ETSI TR 103 890 V1.1.1 (2023-10)

TECHNICAL REPORT

Speech and multimedia Transmission Quality (STQ); Design of a generic approach to test network performance for OTT conversational voice applications

Reference

DTR/STQ-00231m

Keywords

4G, 5G, OTT voice applications, OTT voice generic clients, QoS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00   Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Important notice

The present document can be downloaded from:
https://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

If you find a security vulnerability in the present document, please report it through our
Coordinated Vulnerability Disclosure Program:
https://www.etsi.org/standards/coordinated-vulnerability-disclosure

Notice of disclaimer & limitation of liability

The information provided in the present deliverable is directed solely to professionals who have the appropriate degree of experience to understand and interpret its content in accordance with generally accepted engineering or other professional standard and applicable regulations.
No recommendation as to products and services or vendors is made or should be implied.
No representation or warranty is made that this deliverable is technically accurate or sufficient or conforms to any law and/or governmental rule and/or regulation and further, no representation or warranty is made of merchantability or fitness for any particular purpose or against infringement of intellectual property rights.
In no event shall ETSI be held liable for loss of profits or any other incidental or consequential damages.

Any software contained in this deliverable is provided "AS IS" with no warranties, express or implied, including but not limited to, the warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property rights and ETSI shall not be held liable in any event for any damages whatsoever (including, without limitation, damages for loss of profits, business interruption, loss of information, or any other pecuniary loss) arising out of or related to the use of or inability to use the software.

Copyright Notification

No part may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm except as authorized by written permission of ETSI.
The content of the PDF version shall not be modified without the written authorization of ETSI.
The copyright and the foregoing restriction extend to reproduction in all media.

© ETSI 2023.
All rights reserved.

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Speech and multimedia Transmission Quality (STQ).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Executive summary

OTT voice application testing is technically and economically challenged by a large variety of mobile OTT applications, different platforms (browser, application based), proprietary codecs/clients, dynamic changes/updates to new versions, various levels of encryption depending on the delivery protocol and platform which extensively, in some cases completely, hinder KPIs accessibility. Therefore, a generic approach for OTT telephony application testing is defined.

The present document describes the design of a white box model emulating OTT telephony application using a generic OTT client, and a public SIP sever or an open-source SIP server in the cloud. The client is developed based on most commonly used codec/client technology (standardized, open source) which enables on device, fully controllable testing of a single OTT version using a fully accessible KPIs set. In addition, the present document describes a framework for data driven validation of such a generic solution against native OTT applications. The server design is not in the scope of the present document.

The design presented in the present document aims to represent a template for future similar OTT applications' testing scenarios, within which either the most commonly used codec/client technology changes and/or evolves, and/or a new OTT voice application emerges as the most commonly used on mobile networks.

Introduction

Starting already with 4G, and now with the evolution of 5G and beyond, as well as with the variety of devices supporting these technologies, a rapidly growing number of new 4G/5G services and OTT applications emerges. The variety and diversity of these services which rely on a continuously evolving sophistication correlated with the increased complexity of the mobile access technologies are raising significant technical and cost challenges. In addition, users expect all these new services and OTT applications to perform as flawlessly as everything else they do on their device. To deliver on this expectation of seamless user experience becomes a significant challenge for operators due to the increased complexity of the network, encryption of applications, proprietary codecs/clients/delivery protocols corroborated with the need to minimize network operational costs.

Therefore, it emerges the need of designing generic OTT voice application and client, which can enable cost-effective network focused performance evaluation, root cause detection, analysis, and optimization without the request to test a large variety of complex, proprietary and fully encrypted native OTT voice applications. Additionally, such generic OTT application provides operators with continuous monitoring and benchmarking to ensure that the performance requirements for these applications are met. In this way, operators spend their efforts on what they can fix and optimize (the network) rather than on the OTT application which they cannot control. For the purpose of the present document, the server is considered to be public or an open source in a cloud; and it is not in the scope of the present document.

1 Scope

The present document presents the design of a white box model of native OTT voice application using a generic OTT voice application which uses a generic client developed based on commonly used codec/client technology (standardized, open source) with the scope to enable on device, fully controllable testing of a single OTT version using a fully accessible KPIs set.

The present document presents the benefits and limitations of such a generic approach, describes how the generic OTT voice application and client are designed, and discusses examples of codec and client adaptation and settings. Description of a framework for a data driven validation of the generic application vs. native OTT application is also provided. Using as example a commonly used mobile native OTT telephony application, the performance results of a generic OTT voice application in terms of quality (MOS scoring), speech path delay, call set up time, call set up failure and drop call are discussed.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] Recommendation ITU-T P.565: "Framework for creation and performance testing of machine learning based models for the assessment of transmission network impact on speech quality for mobile packet-switched voice service".

[i.2] Recommendation ITU-T P.565.1: "Machine learning model for the assessment of transmission network impact on speech quality for mobile packet-switched voice services".

[i.3] Recommendation ITU-T P.863: "Perceptual objective listening quality prediction".

[i.4] Recommendation ITU-T G.114: "One-way transmission time".

[i.5] IETF RFC 3550: "RTP: A transport Protocol for Real-Time Applications".

[i.6] Opus Interactive Audio Codec.

[i.7] PJSIP client, PJSIP - Open Source SIP, Media, and NAT Traversal Library.

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

generic OTT telephony application: OTT telephony application which uses a generic OTT voice client

generic OTT voice client: voice client implemented on a mobile device and using a jitter buffer designed with an error concealment mechanism which emulates a commonly used native OTT telephony application

jitter buffer: buffers used to counter jitter introduced by queuing in packet switched networks to ensure continuous playout of an audio or video media stream transmitted over the network

mobile generic OTT telephony application: OTT telephony application which uses a generic OTT voice client and runs over mobile access

mobile native OTT telephony application: native OTT telephony application running over mobile access

native OTT telephony application: real live OTT telephony application (provided by an OTT vendor) running on mobile devices using an OTT application specific client

OTT client: piece of computer hardware or software that accesses over fixed or mobile network an application made available by a server part of the client-server model, and is implemented on the device consuming the application (such as smart phones and tablets)

OTT voice client: client used to access an OTT telephony application

Over The Top (OTT) application: non-carrier media application offered directly to viewers via website on personal computers, as well as via apps on mobile devices (such as smart phones and tables), digital media players or televisions with integrated smart TV platforms

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

4G	4th Generation Network
5G	5th Generation Network
CG	Call Generator
CS	Circuit Switch
CST	Call Setup Time
DTX	Discontinuous Transmission
e2e, E2E	End-to-End
FER	Frame Erasure Rate
Generic OTT	Generic OTT application/client
ICE	Interactivity Connectivity Establishment
IP	Internet Protocol
MOS	Mean Opinion Score
M2M	Mobile to Mobile
MT	Mobile Terminating
M2F	Mobile to Fix
ms	milli-second
NAT	Network Address Translation
OS	Operating System
OPUS	Open- source voice codec
OTT	Over The Top
PJSIP	Open sources SIP VoIP client, called PJSIP
POLQA	Perceptual Objective Listening Quality Assessment
QCI	Quality Class Indicator
QoE	Quality of Experience
RTP	Real Time Protocol
RTT	Round Trip Time
RLC	Radio Link Control
SPD	Speech Path Delay
KPI	Key Performance Indicator

SIP	Session Interruption Protocol
SILK	Open-source VoIP codec, called SILK
SQ	Speech Quality
SSP	Speech Signal Processing
STUN	Simple Traversal of UDP over NAT
VAD	Voice Activity Detection
VoIP	Voice over Internet Protocol
VoNR	Voice over New Radio (NR)
VoLTE	Voice over Long Term Evolution (LTE)
UACK	Un-Acknowledged
WA	Native OTT voice application, WhatsApp^{TM}

4 Characteristics of a generic OTT voice application

4.1 Features

OTT voice application *monitoring and benchmarking* increasingly grows into a significant multidimension challenge for operators due to the large number of mobile OTT voice applications, with worldwide as well as regionally specific varieties, the multiple different platforms and device-based Operating Systems (OSs). In addition, the native OTT applications are generally using non-standardized device-based audio path settings and internal audio cards causing device biased voice quality scores. New versions of native OTT applications are frequently released without any information if the voice application changed.

Troubleshooting voice quality problems on OTT applications is difficult due to their non-transparency and sometimes use of proprietary codecs and clients, and error concealment schemes, as well as the level of encryption of the signalling and transport data. The latter might be the most challenging aspect due to its device OS dependency and its continuous and dynamic changes.

Last, but not least, performing automatic OTT call tests can be against the user agreement resulting in blocked accounts.

Therefore, troubleshooting, monitoring and benchmarking on even a subset of all the possible combinations becomes technically and costly overwhelming.

On the other hand, operators and regulators are mainly concerned by the network's impact on the voice application quality. The influence of a specific device type and a specific OTT application is of lesser importance, since device vendors, and respectively OTT providers, need to ensure the performance of their own products.

This leads to the introduction of a generic OTT voice application using a generic OTT voice client, which represents a network centric benchmark for OTT voice testing that has three main features to compensate for the challenges mentioned above:

1) It is independent from device specific implementation of native OTT voice clients and focuses on core functionality as codec and jitter buffer.

2) It has a stable version without unknown changes, which usually happens with native OTT voice applications.

3) It has a transparent client which exposes the reason(s) for all types of audio degradations, which is not possible to achieve with a native OTT voice client due to encryption.

These features ensure OTT voice applications testing while compensating for the challenges raised by native applications which are not designed to be part of a testing system, i.e. do not have an Application Programming Interface (API) supporting the type of control and extraction of data which is typically required by a testing system.

4.2 Limitations

The benefits of using a generic OTT voice application testing approach, as described above, should be understood within the context of its limitations, which emerged from the fact that clear differences exist between the generic and the native OTT voice applications. There are several reasons for these differences as listed below, but not limited to these:

a) Generic OTT voice can use a public or private SIP server, in different location, and it might support peer to peer calling, likely generating higher jitter and affecting the voice quality.

b) The generic OTT voice client focuses on core elements of a OTT voice client as codec and jitter buffer and does not emulate potential native specific and/or device depending audio processing. Such Speech Signal Processing (SSP) is not necessary negligible, but is too specific and difficult to estimate and account accurately for within the context of a generic testing approach. For example, SSP introduced by the audio system of the device depends on OS/system version, and differences between manufacturers may exist. Another example is the SSP applied by the OTT VoIP application which might be unlikely for some device (e.g. PC/notebook/tablets) based platforms, but possible on others.

c) Procedure, as well as its type (manual or automated), of setting up generic and native voice calls is likely slightly different since the generic OTT app can be fully controlled while the native one cannot (due to various level of encryptions); this can also differ among different native applications. This effect likely impacts the call set up time.

d) Native OTT voice application could have different profiles settings than the generic OTT. Although these settings should be considered when designing the generic OTT, in some cases a full compliance of these profiles settings with the native OTT is not feasible. In addition, different native OTT voice application can show significantly different profiles settings.

e) Native OTT voice applications use dedicated server infrastructure. These servers might be placed at different locations than public servers. Furthermore, load balancing and routing of the servers provided by the native servers are different. The generic OTT client/server cannot consider or emulate this native server infrastructure. Using the described generic approach for troubleshooting is widely restricted to the radio network but not on core or internet connectivity.

Some of these profiles' settings can be the following, but not limited to:

- generic could drop calls faster than native OTT voice;
- generic could give up call attempt quicker than native OTT voice;
- native can reconnect faster after a drop and/or failed call attempt than generic OTT voice.

f) Native OTT codec rates, bandwidth, and other internal settings can be different than generic OTT voice and consequently likely to show different quality scores.

Therefore, the resulted limitations of the generic approach are as follows:

i) Does not reflect the user experience with a specific native OTT voice application per se, but rather evaluates and can be used to troubleshoot the network performance with the OTT voice telephony.

ii) Its design is based on a most commonly used native application, which uses open- source codec/client and therefore it should be noted that other native applications can show slightly different behaviors and/or performances.

Significant changes of the native OTT voice application, which has been used for defining the generic OTT voice application can occur in time and these need to be periodically monitored and evaluated to decide on the need for adjustments of the generic solution.

5 Design of a generic OTT voice application

5.1 Overview

The design of a generic OTT voice application is described in this clause using an example of a native OTT application, WhatsApp^{TM}.

It should be noted that all the design steps described below are to be followed if another native OTT voice application is selected to create the generic solution. It is expected though that selection of another native OTT application results in slightly different settings of the generic OTT voice application.

To be a relevant benchmark, the generic OTT voice application needs to be designed based on one of the most used OTT voice applications. The example selection of the native OTT application WhatsApp^{TM} has the significant advantage of using an open-source codec (OPUS [i.6]) and a jitter buffer of an open -source VoIP client (PJSIP [i.7]), both with many configurable settings. A thorough analysis of native WhatsApp^{TM} application traffic patterns in different network conditions, ranging from very poor to excellent, need to be used to adapt these settings, thereby making generic OTT mimic the voice quality produced by the most used mobile OTT voice application.

The settings for the codec and jitter buffer have been determined based on their software versions available at the time of the development of generic OTT voice. These settings are already standardized in Recommendations ITU-T P.565 [i.1] and P.565.1 [i.2]. The settings should be kept unchanged in order to make generic OTT compatible to ITU-T, as well as to ensure that generic OTT measurements represent a stable, and consistent benchmark reference for the performance of OTT voice applications.

This clause describes the settings adaptation for the generic codec/client using the example of the native WhatsApp^{TM} application.

5.2 Codec settings

To adapt the WhatsApp^{TM} OPUS codec settings, the RTP packets, the audio recordings and the voice quality at good radio quality need to be analysed. The settings used and the reason for them are detailed in Table 1.

Table 1: Codec settings and the reason for the settings

Codec settings		Reason
Codec	Opus (vs.1.3)	Inspection of the WhatsApp^{TM} application shows Opus.
Frame time	20 ms	There are gaps of 20 ms in the live speech and the default frame time is 20 ms.
Frames per packet	3	RTP stream shows 60 ms between each sent packet, except at the beginning of the call, where there is a longer interval of 160 ms. As a generic behaviour this parameter is set to 60 ms.
Sample rate	16 000	Bandwidth of recorded speech is 8 000 Hz and SILK^{TM} audio codec default sample rate is 16 000 Hz.
DTX/Comfort noise/VAD	No	RTP stream shows 60 ms between the smallest packets, which are large, so no DTX and therefore no Voice Activity Detection (VAD) or comfort noise is needed.
Variable bit rate	Yes	RTP stream shows varying payload sizes for every packet.
Bit rate	25 000 bit/s	Shows the best fit between encoded size and IP capture. This is also the default for the SILK^{TM} codec inside Opus.
Packet loss expected by encoder	0 %	The comparison of the best live speech quality to an encoded or decoded reference shows that the best fit are the settings of 0 % expected packet loss and complexity 10.
Packet loss concealment	Use Opus	Using Opus own packet loss concealment seems reasonable instead of using that of PJSIP.
Perceptual enhancement	0	PJSIP (VoIP client) default setting.
Complexity	10	The comparison of the best live speech quality to an encoded or decoded reference shows that the best fit are the settings of 0 % expected packet loss and complexity 10.
All other		PJSIP (VoIP client) default setting.

5.3 Jitter buffer settings

To adapt the WhatsApp™ PJSIP jitter buffer settings, several hundred tests comparing generic OTT and native WhatsApp™, both in good and bad radio quality need to be performed. To ensure accuracy, the comparisons need to be made by simultaneously on the same device running both PJSIP and WhatsApp™ and evaluating voice quality (MOS, Recommendation ITU-T P.863 [i.3]).

To determine the codec and jitter buffer settings, speech and its correspondent RTP stream need to be examined, both at good and poor radio quality. Figure 1, presenting an example of speech capture during poor radio quality, shows that in this case speech is in chunks with large gaps in between; thus, prefetching is used by the jitter buffer. The normal length of WhatsApp™ packets is 60 ms. However, it was observed in the recorded audio that losses of 20 ms packets can occur. These packet losses indicate discards used to shrink the jitter buffer, thus implying its adaptive behaviour.

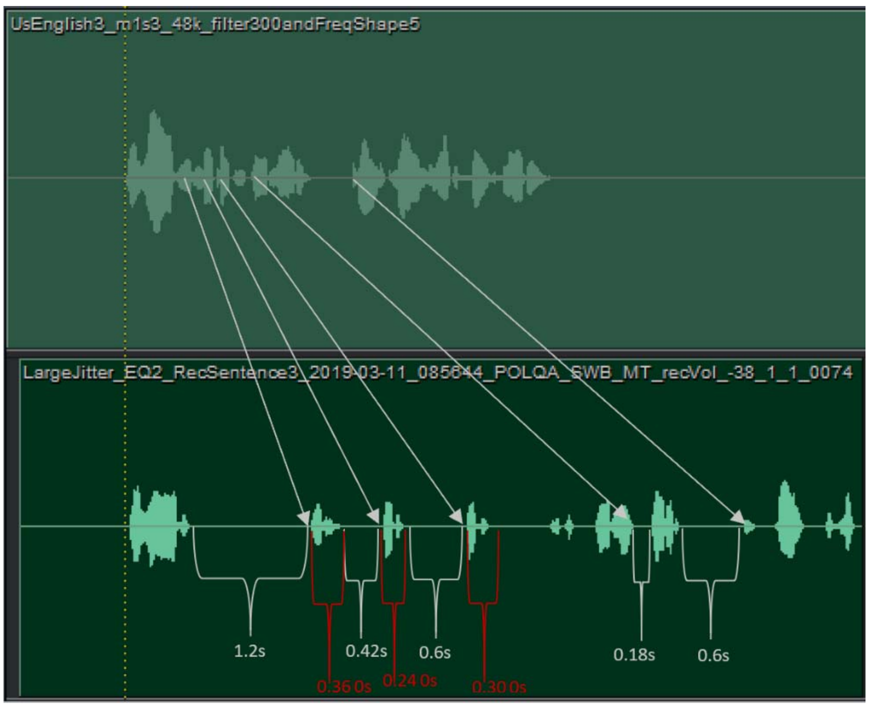

UsEnglish3_m1s3_48k_filter300andFreqShape5
LargeJitter_EQ2_RecSentence3_2019-03-11_085644_POLQA_SWB_MT_recVol_-38_1_1_0074
1.2s
0.42s
0.6s
0.18s
0.6s
0.36 0s
0.24 0s
0.30 0s

Figure 1: WhatsApp™ voice capture

This analysis of the jitter buffer behaviour, monitoring the Recommendation ITU-T P.863 [i.3] MOS results obtained from the two applications and optimizing the generic towards the native OTT scores (Figure 2) are used to determine the jitter buffer setting (Table 2).

To further enhance the accuracy, digital audio (instead of analogue audio) should be used when measuring WhatsApp™.

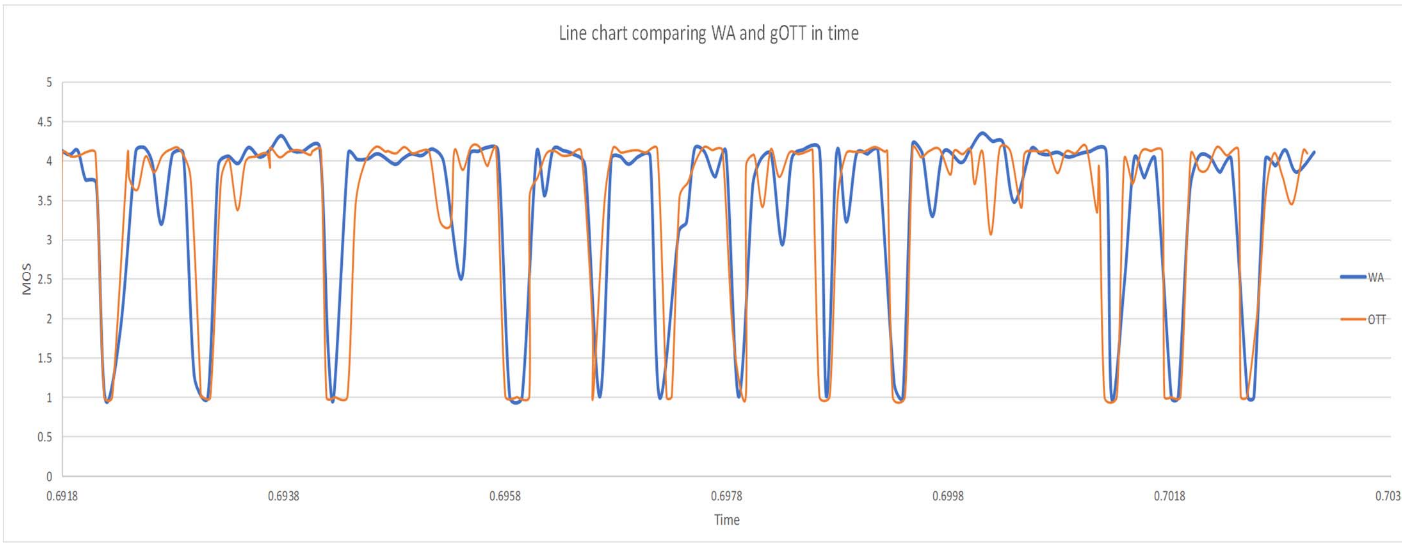


Figure 2: Time view comparing voice quality scoring (MOS) for generic OTT and WhatsApp^{TM}

Figure 2 shows the settings chosen for the jitter buffer and the reasoning behind them.

Table 2: Client jitter buffer settings and the reason for the settings

Jitter buffer settings		Reason
Jitter buffer	PJSIP (vs. 2.8)	Inspection of the WhatsApp^{TM} application shows PJSIP is the most recent.
Adaptive	Yes	Speech delay varies and there are 20 ms gaps in the audio, which fit with the use of discards to adapt the buffer size.
Packet size	60 ms	See "Frames per packet" (body row 2 of Table 1) plus the length of prefetches, observed in the speech, is always a multiple of 60 ms (see Figure 1, the numbers in red).
Jitter buffer max size	1 500 ms	To accommodate the largest observed speech path delay of 3 000 ms, equalling two buffers of 1 500 ms, while also keeping the recording interval below twice that of the audio reference length.
Jitter buffer initial size	0 ms	Same as the prefetch value, see next row.
Jitter buffer minimum prefetch	0 ms	A prefetching minimum of 0 allows the jitter buffer adaptation to control the minimum prefetch; the zero value setting showed that adaptation does not perform unnecessary discards and reflects what has been noticed with real-field WhatsApp^{TM}.
Jitter buffer maximum prefetch	4/5 times jitter buffer maximum size	PJSIP default setting.
All other		PJSIP default settings.

6 Framework for data driven validation of generic vs. native OTT voice testing

6.1 Overview

The generic approach delivers trustful results as a representative benchmark only if it closely mimics a native application. Therefore, the generic application should meet two main conditions:

1) To trigger the same network resources, use the same protocols, and encode and adapt as real applications and OTT codecs and clients would.

2) To show similar performance with the native OTT voice (WhatsApp™ in this case) selected for training and tuning the generic application. By performance is meant the following: voice quality (MOS, Recommendation ITU-T P.863 [i.3]), Speech Path Delay (SPD), call set-up time, drop and blocked call rate.

Therefore, the validation of the generic vs. native OTT voice (WhatsApp™ in this case) needs to consider the following main requirements:

- Req.1: run simultaneously generic and native OTT voice.
- Req.2: closely synchronize the call start of the two applications.
- Req.3: use digital instead of analogue audio to ensure increased accuracy when measuring native OTT app.
- Req.4: run the two applications in a broad range of network conditions, covering uniformly the whole quality range, from very good to very bad. Likely, the very bad conditions need to be created with dampers in radio shielded room or network emulators, since in driving long time in bad live network conditions is not straightforward achievable.

In this clause is described the validation process and expected results based on the native OTT voice application WhatsApp™ .

It should be noted that the validation process has to be followed as described in this clause, if another native OTT voice application is used as example. However, it is expected that the performance results to be slightly different.

6.2 Validation of generic vs. native OTT voice quality (MOS)

It is recommended to collect during several drive tests a large and comprehensive data set covering various network conditions and their correspondent voice quality scores. The test set-up needs to follow Req.1 - Req.4 mentioned above.

The analysis of the data should cover:

- Statistics (average, minimum, maximum, stdev) for generic and native OTT voice MOS scores.
- Generic and native OTT voice MOS scores' distributions.

An example is presented below. The compared OTT voice applications are the generic one as defined above, and the native is the WhatsApp™ app based on which the generic has been trained and tuned. The test network conditions (continuously variable live LTE network access) and test set-up are in conformance with requirements mentioned above.

Table 3 shows that the two OTT applications exhibit no statistically significant difference between the performance metrics (average, standard deviation, maximum, minimum).

Figure 3 shows that the distribution of the generic and native/ WhatsApp™ MOS scores for the given example exhibit statistically close MOS scores distribution. It should be noted that to perform a meaningful comparison a large enough bin size needs to be carefully selected, preferable higher than 0,5 MOS which generally represents human perception resolution.

Table 3: Comparison between generic OTT and WhatsApp™ (WA) MOS in LTE - varying radio conditions

Recommendation ITU-T P.863 [MOS]	Average	Standard	Maximum	Minimum
WhatsApp™	4,24	0,60	4,54	1,04
generic OTT	4,22	0,62	4,55	1,23

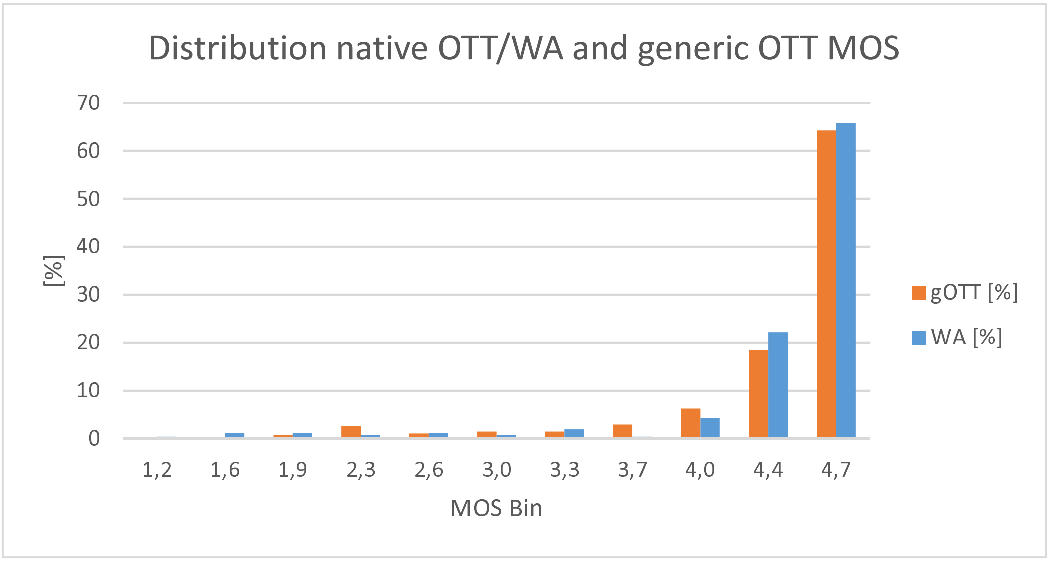


Figure 3: Comparison between generic and native (WhatsApp^{TM}, WA) MOS

6.3 Validation of generic vs. native OTT voice Speech Path Delay (SPD)

6.3.1 Overview

In voice conversations, long delays may cause callers to start talking at the same time and thereby interrupt each other. It can become hard to maintain a meaningful conversation. Therefore, Speech Path Delay (SPD), defined as Round Trip Time (RTT), is another voice quality metric, which generic and native OTT applications have to show close values for, in order for generic OTT app to be validated.

This clause discusses how SPD can be determined for generic OTT voice and how can this be validated against native OTT app.

6.3.2 Defining an SPD baseline

OTT voice speech path delay can easily get larger than for carrier voice (VoLTE, VoNR), mainly due to network resources scheduling for OTT applications. A baseline on how user experience is impacted is presented in Table 4 based on Recommendation ITU-T G.114 [i.4] (one-way delay has been translated into round-trip delay).

Table 4: SPD values impacting the users based on Recommendation ITU-T G.114

Speech Path Delay (Round-Trip Time) [ms]	Experienced Quality
< 400	Users very satisfied
400 ... 600	Users satisfied
600 ... 780	Some users dissatisfied
780 ... 1 100	Many users dissatisfied

6.3.3 SPD calculation for generic OTT voice

As an example, an algorithm for calculating SPD can be based on the ability to synchronize on speech, making possible to play the speech at an exact time and detect when is recorded. SPD can be then calculated as the time between the end of the sending audio and the receiving audio (Figure 4). SPD can be calculated on the Mobile Terminating (MT) side and can be provided for every play/record cycle. This means the SPD is reported about for example every 11,5 seconds (in the case of test speech samples of about 5,5 seconds) from the MT side, except for the first recording in a call, at which time the real time synchronization has not yet reached enough accuracy. If the MOS score drops below 2,5 MOS, the SPD measurement is recommended not to be considered reliable, and no SPD should be reported.

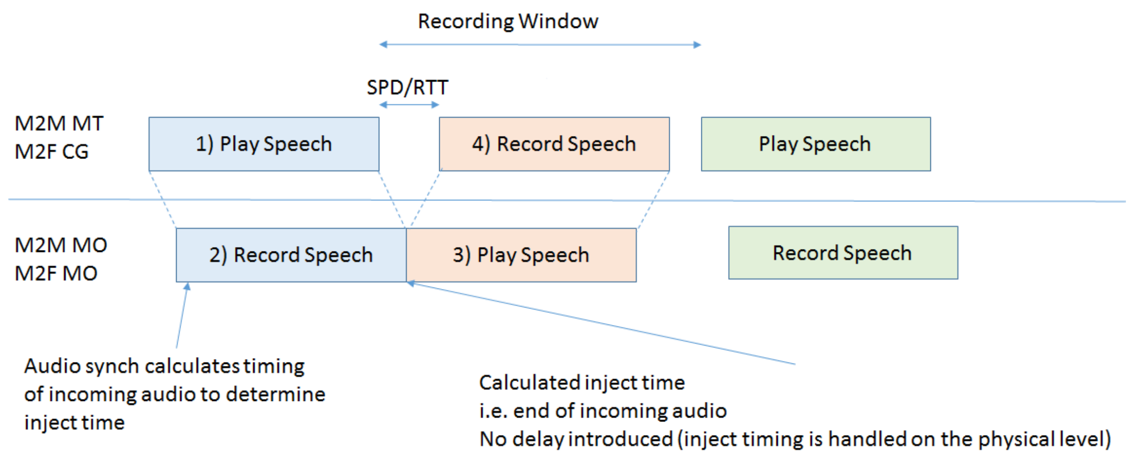

Recording Window
SPD/RTT
M2M MT
M2F CG
1) Play Speech
4) Record Speech
Play Speech
M2M MO
M2F MO
2) Record Speech
3) Play Speech
Record Speech
Audio synch calculates timing of incoming audio to determine inject time
Calculated inject time
i.e. end of incoming audio
No delay introduced (inject timing is handled on the physical level)

Figure 4: Example for the time sequence for SPD calculation

6.3.4 Understanding the delay in OTT voice

The reason for delays in OTT voice applications is complex and very important since it can easily cause long delays, which fall into "many users dissatisfied" quality category. The variation of the delay (jitter), is also a problem since jitter will cause packets to arrive too late or trigger jitter buffer adaptions, therefore impacting the voice quality as well. Delays in OTT voice testing are caused mainly by speech coding, transport network, radio network, jitter buffers and devices. Unlike native OTT voice applications, generic OTT voice makes it possible to investigate the impact from each of these in detail, and consequently enabling the validation of the Speech Path Delay (SPD, Round Trip Time) metric against native OTT SPD.

a) Speech coding delays:

Generic OTT voice app uses OPUS codec which has a small 6,5 ms look ahead and resampling time. The audio packets contain three 20 ms audio frames which adds 60 ms for a total delay of 66,5 ms. The SPD measurement considers the speech coding delay at the transmitting device therefore adding a total of 66,5 ms to the SPD.

b) Radio network:

The typical lowest radio network delay for both VoLTE and LTE data network is 40 ms. The typical delays caused by VoLTE are less varying than the delays experienced by an OTT application using the LTE data network. The reason is that VoLTE uses the highest priority dedicated bearer (QCI = 1) and consequently benefits of optimized radio resource scheduling. In addition, as a real time application, VoLTE runs in UACK (Un-Acknowledged) mode, which is not using RLC retransmissions, and consequently packets are discarded leading to packet loss. On the other hand, OTT voice uses LTE data with RLC retransmissions generating larger jitter and jitter bursts, which are short peaks of high time delay. When the radio becomes so bad that no packets can be transmitted, the delay increases until the radio stabilizes again, and all of the buffered packets can be released and send to the receiving device. Such delay bursts can be up to several seconds and afterwards they will have a residual effect due to jitter buffer adaption. Figure 5 shows how the radio quality varies in an on/off manner and that the SPD follows this radio network behaviour.

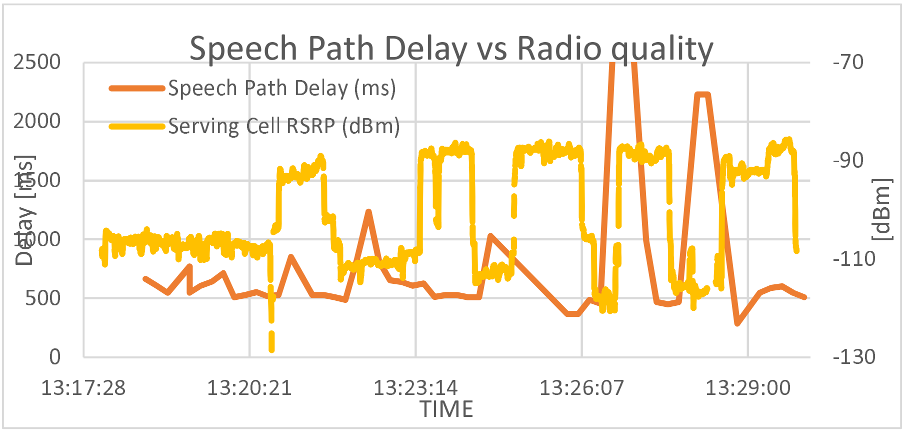


Figure 5: Bad radio quality increases the SPD

c) Jitter buffer:

The jitter buffer aims at handling the network jitter while keeping the delay down. The jitter buffer delay is closely connected to the jitter in the radio network. For LTE data networks the jitter is typically at least two packets, which correspond to 120 ms. This delay exists on both devices, so that the round-trip delay caused is at least 240 ms. To handle the large jitter bursts, which the radio network can produce, the jitter buffer handling of the generic OTT app allows the buffer to fill up to its maximum of 1 500 ms, after which all later packets are discarded. After the burst has ended, the network jitter goes back to normal, and the jitter buffer adapts its delay down to the network jitter by discarding packets. This process can take more than 10 seconds, depending on the size of the delay burst, which means that the delay from the network which triggered the delay burst, will continue to affect the overall delay during more than 10 seconds, thus giving a significant impact on average SPD. As discussed in the Radio Network clause, the jitter bursts are due to radio quality issues so an operator can significantly diminish the SPD by improving the radio network. To help with this, it is recommended that the jitter buffer delay in generic OTT voice to be reported as an information element (for example "VOIP Jitter Buffer Playout Delay Average"). Figure 6 shows how the jitter buffer grows and shrinks when jitter bursts are occurring, and how the SPD is affected.

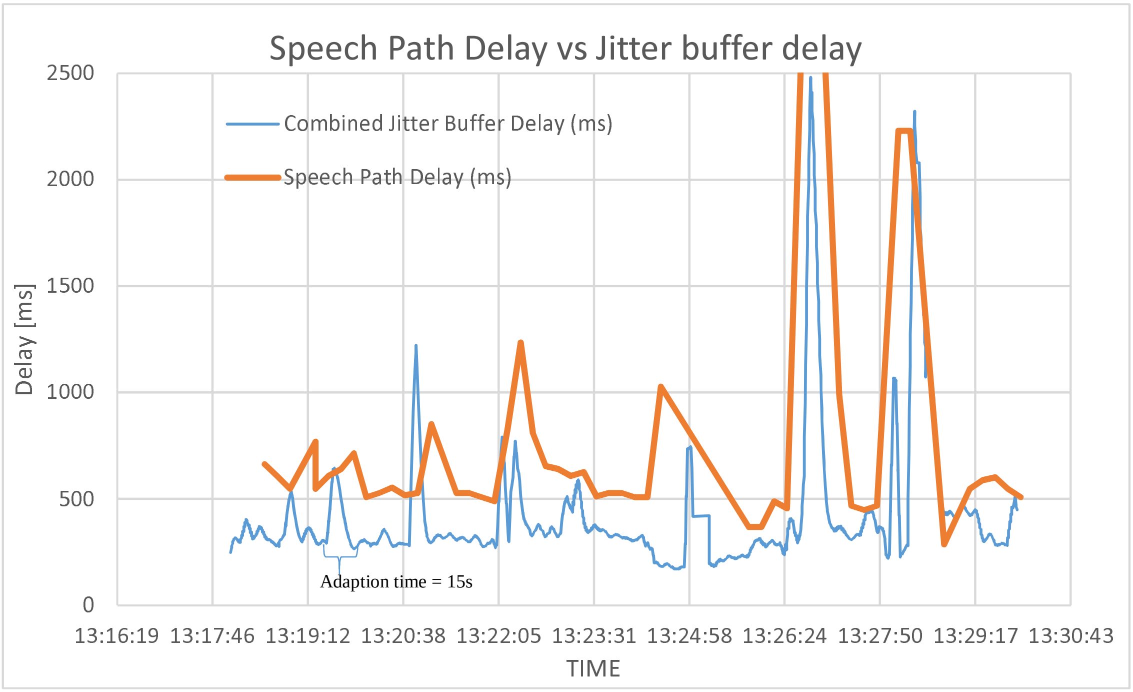


Figure 6: SPD vs. combined jitter buffer delay (sum of buffer delay from both devices)

d) Transport network:

The amount of time spent in the transport network depends on the location of the devices calling each other and the network path needed to resolve Network Address Translation (NAT) traversal issues.

If the devices are connected to the same physical base station, the transport time can be almost zero, which is the best case. However, even when the devices are on the same base station, there can still be transport delays due to Network Address Translation (NAT) traversal issues. In the ideal case the devices communicate directly (peer to peer) but in some cases the RTP traffic needs to be looped via a server to accomplish the NAT traversal. The server can be a server in the operator's network or a server on the Internet, for example with the same IP address as the SIP server. To find out if the traffic is going directly, via a server or via the SIP server, it is required to compare the IP addresses of the RTP messages on both devices and the address of SIP messages towards the SIP server. To find the delay caused by communicating via a server, it is recommended to ping it. A typical round-trip delay added by a NAT traversal server can be around 50 to 100 ms.

e) Device:

Acoustic measurements show a typical value of about 300 ms for the delay caused by the audio path of the device, comprising microphone, loudspeaker and associated buffering within the operating system (Android^{TM}). These parts are highly dependent on the device and their implementation in the device, and consequently not bearing any information on the network's impact on SPD. In order to keep the measurements network centric, the delay coming from the device is not recommended to be considered in the SPD measurement.

6.3.5 Validation of generic OTT voice SPD measurements

a) Validation of average SPD:

The SPD is network centric so it cannot contain the device specific delay caused by the audio path. Therefore, the validation of the correctness of the *average* SPD cannot be performed by directly comparing to the average SPD for WhatsApp^{TM} since this will contain the device specific audio path delay. Instead, it is recommended to compare the average SPD with the sum of all delays making up the network centric delay for different conditions. All the delays specified above, except the Jitter buffer delay, can be approximated with constants during a drive test. Therefore, a delay budget can be created based on approximately constant delays and the measured jitter buffer playout delay. An example is presented in Table 5. Table 5 shows that the delay budget adds up nicely to the measured generic OTT SPD in both good and bad radio cases. This example proves the following:

- the average/median SPD is a reliable measure of the delays making up the network centric delays;
- the jitter buffer playout delay is directly affected by the radio quality, and it is a very important parameter to focus on;
- the generic jitter buffer is well designed.

Table 5: Delay budget vs. generic OTT average SPD

Delay budget	Operator1 good radio direct [ms]	Operator2 good radio via server [ms]	Operator2 bad radio via server [ms]
Coding	67	67	67
Radio network	40	40	> 40
Average Jitter buffer delay	230	270	420
Transport	0	60	60
Sum of delay budget	337	437	587
Measured average SPD	348	459	596
Difference	12	23	10

b) Validation of SPD distribution:

SPD measurements are performed with the same requirements as for the MOS measurements described in clause 6.2. Comparing the distribution of SPD between WhatsApp^{TM} and generic OTT demands a calibration to remove the audio path influence. Figure 7 shows an example of a SPD distribution for WhatsApp^{TM} and generic OTT using a median calibration on the generic OTT. The distributions are very similar showing that generic OTT and WhatsApp^{TM} are reacting in a similar way to varying network conditions.

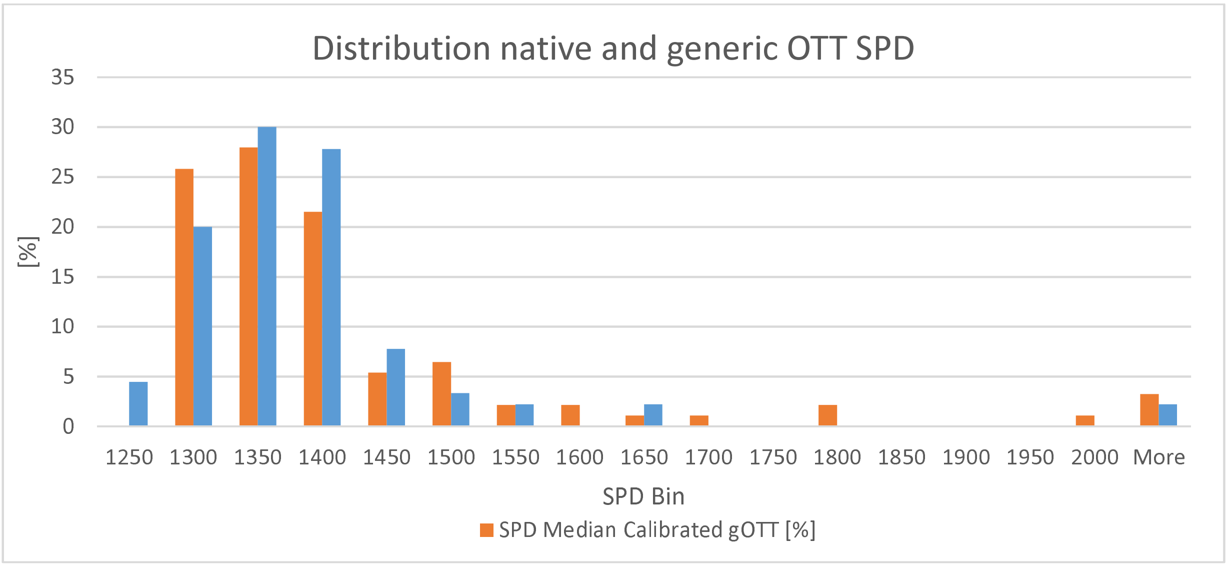


Figure 7: Generic and native OTT SPD distributions with median calibration

6.3.6 Validation of generic vs. native OTT voice call set up time

As mentioned in clause 4.2, the call setup time on native applications is hard to measure reliably. The reason being the encrypted SIP signalling and the delays added by automated dialling and answering. To avoid these problems, it is needed to measure the call setup manually on the native application, and only to include the well-defined part of dialling to ringing. Specifically, ringing on the native app was specified as the appearance of the answering button on the screen. Call set up is easy to be measured on the generic OTT app since the SIP signalling is shown in decrypted form. To measure as close as possible to the native app SIP Invite on mobile origination side to Ringing on mobile terminating side can be used. Without being able to perform automatic tests at the exact same time on native and generic applications, due to the manual measurement needed on the native app, using a controlled and stable degradation is necessary. To perform the degradation, a throttle app or a network emulator can be used. The test presented here uses the throttle app "Throttly^{TM}" which added delay on top of the normal LTE delay. For each latency setting the app randomizes the latency around the setting so a number of measurements need to be made and averaged for each setting. Table 6 and Figure 8, below, show an example of the call setup delay, for WhatsApp^{TM} and generic OTT, in the same average network conditions. WhatsApp^{TM} shows an offset for the zero latency condition, otherwise both applications follow each other, giving a similar trend for the increase in call setup time.

Table 6: Generic and Native call setup time

Throttly^{TM} delay [ms]	Generic OTT average delay [ms]	WA average CS [ms]
0	868	2 640
2 000	5 022	5 753
4 000	9 726	8 555
6 000	10 868	13 400
8 000	19 430	18 136
10 000	21 564	23 236

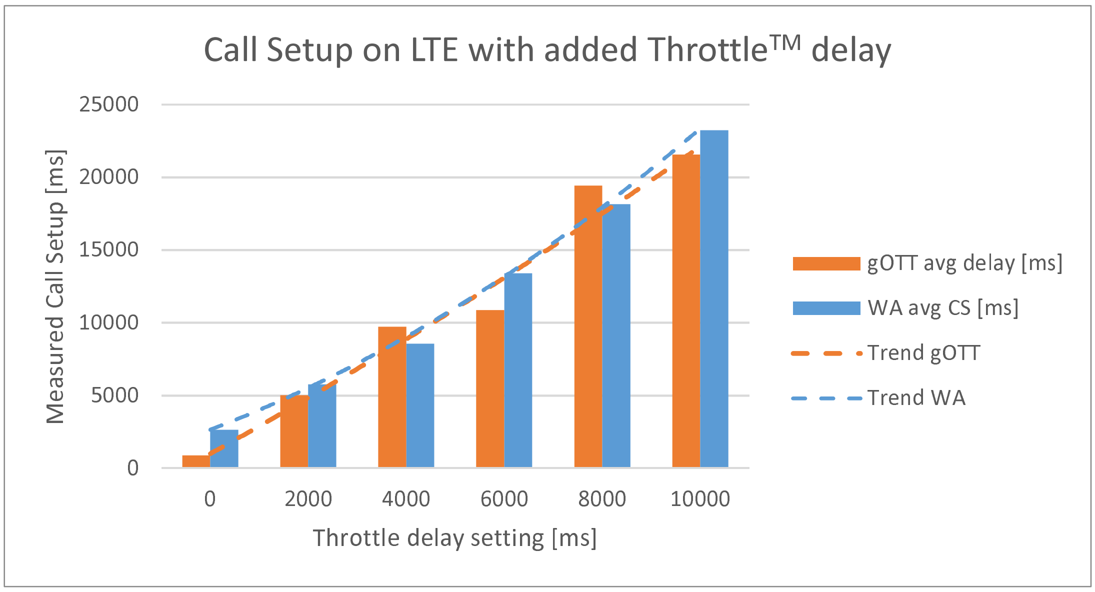


Figure 8: Generic and native OTT Call Setup time

6.3.7 Validation of generic vs. native OTT call setup failure

For the network operator the most important type of call setup failure is the failure due to radio intermission. Radio intermission can be emulated in many ways but using the throttling app, Throttly^{TM}, is an easy solution. Before a call is made the app is set to a throughput of 1 kbit/s on up-link and down-link, effectively shutting down the connection. Call failure on WhatsApp^{TM} is detected by a warning text on the screen. Automatic detection of this takes some time so a manual measurement is preferred. On the generic app a failed call setup will be reported by a session report from the app.

On WhatsApp^{TM} a call setup failure will be detected after a time out of 90 seconds. This is a very long time, and most users will give up waiting after a much shorter time. It is also suboptimal from a network testing perspective since it will limit the possible rate of call setup tests.

On the generic OTT a call setup failure is detected immediately when trying to make a call in a no network condition. From a network testing perspective this is preferrable since it allows call setup tests to be performed at any rate.

6.3.8 Validation of generic vs. native OTT call drop

Call drop due to radio intermission can be tested by shutting down the connection any time after the call setup has succeeded. Using Throttly^{TM} in the same way as in clause above shows WhatsApp^{TM} to have a connection time out of 30 seconds. The generic OTT detects a call drop after 60 seconds. Both 30 and 60 seconds are valid time outs for detecting a call drop.

7 Aspects of a generic OTT voice application implementation

7.1 Generic voice jitter handling and its KPIs

Similarly with native, generic OTT application uses a jitter buffer to handle the varying arrival time of the RTP packets, and it uses RTP packets with three audio frames of 20 ms length. The jitter buffer handles these 20 ms audio frames, instead of 60 ms RTP packets. Since the 20 ms frames are used for audio playout, all audio related problems will be related to missing 20 ms frames at the time of playout. A frame which cannot be played out, when it is needed, is defined as a frame erasure which a generic OTT can for example display as an overall Frame Erasure Rate KPI. This KPI best reflects the correlation with voice quality and can be expressed for example as "VoIP FER combined packet loss (%)". The FER combined comprises several different types of frame erasures due to the internal frame handling by the jitter buffer.

The jitter buffer handling and the corresponding types of frame erasures are discussed further.

Generic OTT uses an adaptive jitter buffer which has two contradicting tasks:

1) Keep buffer large enough to handle the most common jitter delays.

2) Keep the buffer small to have short play out delay.

Both these tasks will affect the voice quality and the effect can be described by the different Frame Erasure Rate KPIs as an example shown in Figure 9.

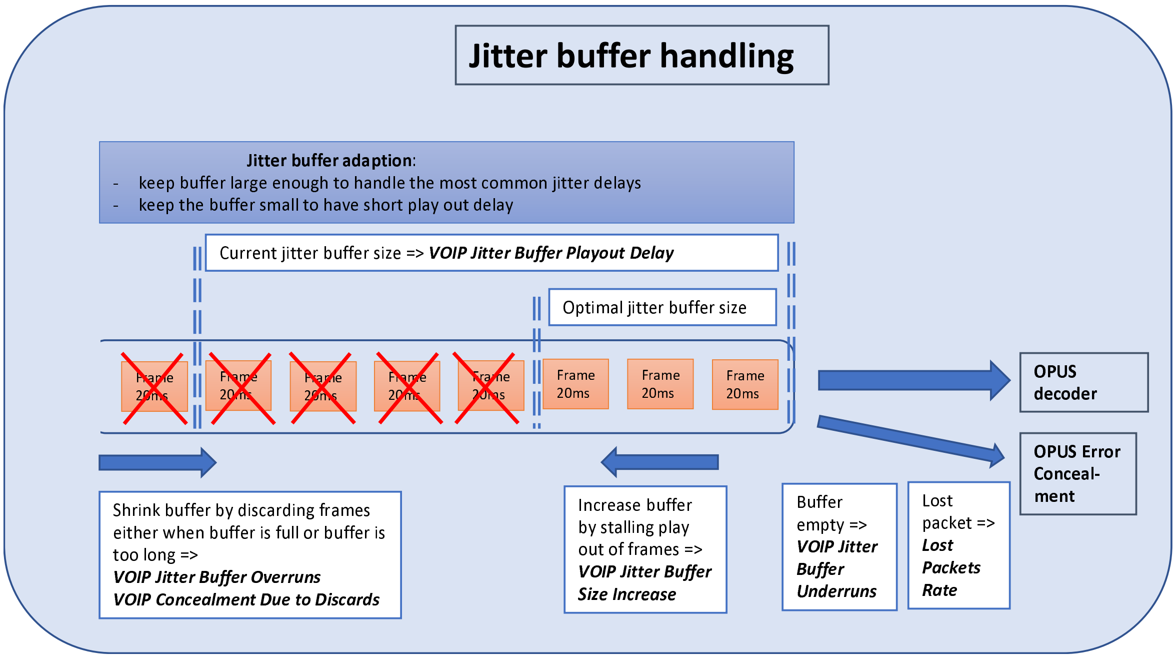

Jitter buffer handling
Jitter buffer adaption:
- keep buffer large enough to handle the most common jitter delays
- keep the buffer small to have short play out delay
Current jitter buffer size => VOIP Jitter Buffer Playout Delay
Optimal jitter buffer size
Frame 20ms
Frame 20ms
Frame 20ms
Frame 20ms
Frame 20ms
Frame 20ms
Frame 20ms
Frame 20ms
OPUS decoder
OPUS Error Conceal-ment
Shrink buffer by discarding frames either when buffer is full or buffer is too long => VOIP Jitter Buffer Overruns VOIP Concealment Due to Discards
Increase buffer by stalling play out of frames => VOIP Jitter Buffer Size Increase
Buffer empty => VOIP Jitter Buffer Underruns
Lost packet => Lost Packets Rate

Figure 9: Example of the generic OTT jitter buffer handling

The first task is performed by increasing the buffer size by stalling play out of frames, so called prefetching. Amount of prefetching is one of the parameters adapted to WhatsApp^{TM} (Table 2). Prefetching is performed at beginning of the call and when the buffer becomes empty. Stalling the play out will trigger error concealment by OPUS and will sound like distorted audio, talking slower or silent periods. A KPI showing the rate of frames being stalled due to increasing the buffer can be used, for example as "**VOIP Jitter Buffer Size Increase**".

The second task is performed by discarding frames either when the buffer is full or the buffer is too long. The aggressiveness of the discards and the maximum buffer size are parameters adapted to WhatsApp^{TM} (Table 2). The first discard in a row of discards will be error concealed while the following discards will only give removed audio frames. Discards will sound like talking faster or distorted audio. A series of KPIs describing the performance due to discards can be defined for example as follows:

- KPI showing the rate of frames being discarded, for example as "**VOIP Jitter Buffer Overruns**".
- KPI showing the rate of frames being error concealed due to discards, for example "**VOIP Concealment Due to Discards**".
- KPI showing play out delay, for example "**VOIP Jitter Buffer Playout Delay average**". This KPI comprises the most important part of the total Speech Path Delay.

Network problems like long jitter delays making the buffer go empty or lost packets will trigger error concealment by OPUS and will sound like distorted audio or silent periods. A KPI showing the rate of frames being error concealed due to empty buffer can be defined, for example "**VOIP Jitter Buffer Underruns".**

In addition, a KPI showing the combined rate of underruns and lost frames at the time of playout can be defined, for example "**RTP Lost Packets Rate Audio**".

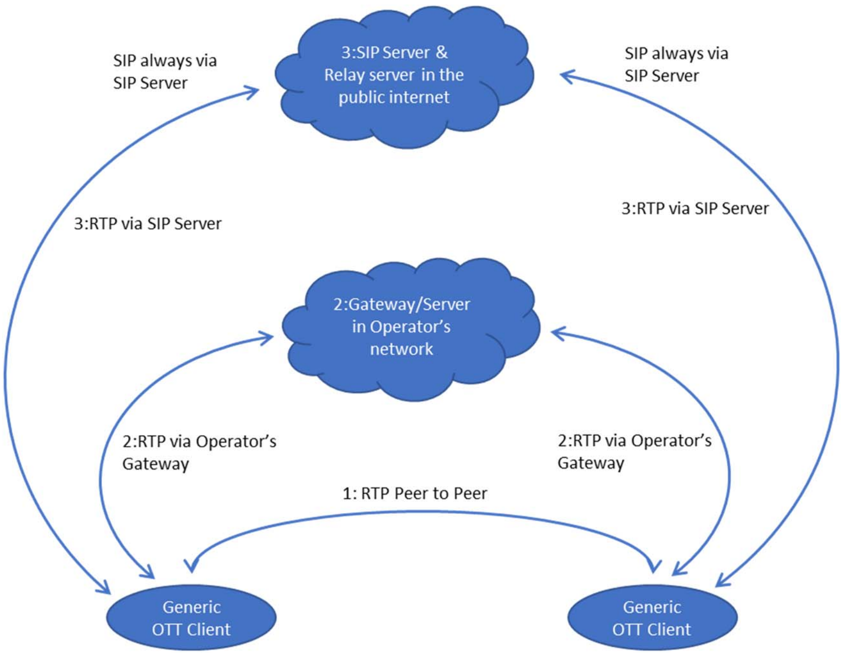
Generic OTT application can report the jitter buffer KPIs once per second. Since the frame time is 20 ms there are usually close to 50 frames per report and consequently a frame erasure rate of 2 % corresponds to one lost frame. Table 7 presents the above discussed KPIs examples and their description.

Table 7: Examples of generic OTT jitter buffer KPIs

Recommended KPIs	Description
VOIP Jitter Buffer Size Increase	Rate of frames being stalled from played out due to increasing the buffer
VOIP Jitter Buffer Overruns	Rate of frames being discarded due to buffer being full or too large
VOIP Concealment Due to Discards	Rate of frames being error concealed due to discards
VOIP Jitter Buffer Underruns	Rate of frames being error concealed due to empty buffer
RTP Lost Packets Rate Audio	Combined rate of underruns and lost packets. Related to frame loss directly due to network issues
VOIP FER Combined Packet Loss	Combined rate of all types of frame loss and discards. Shows very good correlation to the voice quality by including both frame loss due to jitter buffer handling and frame loss directly due to network issues
RTP Jitter Audio I [i.5]	Jitter on the down link RTP stream
VOIP Jitter Buffer Playout Delay Average	The audio delay caused by the jitter buffer. Average during 1 second
VOIP Jitter Buffer Playout Delay Min	The audio delay caused by the jitter buffer. Minimum during 1 second
VOIP Jitter Buffer Playout Delay Max	The audio delay caused by the jitter buffer. Maximum during 1 second

7.2 Understanding the impact of the SIP server

The transport delay of the RTP audio, can be dependent on the SIP Server location. The SIP server will use techniques like ICE and STUN to resolve Network Address Translation (NAT) traversal issues. In most cases this will allow peer to peer communication or relaying via a server in the operator's network. In both these cases the added transport delay will be small, typically an added RTT of 50 to 100 ms. In the worst case, when ICE and STUN fails, the RTP audio will be relayed via the SIP server or another specialized Relay server at the same location, see example in Figure 10.


3:SIP Server & Relay server in the public internet
SIP always via SIP Server
SIP always via SIP Server
3:RTP via SIP Server
3:RTP via SIP Server
2:Gateway/Server in Operator's network
2:RTP via Operator's Gateway
2:RTP via Operator's Gateway
1: RTP Peer to Peer
Generic OTT Client
Generic OTT Client

NOTE: Due to difficult NAT traversal issues, transport can be relayed via the SIP server.

Figure 10: Example of OTT audio transport, usually directly between clients or via gateway in operator's network

If the SIP server is located far away this will add several hundred milli-seconds of delay. The longer delay should add jitter due to more router hops which in turn should cause the jitter buffer to grow and add even more delay. The increased jitter should also increase the Frame Erasures and thereby have a negative effect on the voice quality. Therefore, it is important to measure and understand the effect of the distance to the SIP server/Relay server. This can be done by disabling ICE and STUN to always get relay via the SIP server. Using publicly available SIP servers, or deploying SIP servers on cloud services, makes it possible to test SIP server locations all over the world, see Figure 11 below.

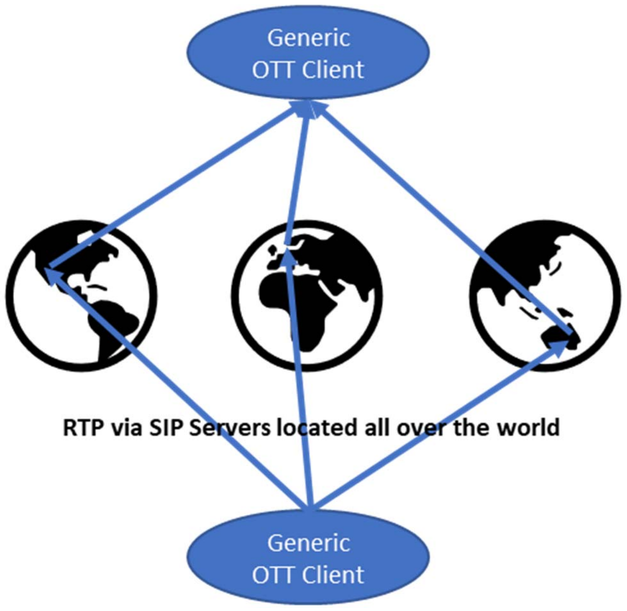


Figure 11: SIP servers located all over the world can be used for testing the impact of delays due to distance

As an example, the measurements performed show the anticipated relationship between server distance and the relevant KPI's (Table 8, Figure 12 and Figure 13):

- Speech Quality (SQ) is slightly degraded.
- Frame Erasure Rate (FER) is slightly increased.
- Speech Path Delay (SPD) is proportional to distance.
- Call Setup Time (CST) is proportional to distance.

The measurements also show that SQ and FER are immune to all but the most extreme distances, in this case it is only the route from Skelleftea, Sweden to Sydney, Australia which shows an impact. The SPD and CS are impacted also by the route to California. This leads to a recommendation that the server should be placed at least in the same continent so the server distance results in a minimal impact and emulates well a native OTT application which can be anticipated to have servers distributed all over the world or at least where their main user base is located. Still tests need to be performed to verify that the used server has a minimal impact on the results regarding SQ, FER, SPD and CS time.

Table 8: Example of SQ, FER, SPD and CST for various SIP server locations with calls placed in Skelleftea, Sweden on LTE

SIP server location	SQ [MOS]	FER [%]	SPD [ms]	Call Setup time (200 ok) [ms]
Sydney	4,18	1,41	1 011	2 096
California	4,45	0,59	726	1 239
Netherlands	4,35	0,68	442	928
Stockholm	4,43	0,54	410	802
Swedish Operator Server	4,41	0,52	422	903

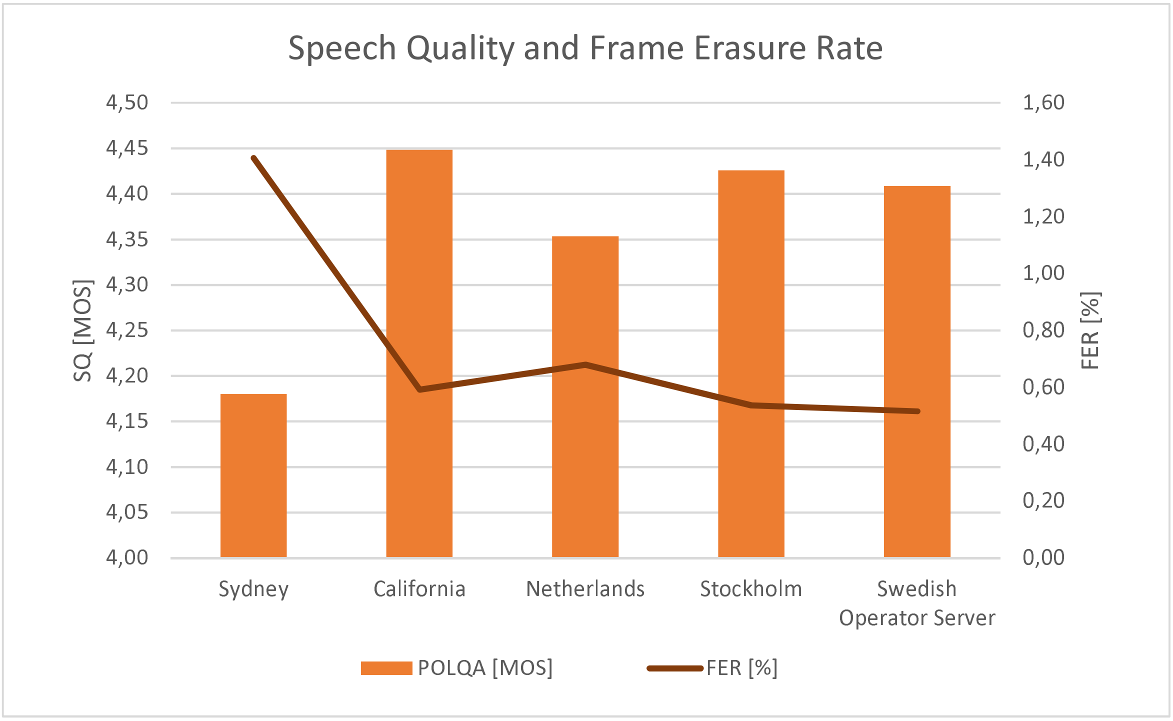


Figure 12: Example of Speech Quality and FER dependency on SIP server location

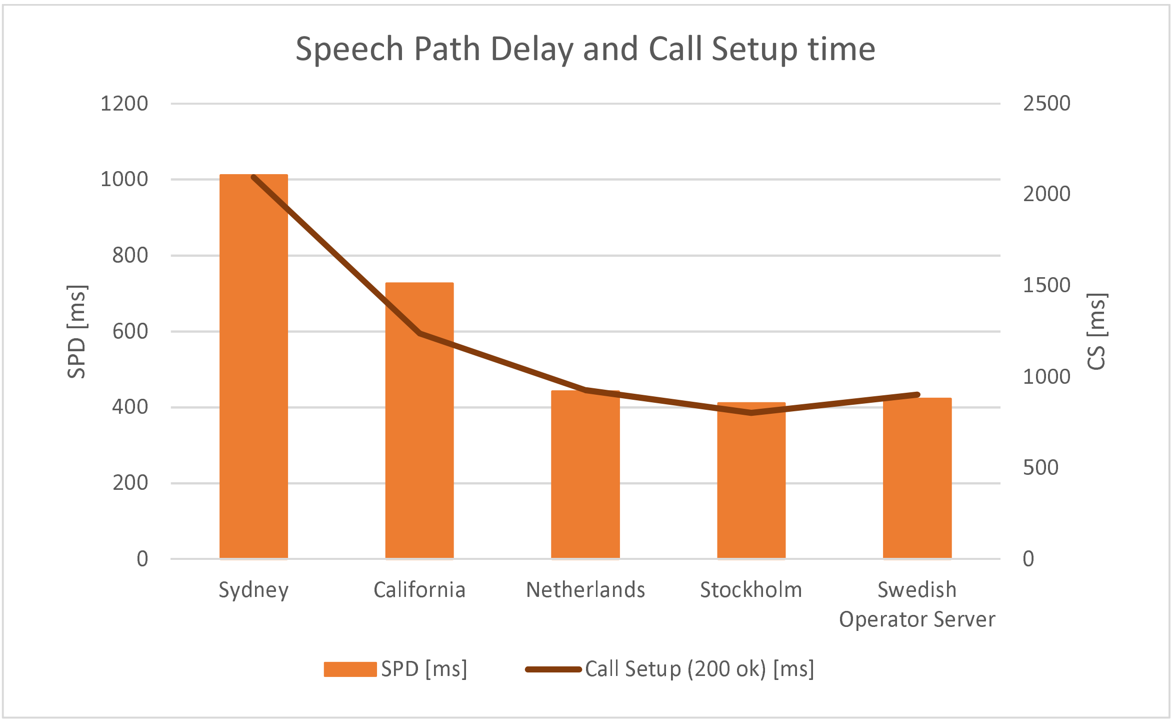


Figure 13: Example of SPD and CST dependency on SIP server location

History

Document history		
V1.1.1	October 2023	Publication