untersuchen mehrere Abtastproben des Audiosignales vor dem eigentlichen Codiervorgang hinsichtlich typischer Merkmale mit dem Ziel, den zu übertragenden Payload zu minimieren. Aus diesem Grund arbeiten frame-basierten Codecs, wie Opus, mit einer variablen Payload-Größe (RTP-Nutzdaten). Um die Opus-Frames aus dem RTP-Strom extrahieren zu können, muss die Größe jedes Frames bekannt sein. *Wireshark* bietet die Möglichkeit, die RTP-Pakete der jeweiligen RTP-Ströme als ein sogenanntes „RTP-Dump“ zu speichern. Hierbei werden die darunter liegenden Protokollschichten (wie beispielsweise UDP und IP) nicht mit extrahiert. Jedes RTP-Paket in dem RTP-Dump besitzt zusätzliche Header-Informationen, die die Payload-Größe angeben.
Die Entwickler des Opus-Referenzcodecs bieten mit den *Opustools* [9] eine Sammlung von

[1] Obige Aufzählung ist nicht vollständig. So zählt beispielsweise der Empfangspuffer auf der Empfängerseite mit dazu.

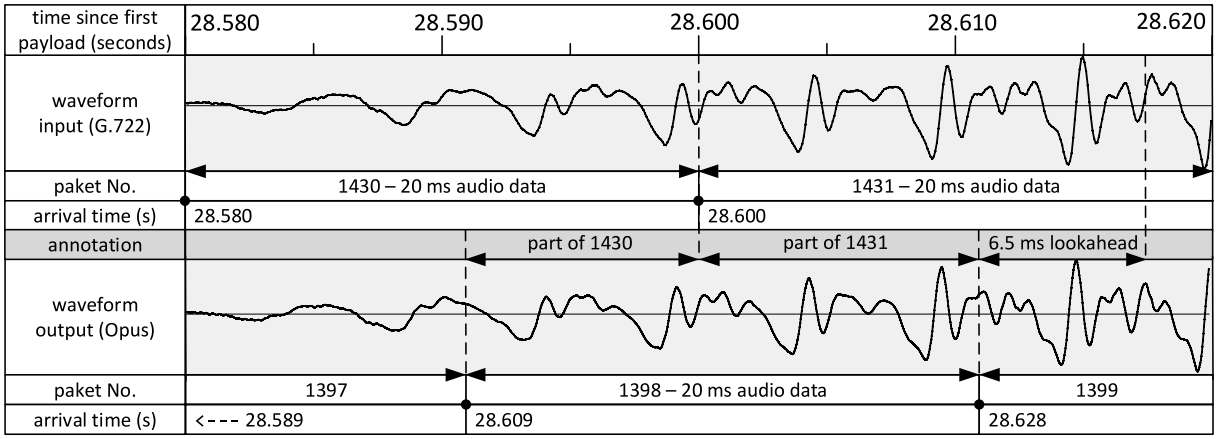

Abbildung 2 - Auschnitt aus dem Transcoding von G.722- zu Opus-Sprachdaten

Programmen an, die es erlauben Wave-Dateien in Opus-Dateien umzuwandeln (Encoder) und diese zu decodieren. Der Opus Referenzdecoder aus den *Opustools* verlangt, dass sich die Opus-Frames in einem Ogg[2] Container-Format befinden. Um das für die Zuordnung nötige Decoding durchführen zu können, modifizierten die Autoren das Programm *Opusrtp* aus den *Opustools* [10] in der Weise, dass es in der Lage ist, ein RTP-Dump mit Opus-codierten-Audiodaten in eine Ogg-Datei umzuwandeln. An dieser Stelle konnten die unterschiedlichen Codec-Ströme decodiert und im Anschluss die korrespondierenden Pakete bestimmt werden.

3.2 Bestimmung des durch Transcoding verursachten Delays

Die dekodierten Audioströme können mit der Open-Source Software *Audacity* angezeigt werden [11]. Jedes RTP-Paket enthält 20 ms an Audiodaten. Dadurch können jeweils 20 ms eines Audiostromes exakt einem RTP-Paket zugeordnet werden, inklusive des relativen Zeitpunktes, zu welchem das Paket von *Wireshark* erfasst wurde. In Abbildung 2 ist beispielhaft ein Ausschnitt des Transcoding-Szenarios G.722 zu Opus dargestellt. Die obere Zeitleiste gibt den Zeitpunkt an, zu dem das erste RTP-Paket von *Wireshark* erfasst wurde. Im Beispiel stimmt diese mit der Ankunftszeit (*arrival time*) der G.722 Pakete überein. Des weiteren sind jeweils 20 ms Audiodaten markiert (Bereich zwischen zwei gestrichelten Linen), die in dem mit *paket No.* angegebenen RTP-Paket enthalten waren, mit der jeweiligen Ankunftszeit des Paketes in *Wireshark*.
Wie in Abbildung 2 durch den Audio-Signalverlauf zu erkennen ist, setzt sich das Opus-Paket mit der Paket-Nummer 1398 aus Teilen der zwei G.722 Pakete mit den Nummern 1430 und 1431 zusammen (vergleiche auch Zeile *„annotation“* in Abbildung 2). Da der Opus-Encoder immer Pakete von 20 ms Dauer bildet, reicht der Inhalt vom G.722 Paket mit der Nummer 1430 nicht aus. Der Encoder muss deshalb 20 ms auf das Eintreffen des nächsten G.722 Paketes warten, um Opus-Paket Nummer 1398 encodieren zu können (Paketierung). Nach Eintreffen des G.722 mit Paketes Nummer 1431 kann der Opus-Encoder sofort mit dem Encoding-Vorgang beginnen, da nun die benötigten 20 ms an Audiodaten und die zusätzlich vom Encoder benötigten 6,5 ms an *Look-Ahead* vorliegen [12]. Im Beispiel lässt sich die Transcoding-Zeit berechnen, indem die Ankunftszeit an *Wireshark* von Paket 1398 (Opus) von der Ankunftszeit des Paketes 1431 (G.722) subtrahiert wird: $28,609s - 28,600s = 9ms$. Verallgemeinert lässt sich die Codec-Verarbeitungszeit des n-ten Paketes t_{n_v} wie folgt bestimmen:

$$t_{n_v} = t_{n_a} - t_{n_e} \tag{1}$$

[2] Ogg ist ein Container-Dateiformat für Audio-, Video- und Textdaten [9]

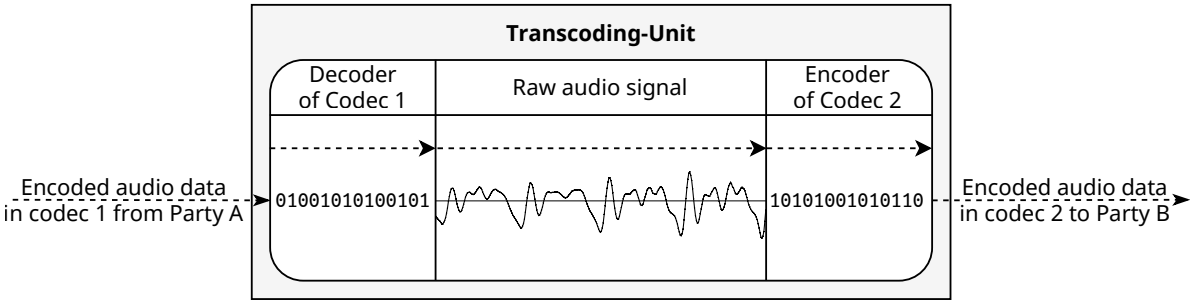

Abbildung 3 - Detailansicht der Transcoding-Unit

wobei t_{n_a} den Zeitpunkt des n-ten Paketes am Ausgang und t_{n_e} den Zeitpunkt des n-ten Paketes am Eingang der Transcoding-Unit darstellt.
Die Zeit Δt_{n_e}, die der Encoder paketierungsbedingt auf das zusätzliche Paket warten muss, berechnet sich folgendermaßen:

$$\Delta t_{n_e} = t_{n_e} - t_{(n-1)_e} \tag{2}$$

Die gesamte Transcoding-Zeit $t_{n_{transcoding}}$ kann also wie folgt ermittelt werden:

$$t_{n_{transcoding}} = \Delta t_{n_e} + t_{n_v} \tag{3}$$

Das Paketierungsintervall Δt_{n_e} kommt zustande, weil der Transcoding-Vorgang innerhalb der Transcoding-Unit zeitlich asynchron stattfindet. In Abbildung 3 ist die schematische Funktionsweise der Transcoding-Unit dargestellt. Hierbei wird ersichtlich, dass sich der Transcoding-Vorgang aus drei Schritten zusammensetzt:

1. Umwandlung (Decoding) der in Codec 1 codierten Audiosignale in Audio-Rohdaten,
2. Übergabe der Audio-Rohdaten an den Encoder von Codec 2,
3. Umwandlung (Encoding) der Audio-Rohdaten in Codec 2 codierte Audiosignale.

Hierbei verarbeitet der Decoder die ankommenden Audio-Pakete zeitlich unabhängig von der Encoder-Funktion. Weiterhin bedeutet dies, dass erst wenn ausreichend Audio-Daten (20 ms Dauer an Audio-Rohdaten) an den Encoder übergeben wurden, diese encodiert werden. Das ist auch der Grund für die fehlenden Audiodaten am Ausgang des Transcoders bei Kommunikationsbeginn. Die Transcoding-Einheit beginnt mit der Decodierung der eingehenden Daten, sobald diese ankommen. Der Encoder befindet sich zu diesem Zeitpunkt allerdings noch in der Initialisierungsphase und die Audio-Rohdaten können nicht an den Encoder übergeben werden. Erst wenn der Encoder bereit ist, werden die aktuell anliegenden, decodierten Daten an diesen übergeben.

3.3 Messsetup

Um nicht-überprüfbare Einflüsse seitens des öffentlichen Telefonnetzes beim Transcoding auszuschließen, wurden die Messungen in einer lokalen Testumgebung durchgeführt, welche in Abbildung 4 dargestellt ist.
Alle verwendeten Endgeräte benutzen ein 64-Bit basiertes Linux Betriebssystem. Diese sind über eine Cisco SG-300 Switch per Gigabit Ethernet miteinander verbunden. Um die ein- und ausgehenden Pakete des Gateways mit Transcoding-Funktion aufzeichnen zu können, wird die „Port Mirroring“ Funktion am Cisco Switch genutzt. Damit werden alle ein- und ausgehenden

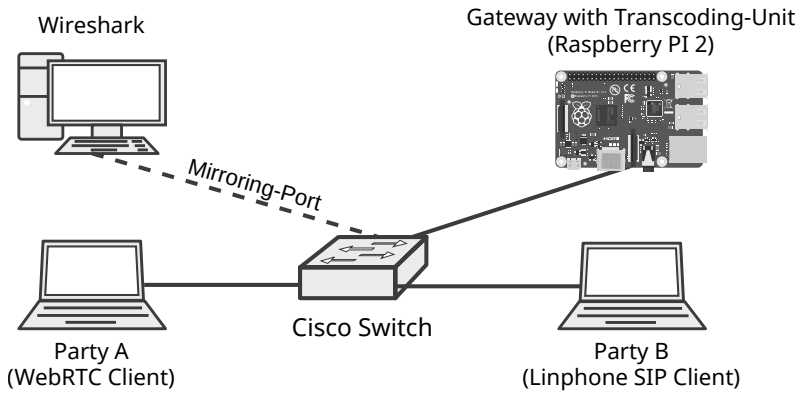

Abbildung 4 - Setup zur Messung von Transcoding-Delay

Datenpakete gleichzeitig an einen weiteren, an der Sprachkommunikation unbeteiligten, PC weitergeleitet, welcher diese mit dem Netzwerk-Protokoll-Analysator *Wireshark* aufzeichnet. Im Gegensatz zum unmittelbaren Paketmitschnitt direkt auf dem Gateway/Transcoding-Unit, wird somit einer zusätzliche Performance-Belastung des Einplatinensystems entgegengewirkt. Zum Einsatz kommt im Testbed ein Raspberry Pi 2 Modell B mit einem 900MHz Quad-Core ARM Cortex-A7 Prozessor sowie einem Arbeitsspeicher von 1 GB RAM [6]. Die beiden Gesprächspartner werden durch einen WebRTC-Client mit Google Chrome-Browser (Party A) und einen *Linphone* SIP-Client (Party B) realisiert.
Zur Bestimmung der Sprachqualität wurde das instrumentelle POLQA-Testverfahren eingesetzt [13]. Dabei werden die an der Transcoding-Unit eingehenden Sprachdaten mit den ausgehenden verglichen. Die erzielte Sprachqualität wird dabei üblicherweise in der Mean Opinion Score (MOS)-Werte-Skala, welche einen Wertebereich von 1 bis 5 beinhaltet, dargestellt.

4 Ergebnisse

4.1 Codecverhalten

Transcoding-Typ	**Transcoding mit Paketierungsintervall ($\bar{t}_{transcoding}$ in ms)**	**Transcoding ohne Paketierungsintervall ($\bar{t}_v$ in ms)**	**POLQA MOS**
Opus → G.722	21,71	1,79	4,63
Opus → G.711	22,03	2,05	3,86
G.722 → Opus	25,67	5,67	4,75
G.711 → Opus	26,16	6,16	4,00
G.722 → G.722	21,11	1,13	4,71
Opus → Opus	27,27	7,27	4,75

Tabelle 1 - Transcoding-Zeiten und MOS-Werte verschiedener Codecs

Zur Evaluierung verwendeten die Autoren natürliche (Fullband) Sprach-Samples von sowohl Männern als auch Frauen aus der EUN-Veith Datenbank [14]. Zur Ermittlung der Transcoding-Zeiten der verschiedenen Codecs wurden Sprachproben von jeweils einer Minute Dauer eingesetzt, welche digital in den WebRTC- und SIP-Client eingespielt wurden. Der Einfluss auf die Sprachqualität wurde im Anschluss mittels POLQA-Testverfahren ermittelt. Alle Messungen wurden mehrmals durchgeführt. Tabelle 1 enthält die Mittelwerte der Messergebnisse.
Wie sich in Tabelle 1 erkennen lässt, sind die Transcoding-Zeiten mit Paketierungsintervall

stets um ca. 20 ms länger als die unmittelbaren Transcoding-Zeiten ohne Paketierung. In Spalte 3 wird ersichtlich, dass die reine Verarbeitungszeit bei den Sample-basierten Codecs bei ca. 2 ms liegt. Wird hingegen in den Opus-Codec encodiert, steigt diese Zeit auf einen Wert von mehr als 5 ms an. Die Begründung hierfür ist in der systembedingt hohen Komplexität des Opus-Encoders [1] zu vermuten. Interessanterweise ergeben sich beim (None-) Transcoding-Fall Opus $\rightarrow$ Opus bzw. G.722 $\rightarrow$ G.722 ähnliche Werte, wie wenn ein Transcoding in jeweils unterschiedliche Codecs stattfindet. Dies lässt vermuten, dass im *webrtc2sip*-Framework auch dann ein Transcoding stattfindet, wenn der gleiche Codec-Typ am Ein- und Ausgang eingesetzt wird.
Die gesamte Transcoding-Verzögerung mit Paketierungsintervall, die ein Raspberry Pi 2 als Transcoding-Unit zur Mund-zu-Ohr-Verzögerung beiträgt, liegt zwischen 21 ms und 27 ms. Diese Transcoding-Zeit wirkt sich somit nicht signifikant auf die Mund-zu-Ohr-Verzögerung aus, wenn alle weiteren Netzwerk- und Endgeräte-bedingten Verzögerungen einen One-Way-Delay-Wert in Höhe von circa 375 ms nicht überschreiten [8].
Für den Einfluss auf die Sprachqualität wurden alle Transcoding-Typen außer Opus $\leftrightarrow$ G.711 im POLQA-Superwideband-Modus getestet. Der Transcoding-Typ Opus $\leftrightarrow$ G.711 wurde im POLQA-Narrowband-Modus getestet. POLQA legt für jeden dieser Modi Obergrenzen für den maximal zu erreichenden MOS-Wert fest. Im Superwideband-Modus wird diese Obergrenze mit einem MOS-Wert von 4,75 angegeben, währenddessen im Narrowband-Modus ein oberer MOS-Grenzwert von 4,5 vorgegeben ist. Aus Spalte 4 von Tabelle 1 wird weiterhin ersichtlich, dass beim Transcoding-Typ Opus $\leftrightarrow$ G.711 MOS-Werte zwischen 3,86 und 4,00 erreicht werden. Dies lässt sich mit dem Informationsverlust durch die Frequenzband-Begrenzung beim Transcoding-Vorgang von einem Fullband- in einen Narrowband-Codec erklären. Aus dem Transcoding-Typ Opus $\leftrightarrow$ G.722 resultieren sehr gute MOS-Werte zwischen 4,63 und 4,75. Bei einem Transcoding zu Opus hin wird die Obergrenze im POLQA-Superwideband-Modus erreicht. Auch beim Transcoding von gleichen Codec-Typen wird der maximal zu erreichende Wert für den Opus-Codec erzielt, wohingegen bei G.722 $\leftrightarrow$ G.722 der MOS-Wert 4,71 beträgt.

4.2 Stresstest

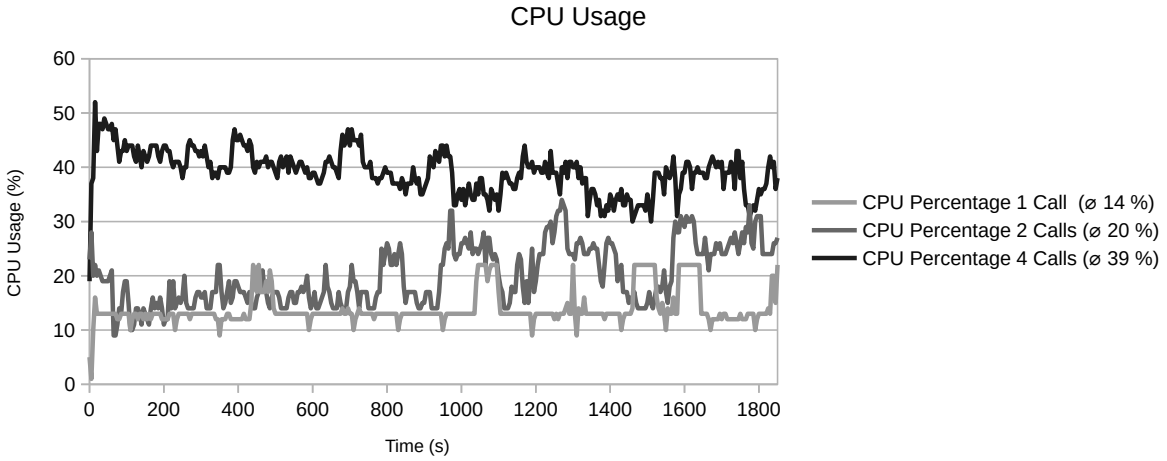

Abbildung 5 - CPU-Auslastung über eine Laufzeit von einer halben Stunde

Um das Verhalten eines Einplatinencomputers als Transcoding-Unit zu analysieren, wurde als zentraler Performance-Kennwert die CPU-Auslastung näher betrachtet. Hierbei wurde der Fokus insbesondere auf die Transcoding-Fälle der HD-Voice-fähigen Codecs Opus $\leftrightarrow$ G.722 ge-

legt. Es wurden dabei mehrere gleichzeitige Sprachverbindungen untersucht. In Abbildung 5 ist die CPU-Auslastung beim Transcoding von einem, zwei beziehungsweise vier gleichzeitigen Gesprächen für die Dauer einer halben Stunde dargestellt. Während die durchschnittliche CPU-Auslastung bei einem Call ca. 14 % beträgt, steigt sie für zwei Calls auf 20 % und für vier Calls auf 39 % an. Mit diesen Resultaten ist ersichtlich, dass ein Raspberry Pi 2 mit *webrtc2sip*-Gateway-Software durchaus in der Lage ist, mindestens vier gleichzeitige bidirektionale Calls zu transcodieren.
Tabelle 2 stellt die Transcodingzeiten und MOS-Werte bei zwei beziehungsweise vier gleichzeitigen Calls dar. Dabei ist ersichtlich, dass bei zwei Calls die Delay- beziehungsweise POLQA-MOS-Werte mit denen aus Tabelle 1 vergleichbar sind. Im Fall von vier Calls ist festzustellen, dass die Delay-Zeiten um 1 bis 2 ms ansteigen, währenddessen die POLQA-MOS-Werte unverändert bleiben.

Transcoding-Typ	**Transcoding mit Paketierungsintervall ($\bar{t}_{transcoding}$ in ms)**	**Transcoding ohne Paketierungsintervall ($\bar{t}_v$ in ms)**	**POLQA MOS**
2 Calls			
Opus → G.722	22,36	2,38	4,63
G.722 → Opus	25,48	5,48	4,75
4 Calls			
Opus → G.722	23,12	3,14	4,63
G.722 → Opus	27,66	7,66	4,75

Tabelle 2 - Transcoding-Zeiten und MOS-Werte bei mehreren gleichzeitigen Calls

5 Schlussfolgerungen

Dieser Beitrag zeigt, dass der Einplatinencomputer Raspberry Pi 2 durchaus geeignet ist — ohne nachweisbaren Qualitätsverlust bei Sprachverbindungen — als Transcoding-Einheit eingesetzt zu werden. Dies gilt unter folgenden Bedingungen:

- getestet wurde ausschließlich das Transcoding mit den Codecs Opus, G.722 und G.711,
- der Eignungsnachweis erfolgte ausschließlich unter prototypischen VoIP-/WebRTC-Labortestbedingungen und nicht am öffentlichen NGN-Anschluß,
- für die notwendige WebRTC-zu-VoIP-Wandlung wurde ausschließlich die *webrtc2sip*-Gateway-Software getestet,
- der Eignungsnachweis erfolgte nur für maximal vier gleichzeitige Sprachverbindungen,
- die durch das Transcoding auftretende Verzögerung von maximal 25 ms führt nicht zu einer kritischen Erhöhung der insgesamt zulässigen Mund-zu-Ohr-Verzögerung.

Im Heimbereich ist es damit möglich, Webbrowser-basiert und in HD-Voice-Qualität zu kommunizieren, ohne Qualitätseinbußen hinnehmen zu müssen.
Weitere Untersuchungen unter Einbindung von HD-Voice-fähigen Codecs des Mobilfunks (z. B. AMR-WB) sind sinnvoll. Die Autoren halten eine Analyse des Einflusses auf die Ende-zu-Ende Sprachqualität angebracht für den Fall, dass bei der Kommunikation in öffentlichen Netzen, über weite Strecken oder Carrier-Grenzen hinweg, mehrere Transcoding-Vorgänge zeitlich nacheinander stattfinden müssen.

Danksagungen

Die Autoren danken der SwissQual AG, Schweiz (insbesondere Herrn Dr. Jens Berger) für die Bereitstellung der POLQA-Testsoftware. Weiterhin gilt der Dank Herrn Ferdinand Malcher für die Unterstützung bei der Verwendung der *webrtc2sip*-Software auf dem Einplatinensystem Raspberry Pi 2.

Literatur

[1] J. FALIN, K.VOS und T. TERRIBERRY, *Definition of the opus audio codec*, RFC 6716 (Proposed Standard), Internet Engineering Task Force, Sep. 2012. Adresse: http://www.ietf.org/rfc/rfc6716.txt.

[2] C. JENNINGS, A. NARAYANAN, D. BURNETT und A. BERGKVIST, "WebRTC 1.0: Real-time communication between browsers", W3C, W3C Editor's Draft, Okt. 2015. Adresse: http://w3c.github.io/webrtc-pc/ (besucht am 10. 11. 2015).

[3] GOOGLE INC. (Sep. 2015). Webrtc, Adresse: http://www.webrtc.org/.

[4] F. MALCHER, "Integration von WebRTC in einen All-IP-Telefonanschluss der Deutschen Telekom", Bachelorarbeit, HfTL, Leipzig, 28. Aug. 2015.

[5] M. MARUSCHKE, O. JOKISCH, M. MESZAROS und V. IAROSHENKO, "Review of the opus codec in a WebRTC scenario for audio and speech communication", in *Speech and Computer*, Springer, 2015, S. 348–355.

[6] RASPBERRY PI FOUNDATION. (2015). Raspberry Pi 2 Model B, Adresse: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/.

[7] DOUBANGO TELECOM. (2015). Webrtc2sip - smart SIP and media gateway to connect WebRTC endpoints, Adresse: http://webrtc2sip.org/.

[8] ITU-T, "General recommendations on the transmission quality for an entire international telephone connection-one-way transmission time", International Telecommunication Union (Telecommunication Standardization Sector), REC G.114, Mai 2003. Adresse: http://www.itu.int/rec/T-REC-G.114-200305-I/en.

[9] XIPH.ORG FOUNDATION. (2016). Xiph.org, Adresse: https://www.xiph.org/.

[10] M. MESZAROS. (2015). Opus-tools · GitLab, Adresse: https://gitlab.com/MoeRT09/opus-tools.

[11] AUDACITY TEAM. (2016). Audacity: Free Audio Editor and Recorder, Adresse: http://audacityteam.org/.

[12] J.-M. VALIN, G. MAXWELL, T. B. TERRIBERRY und K. VOS, "High-quality, low-delay music coding in the opus codec", in *Audio Engineering Society Convention 135*, Audio Engineering Society, 2013.

[13] ITU-T, "Methods for objective and subjective assessment of speech quality (POLQA): Perceptual objective listening quality assessment", International Telecommunication Union (Telecommunication Standardization Sector), REC P.863, Sep. 2014. Adresse: http://www.itu.int/rec/T-REC-P.863-201409-I/en.

[14] O. JOKISCH, A. WAGNER, R. SABO, R. JAECKEL, N. CYLWIK, M. RUSKO, A. RONZHIN und R. HOFFMANN, "Multilingual speech data collection for the assessment of pronunciation and prosody in a language learning system", in *Proc. of 13th Intern. Conf. SPECOM*, 2009, S. 515–520.