TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

JONI RÄSÄNEN

IMPLEMENTATION OF RECORDING AND PLAYBACK IN VIDEO CALL

Master of Science Thesis

Examiners: Ass. Prof. Jarno Vanne and Prof. Timo Hämäläinen
Examiner and topic approved by the Faculty Council of the Faculty of Computing and Electrical Engineering on 9th of December 2015

ABSTRACT

JONI RÄSÄNEN: Implementation of recording and playback in video call
Tampere University of Technology
Master of Science Thesis, 53 pages, 1 Appendix page
August 2016
Master's Degree Programme in Information Technology
Major: Computer Science
Examiners: Ass. Prof. Jarno Vanne and Prof. Timo Hämäläinen
Keywords: MP4, Voice Over IP (VOIP), Real-time Transport Protocol (RTP), RTP Control Protocol (RTCP), RTP/RTCP Reception Hint Track, Demultiplexing, Video call, Open-source

Recording a video call recording is beneficial in cases like job interviews and business meetings. Skype or Google Hangouts for example have no recording implemented without additional plugins. MP4 file container is capable of containing different types of media tracks, is widely used and is supported by media players and has a feature called RTP/RTCP Reception Hint Tracks. Hint tracks contain media transmission instructions, which can be used, e.g., to record RTP stream into a file. Without this information the video call session cannot be replayed afterwards. The purpose of this Thesis is to implement and verify the usage of RTP and RTCP Reception Hint Tracks in video call recording.

No open-source MP4 multiplexing library or a media player with support for RTP/RTCP Reception Hint Tracks was found, so the support had to be implemented in both the library and the media player. The setup includes Linphone and L-SMASH for recording and VLC media player for playback. The created MP4 file has two RTP Reception Hint Tracks, two RTCP Reception Hint Tracks, and two media tracks. The GSM audio is chosen because it is supported by Linphone, L-SMASH, and VLC media player. H.264/AVC is chosen for video, because it is the best available codec supported by the three software.

Tests were carried out using two laptops with both having recording enabled. From the tests it is concluded that using the RTP Reception Hint Track increases the total CPU usage by less than 1% and the size of the recorded video call by 4% over the conventional media tracks. The implementation shows that RTP Reception Hint Tracks meet well the needs of implementations with choice of different codecs.

TIIVISTELMÄ

JONI RÄSÄNEN: Videopuhelun tallentaminen ja toisto
Tampereen Teknillinen Yliopisto
Diplomityö, 53 sivua, 1 liitesivu
Elokuu 2016
Tietotekniikan koulutusohjelma
Pääaine: Ohjelmistotiede
Tarkastajat: Apulaisprof. Jarno Vanne ja Prof. Timo Hämäläinen
Avainsanat: MP4, Voice Over IP (VOIP), Real-time Transport Protocol (RTP), RTP Control Protocol (RTCP), RTP/RTCP Reception Hint Tracks, demultipleksointi, Videopuhelu, Avoin lähdekoodi

Videopuheluiden tallentaminen on hyödyllinen ominaisuus työhaastatteluissa ja puhelinkokouksissa. Skypessä tai Google Hangoutissa ei esimerkiksi ole mahdollisuutta tallentaa puhelua ilman lisäosia. MP4 säiliö voi sisältää erityyppisiä mediaraitoja ja siinä on ominaisuus nimeltä RTP Reception Hint raita. Hint raidoissa on median lähetysohjeita ja niihin voidaan tallentaa esimerkiksi RTP virtaa. Ilman näitä tietoja videopuheluistuntoa ei voida toistaa. Tämän työn tavoitteena on toteuttaa ja tarkistaa RTP/RTCP Reception Hint raidan soveltuvuutta videopuhelun tallentamiseen.

Yksikään avoimen lähdekoodin MP4 multipleksointi kirjasto tai mediasoitin ei tue RTP/RTCP Reception Hint raitoja, joten tuki täytyi toteuttaa molemmissa. Järjestelmä käyttää Linphonea ja L-SMASH:ä tallentamiseen ja VLC mediasoitinta toistoon. Luotu MP4 tiedosto sisältää kaksi RTP Reception Hint raitaa, kaksi RTCP Reception Hint raitaa ja kaksi mediaraitaa. GSM audio valittiin, koska se on ainoa Linphonen, L-SMASH:n sekä VLC mediasoittimen tukema audiokoodekki. H.264/AVC valittiin, koska se on paras näiden kolmen ohjelman tukemista videokoodekeista.

Testaus suoritettiin kahdella kannettavalla tietokoneella, jotka molemmat tallensivat. Näistä testeistä kävi ilmi, että RTP Reception Hint raitaa käytettyessä prosessorin käyttö kasvaa alle 1% ja bittinopeus kasvaa 4% verrattuna mediaraitana tallentamiseen. Toteutettu järjestelmä osoittaa RTP Reception Hint raidan täyttävän hyvin sellaisten videopuheluiden tarpeet, joissa on useampi koodekki valittavana.

PREFACE

First I would like to thank Nokia for presenting this opportunity to research the topic of video calls and get more familiar with the internal workings of video call applications and MP4 file format. Their guidance with MP4 was greatly appreciated. I would also like to thank my co-worker and friend Marko on helping with tougher parts of the implementation work and listening to me describe in great detail every frustration and small success I had. A thank you also goes to the rest of the Ultravideo team for the advice they gave when I asked.

A big thanks goes to my supervisor and Thesis counselor/reviewer Jarno for overseeing the project work and for help and encouragement in my quest for Thesis completion. His timely and throughout review helped me developed as a better writer and the amount of effort it took did not go unnoticed. Thanks also goes to reviewer Timo for very fast review of my thesis so I could graduate in August instead of September.

Tampere, 03.08.2016

Joni Räsänen

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

API	Application Programming Interface
AVC	Advanced Video Coding
AVPF	Audio-Visual Profile with Feedback
CSRC	Contributing Source
DT	Decoding time
ETSI	European Telecommunications Standards Institute
FIR	Full Intra Request, RTCP feedback message
GSM	Global System for Mobile Communications Codec
GPL	GNU General Public License
GUI	Graphical UI
GOP	Group of Pictures
H.263	A Video coding format
H.264	A Video coding format also known as AVC
H.323	Audio-visual protocol standard
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISO	International Organization for Standardization
ISOBMFF	ISO Base Media File Format
LGPL	GNU Lesser General Public License
MP4	MPEG-4 Part 14, Multimedia format
MPEG	Moving Picture Experts Group
NAL	Network Abstraction Layer
NALU	NAL Unit
NP-API	Netscape Plugin API
NTP	Network Time Protocol
PDU	Protocol Data Unit
PLI	RTCP feedback message
PPS	Picture Parameter Set
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RFC	Request for comments
RPSI	RTCP feedback message
RTCP	RTP control protocol

RTP	Real-time transport protocol
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SLI	RTCP feedback message
SPS	Sequence Parameters Set
SRTP	Secure RTP
SRTCP	Secure RTCP
SSRC	Synchronization Source
UI	User Interface
VCL	Video Coding Layer
VOIP	Voice-over IP
TLV	Type-Length-Value
WAV	Waveform Audio File Format
WebRTC	Web Real-Time Communication
W3C	World Wide Web Consortium

1. INTRODUCTION

Real-time Transport Protocol (RTP) [1] is developed by Internet Engineering Task Force (IETF) for real-time data transfer over wired and wireless networks. The quality of an RTP stream is monitored by RTP Control Protocol (RTCP) [1]. RTP and RTCP are commonly used in Voice over Internet protocol (VoIP) applications which can also be accompanied by video communication. A typical video call encompasses video and audio transmitted over a network in order to enable conversation between participants. However, VoIP applications tend to miss recording and playback functionalities for video. Skype [2], for example, requires external software or an add-on for recording [3]. One reason is that video storage and streaming are technically very different tasks.

MP4 (MPEG-4 Part 14) is a container format developed by the Moving Picture Experts Group (MPEG). The format is an extension of ISO Base Media File Format (ISOBMFF) [4]. MP4 is able to encapsulate various media contents such as video, audio, or subtitles into a single container. ISOBMFF was further extended with a feature called RTP Reception Hint Tracks, which are used to record an RTP stream to a file with additional information such as sycnhronization information that would be lost if the recording was done using traditional media tracks. RTP Reception Hint Track follows the structure of RTP Hint Track, which does not contain the media data in a track, but references to an existing media track. This Thesis presents a demostration, which takes a live environment of a video call and combines it with a passive environment of the video recording. For this the RTP/RTCP Reception Hint Tracks are well suited.

The goals of the Thesis are: 1) Implement a video call system where user is able to record the video call for later playback; and 2) verify and evaluate the usage of RTP Reception Hint Tracks in recording RTP streams of a two-way video call.

RTP and RTCP Reception Hint Tracks provide many benefits, such as: 1) An original video call can be reproduced from an MP4 file and re-sending of RTP packets can be done from an MP4 file to a new destination without losing synchronization information. 2) The more information (IP addresses, ports used, etc.) being retained in a file, the better tracking of the media sources can be facilitated within the file itself. 3) Desynchronization caused by potential clock drift can be corrected. 4) Packet losses can be detected during playback. 5) The recording operation does not require reconstruction of a correct media bit stream from the packet payloads, which could be challenging particularly in the case of packet losses.

In this work, an end-to-end video call demonstration has been used as a proof of concept to illustrate and validate the benefits of RTP and RTCP Reception Hint Tracks. The proposed system has been built on three open-source software tools: 1) Linphone version 3.8.2 [5], which has been extended for recording RTP/RTCP Reception Hint Tracks for incoming streams and media tracks for outgoing streams; 2) L-SMASH library version 2.11.2 [6], which has been extended for supporting RT-P/RTCP Reception Hint Track recording; and 3) VLC media player version 2.2.2 [7], which has been modified to demultiplex an MP4 file containing Reception Hint Tracks and playing its content. The supported operating systems for all modifications are Windows and Linux. GSM audio codec is chosen because it is supported by Linphone, L-SMASH, and VLC media player. H.264/AVC codec is chosen for video recording, because it compresses the video more efficiently than any other video codec supported by Linphone, L-SMASH, and VLC.

The remainder of this Thesis is organized as follows. Chapter 2 specifies the background for the implementation aspects of a video call system recording and playback using the RTP and RTCP Reception Hint Tracks. Chapter 3 describes the designed extensions to Linphone and L-SMASH for recording RTP/RTCP Reception Hint Tracks in MP4 file format. Chapter 4 shows the proposed implementation for the recorded MP4 file playback in VLC media player. Chapter 5 integrates recording and playback functionalities together into a bidirectional video call demonstrator. Chapter 6 evaluates the performance of the system and discusses its quality and possibilities for further research. Chapter 7 gives the conclusion.

2. IMPLEMENTATION ASPECTS OF A VIDEO CALL

2.1 Video call

Multimedia communication over the Internet is getting more popular. Previously voice was carried over Public Switched Telephone Network (PSTN), but nowadays it is increasingly done over the internet. Whatsapp [8], for example, has recently added an option for VoIP calling. Sometimes, VoIP is used interchangeably to include video calling. In this Thesis, VoIP refers to voice only communication and video call refers to video and audio communication between two participants. Video conferencing refers to communication between two or more participants with video and audio.

Video call as a concept got introduced short time after the telephone was first patented [9]. First actual videophone service operated in Germany in 1936 - 1940 [10] [11]. In the 1960s, videophones were announced by the Bell Labs. Videophones still exist today, but they have been mostly replaced by software based solutions or software phones. VoIP was invented by Alon Cohen and Lior Haramaty in 1995. Video calling was made popular by Skype [12].

Video call and conferencing are increasingly used in business context to replace remote meetings. They need at least a camera for the video input, a microphone for the audio input, a screen for the video output, and speakers or headphones for the audio output for each participant. The screen sharing feature can also be included in case of personal computers. A connection between participants is required for all types of calls. For software phones, an Internet connection is enough. A list of contacts can be kept in a remote server or each participant can have their own list.

A fully featured software phone has the following parts: 1) A way for user to interact with the software such as a Graphical User Interface (GUI); 2) a way to initiate a call session; 3) a way to set call parameters; 4) media processing such as encoding and decoding; 5) media transportation; and 6) Quality of Service (QoS) monitoring. Multimedia applications have several general-purpose protocols defined for these purposes such as Session Initation Protocol (SIP) [13] for session initation, Session Description Protocol (SDP) [14] for session negotiation, RTP for media transportation and RTCP for QoS. They are detailed in Sections 2.2.4, 2.2.5, 2.2.1 and 2.2.3, respectively. There also exists a standard called H.323 for audio-visual communication, which is a collection of protocols including RTP [15].

There are numerous closed and open-source implementations of software phone out of which Skype is currently the most popular having over 300 million active users as of 2016 [16]. Skype was first released by Ahti Heinla, Priit Kasesalu, and Jaan Tallinn and it is currently owned by Microsoft. Skype uses proprietary protocol for its networking [17]. It was originally based on a peer-to-peer architecture, but moved away to use an architecture based on supernodes. A more recent contender to video calling is a standard called Web Real-Time Communication (WebRTC) [18] and software built on it. WebRTC was developed by Google and released in 2011. There are ongoing standardisation activities in IETF [19] for relevant protocols and in World Wide Web Consortium (W3C) [20] for browser Application Programming Interfaces (API). Browser API enables video conferencing capabilities for websites and mobile applications without having to implement features such as codecs. Google Hangouts is the most wellknown application using WebRTC.

Table 2.1 lists open-source video call applications. Video conferencing is only present in Jitsi with use of Jitsi Videobridge and in Homer conferencing when broadcasting a media file. Licenses of applications are mostly GPL with Jitsi being Apache 2.0 license.

Table 2.1 *Open-source video call applications considered.*

Name	Licence	Language	Recording	Video conferencing
Empathy [21]	GPL	C	No	No
Ekiga [22]	GPL	C/C++	No	No
Homer Conf. [23]	GPLv2	C++	Yes	Only broadcast
Jitsi [24]	Apache 2.0	Java	No	With Jitsi Videobridge
Linphone [5]	GPL	C	Audio	No

Recording is not a common property in video call systems. Skype, for example, does not have a built-in recording feature, but it relies on external application or plugins for recording a video call. Among open-source software, only Homer conferencing and Linphone had some kind of recording feature implemented.

2.1.1 Video encoding

In a video call, the media is often compressed to reduce bit rate. The process of converting media data to a compressed format is called encoding. The process of decompressing the bit stream back to original format is called decoding. Encoding is usually done after media capture and decoding before playback.

The current mainstream video coding standard is Advanced video coding (AVC). AVC is published as twin text by ITU, ISO and IEC as ITU-T Rec. H.264 [25] | ISO/IEC 14496-10 [26]. It is commonly referred to as H.264/MPEG-4-AVC, H.264/AVC or MPEG-4 Part 10 AVC. In this Thesis, H.264/AVC is used. The first version of H.264/AVC was released in 2003.

The H.264/AVC bit stream consists of Intra and Inter pictures among other types. Inter frames require information from a previous Intra frame in order to be displayed. H.264/AVC picture can be divided into smaller framgents called slices. To send an H.264/AVC picture over a network with RTP, an H.264/AVC picture has to be converted to Network Abstraction Layer (NAL) Units (NALU) which are designed to be network friendly representation of the bit stream. Each NALU contains one H.264/AVC picture slice. The following NALU types are used in this work: 1) type 1 is a coded slice of Inter picture; 2) type 5 is a coded slice of an Intra picture; 3) type 7 is Sequence parameters set (SPS); and 4) type 8 is Picture Parameter Set (PPS). SPS and PPS specify parameters of the stream, which are needed for the decoder to be able to decode the stream [25].

2.1.2 Audio encoding

A previously popular audio codec used for audio compression in video call systems is Full-Rate Global System for Mobile Communications (GSM). GSM full-rate and GSM half-rate were used in GSM mobile phones working in 2G cellular networks. GSM has been specified by European Telecommunications Standards Institute (ETSI) and the latest version of GSM standard [27] was released in 2001. In the 1990s, it was a good compromise between bit rate and quality, but it loses to any modern speech codec. GSM sample rate is 8000 samples/s and GSM codec encodes 160 speech samples to 260 bits which is one speech frame. The libgsm [28] is a commonly used free implementation of GSM codec. The libgsm implementation pads the GSM speech frame to 33 bytes. This makes the libgsm bit rate constant 13.2 kbit/s regardless of content whereas H.264/AVC video bit rate is dependent on the content. This Thesis utilizes H.264/AVC for video encoding and GSM for audio encoding.

2.1.3 Linphone

This Thesis utilizes an open-source video call application called Linphone [5]. Linphone is a voice-over internet protocol (VOIP) software supporting both video and audio calls and audio conferences. It was first launched in 2001 on Linux by Simon Morlat and it has since then been ported to Windows, iOS, android, Blackberry OS5-7, Windows Phone, and web browsers. Linphone name is contraction of Linux phone and it is developed by Belledonne Communications. The libraries and their relations in Linphone are shown in figure 2.1. This Section describes Linphone, Liblinphone, oRTP, and Belle-sip, Section 2.1.4 describes the functionality of Mediastreamer2.

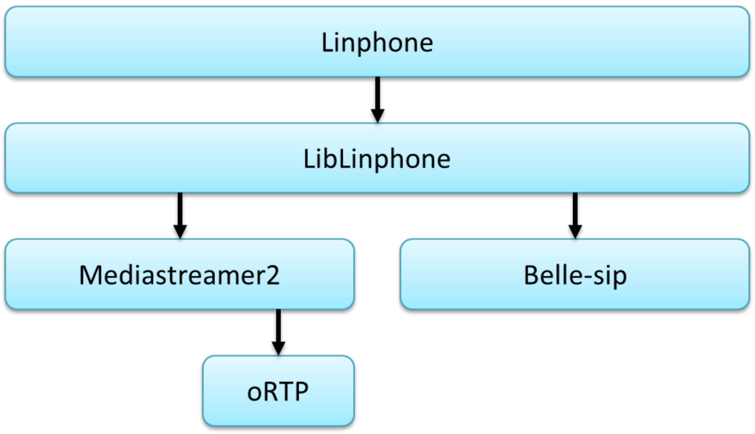

***Figure* 2.1** *Libraries of Linphone.*

Program front end is responsible for showing the program interface to user. Linphone has different front ends such as console interface, a Desktop GUI, a front end for iOS, a front end for Android written in Java, a front end for Windows Phone 8 written in C#, and a web software phone written in Javascript. In addition there is a wrapper written in Python. The support of Web browser will be removed due to the removal of the Netscape Plugin API (NP-API) from Google Chrome in April 2015 and coming removal from Firefox at the end of 2016. Windows, Linux, and Mac OS versions use the GTK+ [29] for GUI elements and it is written in C [5].

Front end is dependent on Liblinphone, which is the core engine responsible for configuration, initiation, and running of video calls. LibLinphone allows for the adjusting of call parameters and it records parameters to a configuration file that is loaded at the start of the program. Liblinphone negotiates the call with Belle-SIP library and starts the call using API of Mediastreamer2 library. Belle-sip library is responsible for SIP [13] and SDP [14] communication with clients. SDP is described in Section 2.2.5 and SIP is described in Section 2.2.4. RTP and RTCP [1] are implemented by oRTP library, which also handles operating system abstraction. RTP is described in Section 2.2.1 and RTCP is described in Section 2.2.3. Liblinphone, Belle-SIP, oRTP, and Mediastreamer2 are written in C.

2.1.4 Filter graphs

Mediastreamer2 is a Software Development Kit (SDK) for media processing and streaming. The key features of Mediastreamer2 are the audio and video filter graphs implemented according to pipe and filter design pattern which is also sometimes called pipeline design pattern. Pipe and filter design pattern means there are successive filters for data processing. These filters are run on their own threads and they concurrently wait for input to process. Mediastreamer2 library takes care of filter graph initiation, timing, transferring of data, and destruction. There are also producer filters without inputs and sink filters without outputs.

Mediastreamer2 library supports the following audio formats: OPUS, Speex, G711, GSM [27], iLBC, AMR, AMR-WB, G722, SILK, and G729; and the following video formats: VP8 in web version, H263, H263-1998, MPEG4, and H.264/AVC [25]. Mediastreamer2 utilizes RTP, RTCP, Secure RTP (SRTP) [30], and zRTP. For RTCPs Full Intra Request (FIR) control message defined in [31] is supported among other control messages. Support for different sound architectures and optimized rendering for different outputs is included as well as support for different camera APIs. Mediastreamer2 supports audio conferencing and video calls, but not video conferencing.

Figure 2.2 and figure 2.3 describe complete audio and video filter graphs of Mediastreamer2 library. In these graphs a box represents a filter and arrow shows how the data moves. Left columns of the graphs are for data sending and right columns for data receiving. For both audio and video, chosen codec determines decoder, encoder, and RTP payload format used. There is no connection between audio and video filter graphs, i.e., they run independently of each other. Both audio and video graph have a separate timing module and there is no direct connection between the filter graphs. All filters are initiated and terminated at the same time. Table 2.2 shows descriptions for filters common to both audio and video filter graphs.

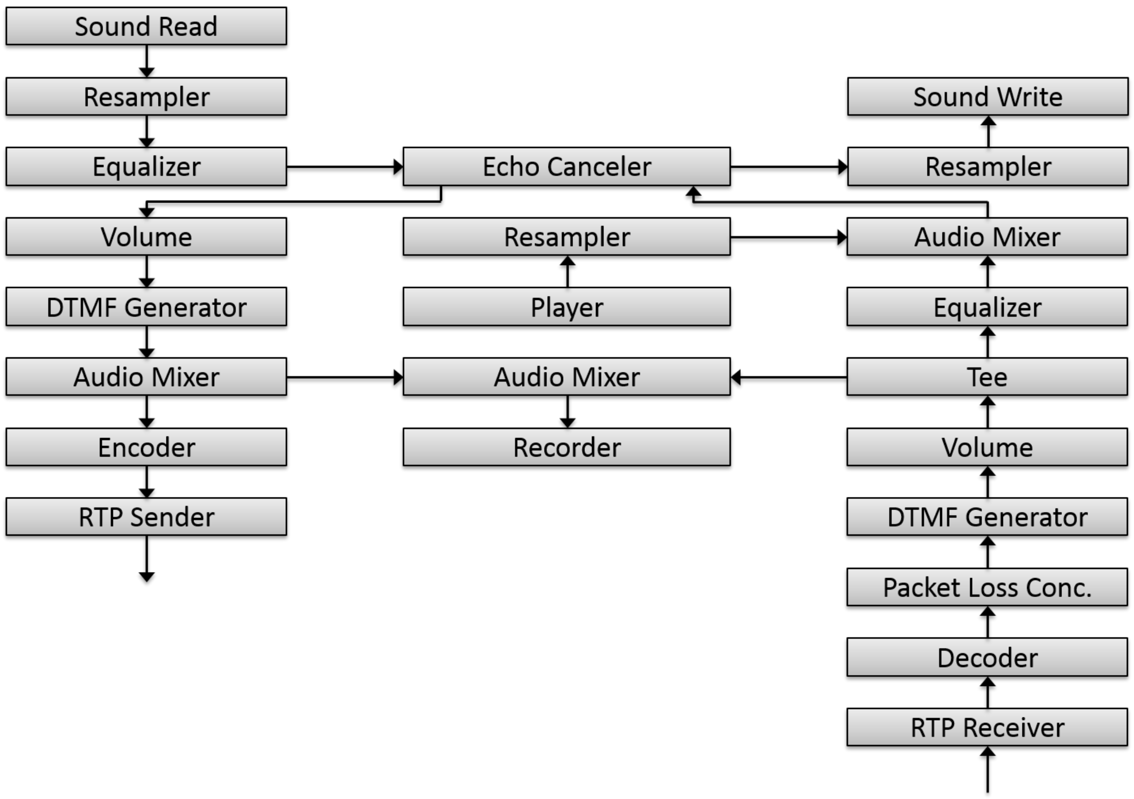

Figure *2.2 Audio filtergraph of Mediastreamer2 library.*

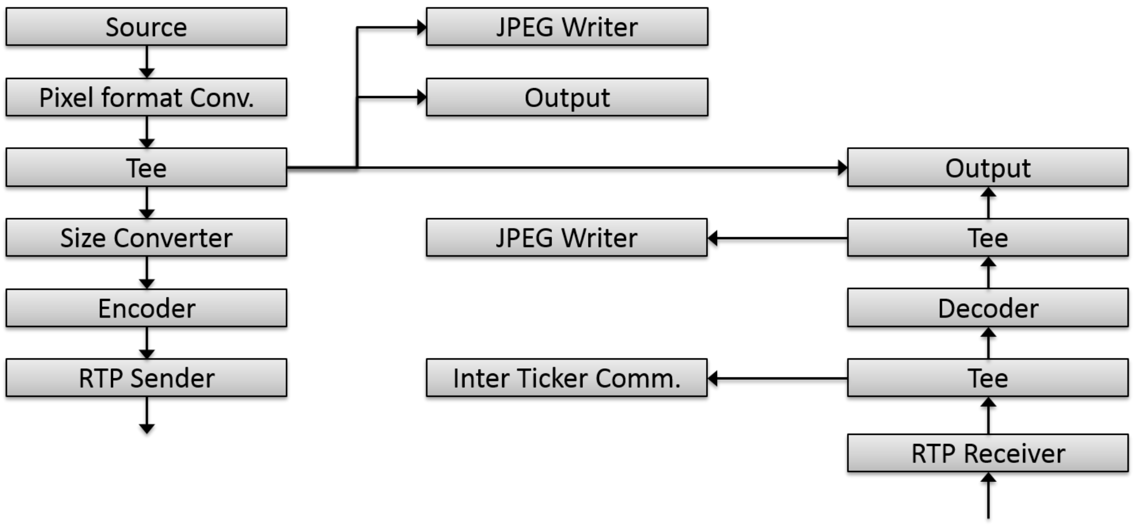

Figure 2.3 *Video filtergraph of Mediastreamer2 library.*

Table 2.2 *Mediastreamer2 filters common to both graphs.*

Filter name	**Purpose**
Encoder	Encodes the bit stream to chosen codec format.
Decoder	Decodes the encoded bit stream for playback.
RTP Sender	Packetizes the the bit stream packets to RTP and sends them to receiver.
RTP Receiver	Receives the RTP packet stream and converts it to media packets by stripping RTP header.
Tee	Copies incoming packets to all inputs.

Table 2.3 describes filters exclusive to audio filter graph. The audio stream contains a specific set of filters such as volume adjusting, volume equalization, and an Audio Mixer filter that combines both uncompressed incoming and outgoing audio stream and records combined audio stream to a Waveform Audio File Format (WAV) file with a recording filter. There is also a player filter that can play sound files from the computer during a video call. In case of GSM codec, the GSM encoder module in Mediastreamer2 references libgsm-library discussed earlier. In addition there is a separate filter graph for audio conferencing not depicted here.

Table 2.3 *Mediastreamer2 audio filter explanations.*

Filter name	**Purpose**
Sound Reader	Reads sound from soundsystem. Chosen reader depends on sound architecture in use.
Audio Resampler	Converts the audio sample rate.
Equalizer	Equalizes audio frequencies for better sound frequency balance.
Echo Canceler	Eliminates possible echo in video call by comparing outgoing and incoming streams.
Volume	Adjusts the volume of the stream.
DTMF Generator	Generates DTMF dial tones.
Audio Mixer	Mixes two audio stream to one.
Player	Plays sounds from computer such as sound call ringing or other sound effects.
Recorder	Records the stream to a WAV file.
Sound Write	Uses sound architecture to play sound.
Packet Loss Concealment	Conceals packet losses.

Table 2.4 explains filters exclusive to video filter graph. Video filter graph has no recording filter, i. e., no video recording has been implemented either. There is no filter graph for video conferencing. For H.264/AVC video encoding, an H.264/AVC plugin has to be installed to Mediastreamer2. This plugin uses x264 [32] encoder.

Table 2.4 *Mediastreamer2 video filter explanations.*

Filter name	**Purpose**
Source	A web camera or still image indication lack of web camera if web camera is not present.
Pixel format Converter	Converts pixel format to another.
Size Converter	Scales the image to a desired resolution.
JPEG Writer	Writes a screen capture image to a file.
Output	A screen that can be displayed to a user.
Inter Ticker Communication	A way of communication ticker information with audio filter graph.

2.2 Media Streaming

There are several protocols defined for use with media streaming applications. Here, the emphasis is on protocols used by Linphone where the session can be initialized by SIP, the session format can be described by an SDP message as a part of SIP, and the media can be transported with RTP.

In inter-computer communication, special attention has to be paid for the order in which bytes are arranged in a computer memory. There are two commonly used byte orders: litte-endian and big-endian. In big-endian format, the most significant byte is stored in the first byte and following bytes are in decreasing order of significance. Little-endian is a reverse of big-endian. Because different machines can have different byte orders, the default byte orders of internet protocol suite is defined as big-endian and referred to as network byte order. This means that all fields of network protocols including RTP packet header are in network byte order.

2.2.1 RTP stream

RTP is an essential part of video call systems. It is on top of low level protocols such as TCP or UDP that do not take into account such features as synchronization of transportation. That is, RTP is an applications layer protocol. RTP is needed on top of IP protocol because IP is not designed to support real-time voice or video communications [12]. Problems in the network such as packet loss and packet delays need to be addressed in a video call software. RTP is often accompanied by RTCP for Quality of service monitoring (QoS). RTP and RTCP are defined in [1] which is an Internet Engineering Task Force (IETF) request for comments (RFC) standard.

RTP is designed for multi-participant multimedia conferences. An RTP stream is composed of RTP packets, each of which consists of a header and a payload. Protocol Description Unit (PDU) [33] is one measurement of protocol characteristics. For RTP stream, PDU consists of the header and payload of an RTP packet. Payload is an audio or a video sample or part of it.

RTP header structure is shown in figure 2.4. The RTP header fields and their functions are as follows: 1) Version (V) of the RTP is set to 2 for RFC 3550 [1]; 2) padding (P) is set if the packet contains padding after payload; 3) extension (X) is set if there is a header extension after the header; 4) CSRC count (CC) tells the number of contributing source (CSRC) identifiers following the fixed header; 5) marker (M) is profile dependent identifier for significant events; 6) payload type (PT) identifies the format and interpretation of payload, which for GSM is 3 and for H.264/AVC is dynamically set within SDP message; 7) sequence number identifies the packet by increments of one and can be used by a receiver to detect packet losses; 8) timestamp is described in Section 2.2.2; 9) synchronization source (SSRC); and 10) CSRC identifiers is a list of 0 to 15 items.

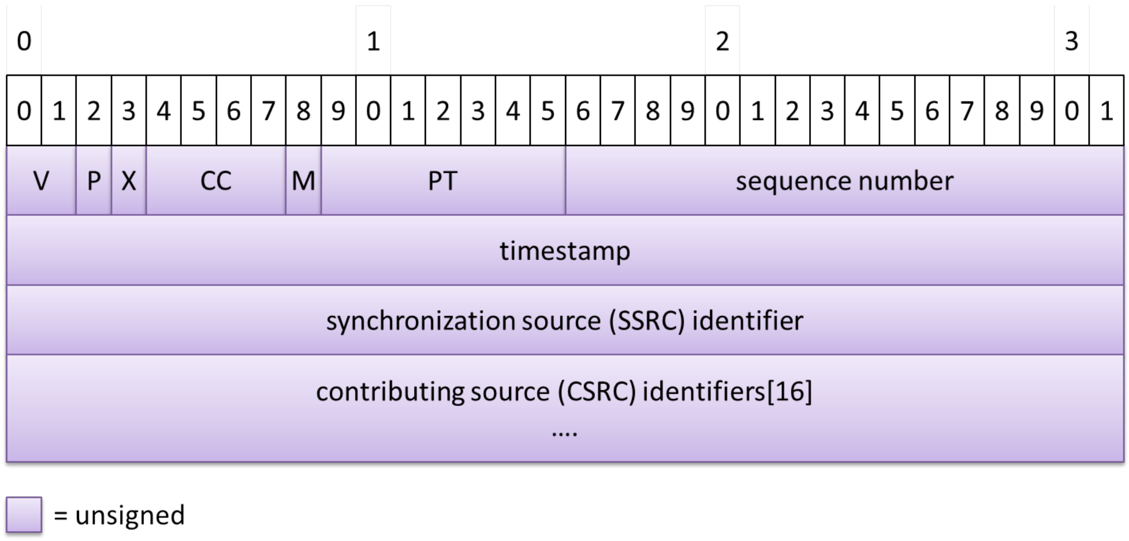

Figure 2.4 *RTP packet header structure. One row is 32 bits.*

H.264/AVC video has its own RTP payload specification defined in [34]. NAL is an H.264/AVC representation intended for network usage. Creation of NALU is done in two phases. The Video Coding Layer (VCL) outputs encoded slices and NAL encoder encapsulates VCL output into a NALU. H.264/AVC has three different packetization modes available: 1) Single NAL unit mode; 2) Non-interleaved mode; and 3) Interleaved mode. Linphone uses single NAL unit mode for H.264/AVC in which a single NAL unit packet contains only one NAL unit.

2.2.2 RTP timestamp

RTP timestamp reflects the sampling instant of the RTP packet data. The frequency of RTP timestamp follows the sample rate of contained media. For GSM audio, the clock frequency is 8000 samplings per second and for H.264/AVC video 90000 per second. Sample rate is included in the SDP message. RTP timestamp is incremental and designed to allow for synchronization and jitter calculations. The timestamps of consecutive packets will have the same value if they are generated at the same time. This is the case with H.264/AVC video picture divided into multiple slices/NALUs. The starting value of RTP timestamp is usually random for security reasons. Because RTP timestamps can advance at different rates and have random offset at the start, the synchronization of two independent RTP streams needs a common reference clock for both streams. This reference clock is the wallclock time of a RTP sender and synchronization can be done with a RTCP Sender.

2.2.3 RTCP stream

The primary function of RTCP streams is to provide feedback for the quality of the RTP stream. The RTCP messages are sent periodically and the recommended minimum interval is 5 seconds. The RTCP transmission interval in oRTP is 10 seconds. For RTCP [1] defines the following packet types: 1) Sender report; 2) Receiver Report; 3) Source description items; 4) BYE; and 5) Application-specific functions. RTCP packet structure consists of a fixed RTCP header and a structured element that depends which of the listed packet type it is. Only RTCP sender report is discussed in detail because others are not relevant to this work.

RTCP sender report holds the following parts: 1) RTCP header; 2) a sender information section shown in figure 2.5; and 3) an arbitary number of reception report blocks. The RTCP header has the following fields: 1) version (V) equal to 2; 2) padding (P) bit; 3) reception report count (RC); 4) packet type (PT) with sender report type being indicated by a value of 200; 5) length of this RTCP packet minus one; and 6) SSRC. The fields and their meaning of sender info shown in figure 2.5 are as follows: 1) 64-bit NTP timestamp indicates the sending wallclock time of the report; 2) RTP timestamp indicates the corresponding time as previous NTP timestamp with random offset; 3) senders packet count tells the number of packets that have been sent in this stream session; and 4) senders octet count tells the amount of data transferred in this stream session. RTP timestamp scale is the sample rate of the corresponding RTP stream. The NTP RTP timestamp relation can be used to synchronize two or more separate RTP streams by taking into account random offset and possibly different sample rates.

RTCP has additional messages defined in RTP Audio-Visual Profile with Feedback (AVPF) [35]. The AVPF RTP profile provides a method of timely feedback

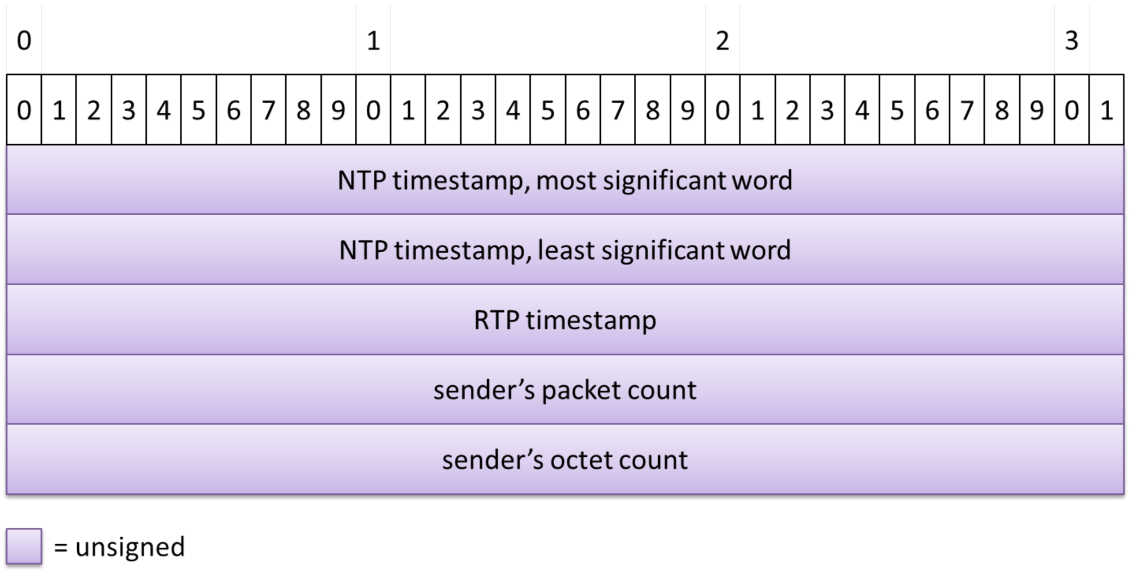

Figure 2.5 *RTCP sender info. One row is 32 bits.*

to repair a broken stream. This is done by low delay RTCP feedback messages. When AVPF is set to immediate feedback mode, it sends the feedback right after an event has occurred instead of the several second interval as in a traditional RTCP transmission. AVPF has three types of feedback messages defined in [35]: 1) transport layer messages; 2) Payload-specific feedback messages; and 3) application specific feedback messages. Transport layer has one message called NACK. Profile defines the following payload specific feedback messages: 1) Picture Loss Indication (PLI); 2) Slice Loss Indication (SLI); and 3) Reference Picture Selection Indication (RPSI). Applications layer messages are applications specific. Furthermore, IETF RFC standard [31] defines the following RTCP AVPF feedback messages: 1) H.271 Video Back Channel; 2) FIR; 3) Temporary Maximum Media Stream Bit Rate; and 4) Temporal-Spatial Trade-off. This work uses only FIR messages. When FIR is received a Decoder Refresh point is sent as soon as possible in terms of an encoder state and network resources. With H.264 RTP stream, this means sending out an Intra picture, so that decoding can be restarted from this picture. Mediastreamer2 supports all listed RTCP feedback messages.

2.2.4 SIP

SIP [13] has been developed by IETF and it is used for session control. Its use cases include discovering other instances or agreeing on what kind of session shall be created. SIP is a general purpose protocol and is meant for creating, modifying, and terminating sessions such as video calls. SIP is not dependent on the session type.

Linphone uses SIP for determining contact list informations such as location and availability and for managing the video call. At the start of a call, Linphone uses SIP to ask for capabilities and sets up the video call. SIP can also be used to modify video call by, for example, enabling or disabling the video feed of a video call. SIP is a plain text format meaning the messages are transmitted in human readably format. The SIP relies on SDP for describing the session outlook.

2.2.5 SDP

SDP [14] is an RFC standard developed by the IETF for negotiating parameters for audio and video streaming. As with SIP, SDP is also a plain text format. SDP is often used alongside SIP and RTP, but can also be used as a standalone protocol. Whereas SIP negotiates the session, SDP can be used to format the description of the session. The SDP message format is shown in tables 2.5, 2.6, and 2.7.

The whole session description message is shown in table 2.5. The left column is for the symbol of the field and the right column gives the explanation for them. Symbols marked with * are optional. The session is separated into additional sections called time description and media description. The purpose of the session description is to convey enough information that an application can join the session.

Time description part of SDP message is shown in table 2.6. Session timing specifies the start and end time of the session both of which can be set to zero for permanent session.

Media Description shown in table 2.7 concerns the description of media used in a session. It starts with an "m=" field and ends with either next "m=" field or the end of session description. The format of "m=" line is m=<media> <port> <proto> <fmt> where <media> is either "audio", "video", "application" or "message", <port> is the port which stream is sent to, <proto> is the transport protocol (in case of Linphone it is RTP), and <fmt> is the media format description meaning payload numbers if <proto> is RTP. Attributes start with "a=" and are a way of extending SDP. AVPF messages discussed in Section 2.2.3 are specified as SDP

Table 2.5 *SDP session description.*

Symbol	Meaning
v=	protocol version
o=	originator and session identifier
s=	session name
i=*	session information
u=*	URI of description
e=*	email address
p=*	phone number
b=*	zero or more bandwidth information lines
	One or more time descriptions (see table 2.6)
z=*	time zone adjustments
k=*	encryption key
a=*	zero or more session attribute lines
	Zero or more media descriptions (see table 2.7)

Table 2.6 *SDP time description.*

Symbol	Meaning
t=	time the session is active
r=*	zero or more repeat times

attribute fields. One attribute type is rtpmap which maps dynamic RTP payload types to codec used. The rtpmap format is a=rtpmap:<payload type> <encoding name>/<clock rate> [/<encoding parameters>] where <payload type> is the payload to be mapped, <encoding name> is the encoding name that this media stream will contain, <clock rate> is the sample rate of this stream which is 8000 for GSM audio and 90000 for H.264 video.

Table 2.7 *SDP media description.*

Symbol	Meaning
m=	media name and transport address
i=*	media title
c=*	connection information
b=*	zero or more bandwidth information lines
k=*	encryption key
a=*	zero or more media attribute lines

2.3 Media capture

A media container holds one or more data tracks that have a synchronized presentation. Possible container track types are audio, video, and data tracks such as hint tracks. A container is created through a process called multiplexing in which track data is inserted into container and information about tracks and the file itself is recorded to a file header. The playback of a container tracks requires a process called demultiplexing in which data from container track is processed into decodable form. Media or hint data recorded to containers can be either readily available in a file or it can be live-captured as in case of a video call recording.

MP4 (MPEG-4 Part 14) is a media container format developed by the MPEG. The format is an extension of ISOBMFF [4]. ISOBMFF and MP4 tracks are either audio, video, or hint tracks and they use box as a basic syntax element. A box is comprised of a four-character boxtype, the size of the box, and its payload. Figure 2.6 shows the mandatory boxes of an MP4 file. Each rectangle is an MP4 box. As seen in the figure 2.6, boxes may be nested so that the payload contains other boxes. Box colours help distinguish the layers of the box hierarchy. Table 2.8 explains the meaning of the boxes.

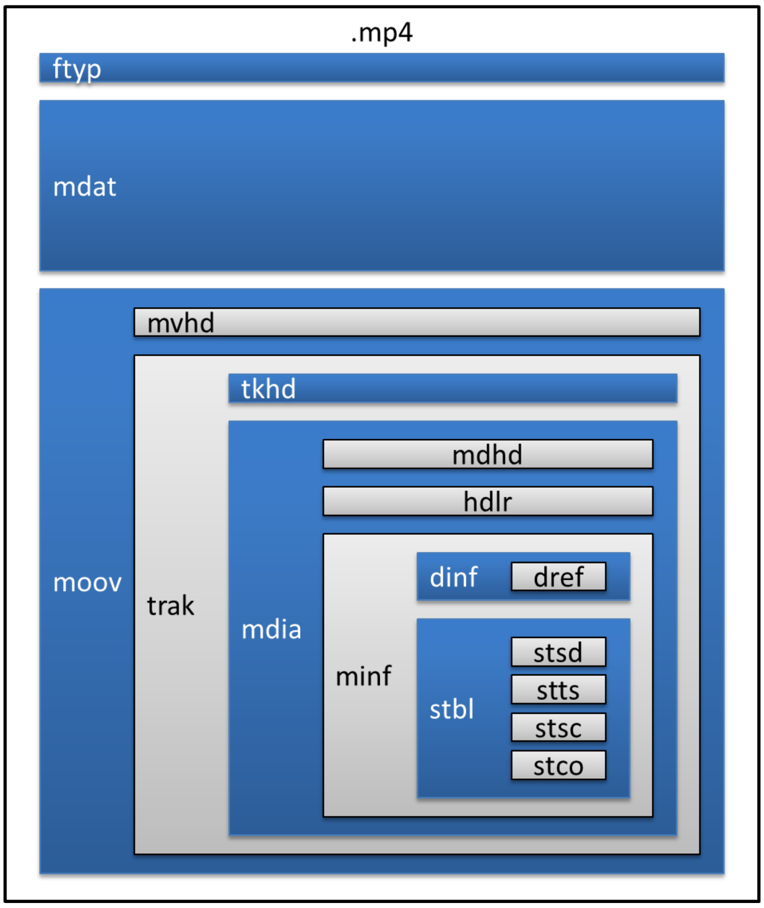

Figure 2.6 *Structure of an MP4 file showing the mandatory boxes.*

Table 2.8 *Explanations for mandatory boxes in an MP4 file.*

Four-cc	**Meaning**
ftyp	File type box holds major brands, minor brands, and compatible brands information meaning which specification they adhere to.
mdat	Media Data box holds media data which can be video pictures, audio frames or hint samples. The samples are recorded so that samples played at the same time are easy to access.
moov	Movie box holds the metadata for the presentation and is usually close to beginning of the file enabling playback of incomplete file or close to the end of the file.
mvhd	Movie Header box defines media independent information relevant for the entire presentation such as creation time, modification time, movie timescale, and duration.
trak	Track box is a container for a single track of the presentation. There can be more than one track in the presentation and each one has its own 'trak'-box. Track either contains media data or packetization instructions in case of hint tracks.
tkhd	Track Header box specifies the characteristics of the track.
mdia	Media box has all the information about track media data.
mdhd	Media Header box holds information on the media in track, but is not dependent type of media.
hdlr	Handler Reference box declares presentation process of the media. Hint Track would be handled by a hint handler.
minf	Media Information box holds characteristic information.
dinf	Data Information box holds media location objects.
dref	Data Reference box declares locations of media data.
stbl	Sample Table box contains information to determine sample on time axis and determining their type, size, container, and offset from container.
stsd	Sample Description box gives information about coding type and initialization information for coding. The information is track specific. Decoding time is the time during playback that the sample is to be decoded.
stts	Decoding Time to Sample box allows indexing of decoding time to sample number. Samples are grouped to chunks.
stsc	Sample To Chunk box table can be used to find which sample belongs to which chunk.
stco	Chunk Offset box gives the location of each chunk.

2.3.1 MP4 Hint Tracks

Hint tracks contain media transmission instructions which can be used, e.g., to record an RTP stream into a file [36]. All MP4 Hint Tracks are defined in [4]. Hint Track contains: 1) instructions on how to extract media data for playback from a track and 2) information for resending the contained media stream recorded to Hint Track itself or to media track the Hint Track is referencing. In case of reception Hint Tracks the media data is contained in hint samples instead of media samples. The following boxes are associated with the Hint Tracks: 1) Hint Media Header box ('hmhd') which contains information on bit rate and the size of PDUs within the hint track; 2) sample entry box inside stbl which has the hint track type four character code of the track; 3) Track Reference box ('tref') box for referencing information in other tracks; and 4) User Data Box ('udta') for additional information.

The primary purpose of this Thesis is to verify the usage of RTP and RTCP Reception Hint Tracks in video call recording. RTP Reception Hint Track is modelled after the RTP Hint Track which is also called the RTP server hint track for its intended use with streaming servers. Reception Hint Tracks enables the recording of a stream whereas other hint tracks contain only instructions on sending a media track as a media stream. Reception Hint tracks contain data necessary for playing and streaming the media stream contained. There are currently five types of Reception Hint Tracks. RTP, RTCP, SRTP, SRTCP, and protected RTP Reception Hint Track.

2.3.2 RTP and RTCP Reception Hint Track

An essential part of RTP Reception Hint Track is recording of SDP message. Media specific information of SDP message is stored as Track SDP information and session-level information is stored as Movie SDP information. Storing of SDP message inside 'udta'-box enables determining the type of the media in the recorded RTP stream during playback.

Hint track samples differ from media samples by having additional structures for specifying each sample. RTP sample header is shown in figure 2.7. The field packetcount tells how many RTP packets are contained within this sample. The fields reserved are not in use any of the hint track boxes.

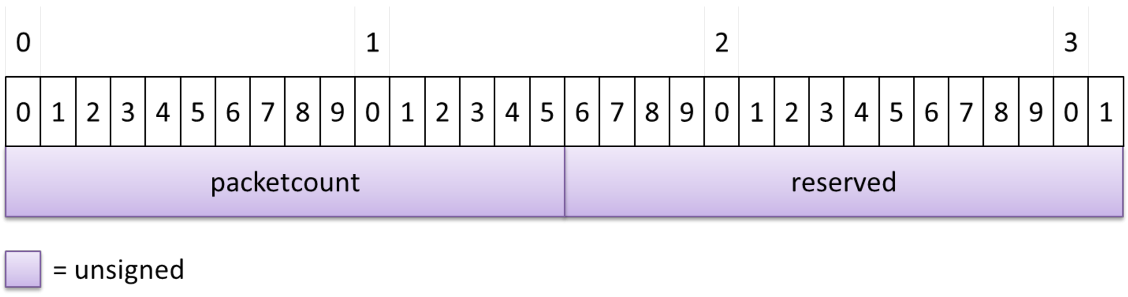

Figure 2.7 *MP4 RTP sample header and RTCP sample header structure.*

Recorded RTP packets use an RTP packet structure shown in figure 2.8. The fields V, P, X, CC, M, PT are same as in figure 2.4. If extra flag (E) is set to 1 then there is one or more Type-Length-Value (TLV) boxes. In RTP Reception Hint Tracks bframe (B) and repeat flag (R) are zero.

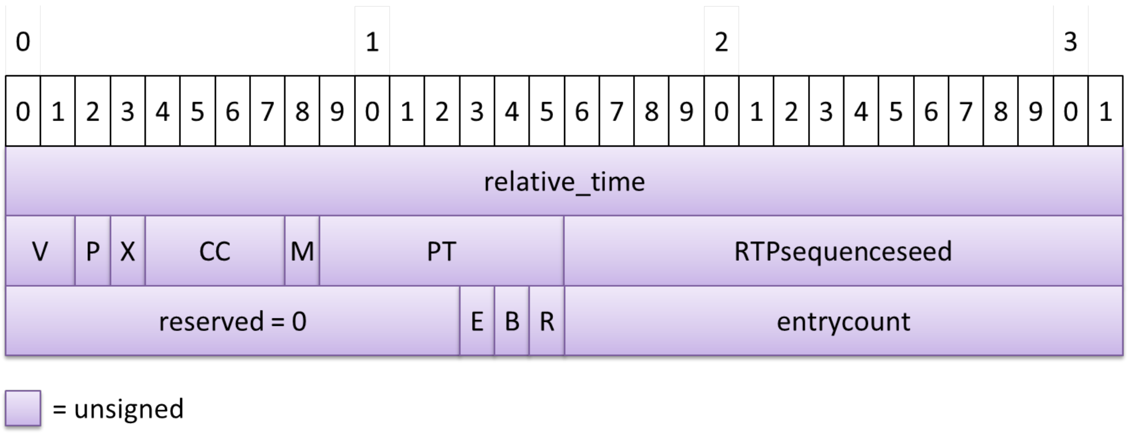

Figure 2.8 *MP4 Hint Track RTP packet.*

Each RTP packet structure contains one constructor. There are four types of constructors [4]. RTP Reception Hint tracks contains one RTPsample constructor shown in figure 2.9. RTPsampleConstructor has the following fields: 1) constructor type equal to 2; 2) trackrefindex tells which track the sample refers (for RTP Reception Hint Track this is set to -1 indicating the hint track itself). 3) length is the length of the payload; 4) samplenumber indicates which sample the payload belongs to; 5) sampleoffset field tells where the payload is located; 6) bytesperblock; and 7) samplesperblock. The latter two are legacy fields prior to MP4 which are set to 1.

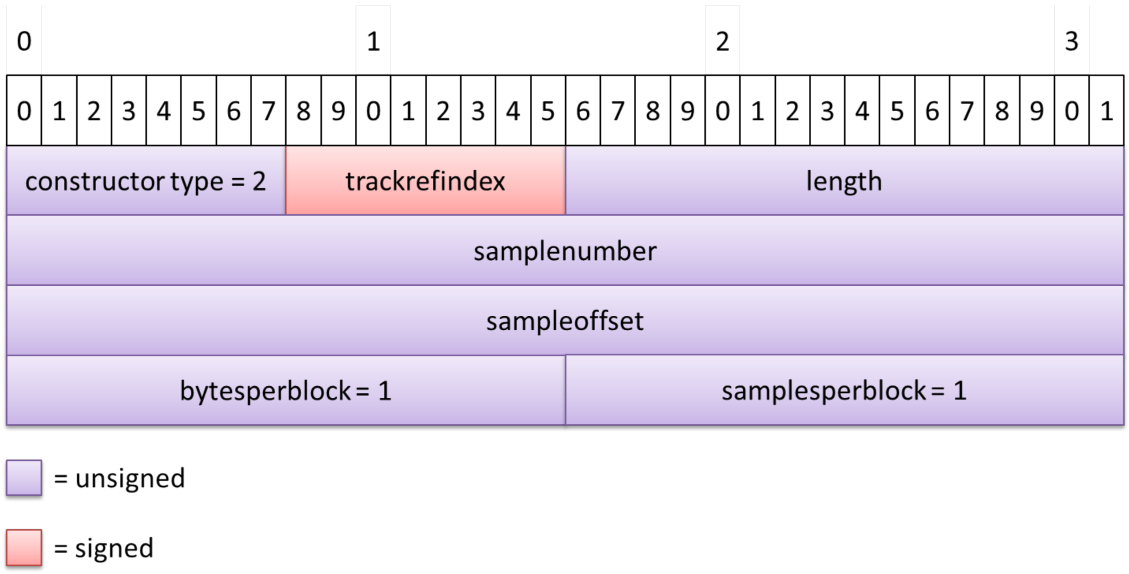

Figure 2.9 *MP4 Hint Track RTP sample constructor.*

Inside every 'stbl'-box, there is a sample entry with four character code. The RTP Reception Hint Track sample entry ('rrtp') is shown in figure 2.10. The fields hinttrackversion and highestcompatibleversion are set to 1. maxpacketsize field is the largest possible size for packets in a stream. There are the following additional boxes: 1) The timescale entry box ('tims') matches the timescale of the RTP timestamps. 2) The time offset ('tsro') box is the RTP timestamp of the first recorded RTP packet in a stream. Because the starting value of RTP timestamp is random the offset has to be recorded for synchronization between tracks. 3) The timestamp synchrony ('tssy') box identifies whether the RTP timestamps present in recording are synchronized. The value 0 indicates that the synchronization status is unknown, 1 indicates RTP timestamps have not been modified, and 2 indicates that the RTP timestamps have been synchronized with the RTP timestamps of other RTP Reception Hint Tracks. The RTCP Reception Hint Track sample entry follows the structure of RTP Reception Hint Track without additional boxes.

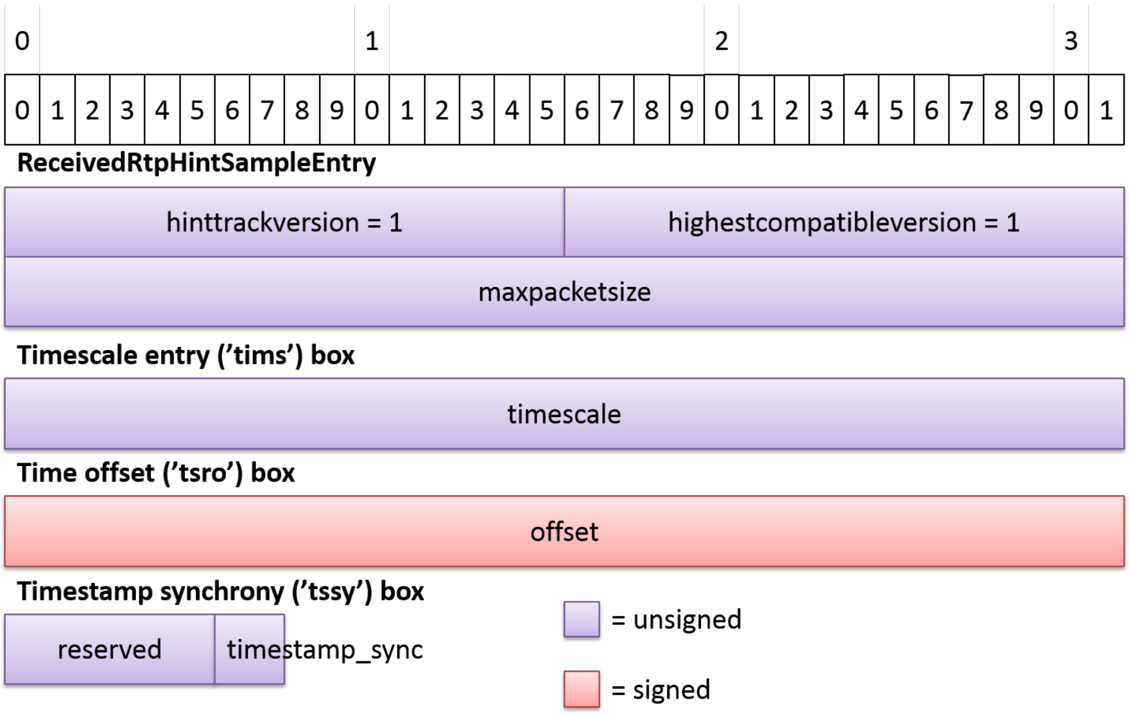

Figure 2.10 *MP4 RTP Reception Hint Track Sample Entry.*

Equation 2.1 shows the relation of RTP timestamps and recorded time information in RTP Reception Hint Track sample. The Decode time (DT) of the first RTP packet is zero. For the following packets DT is the RTP timestamp difference from the first recorded timestamp and tsro.offset is set to the RTP timestamp of the first recorded packet. The additional offset in the equation is the offset field of the rtpoffsetTLV-box.

$$RTPtimestamp = (DT_i + tsro.offset + offset)mod^{32} \tag{2.1}$$

RTP Reception Hint Track samples are composed of a sample header and an equal number of the following: 1) details from an RTP packet header; 2) a sample constructor; and 3) an RTP packet payload. Sample gathers all RTP packets that have the same RTP timestamp. As H.264 video picture is divided to multiple NALUs, a H.264 RTP Reception Hint Track sample has all these RTP packets in same sample.

The figure 2.11 shows an example of RTP Reception Hint Track sample holding three recorded RTP packets with the same timestamp. The data boxes in red are presented earlier. The blue boxes represent the payload whose size depends on codec. If a sample would have an additional packet, it would add an additional RTP packet structure, RTPsampleconstructor structure to the end of packets and one payload to extradata.

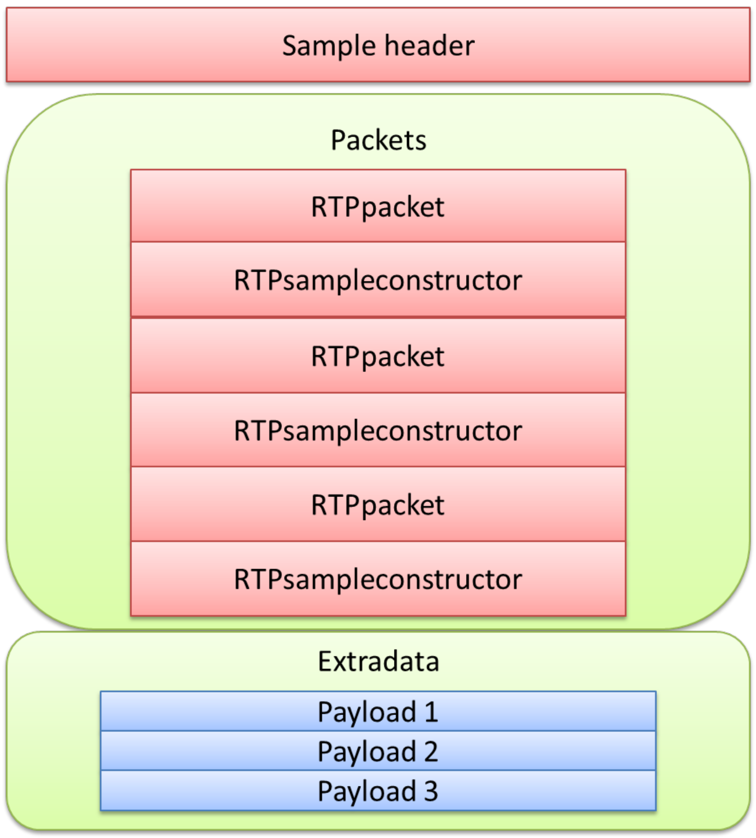

Figure 2.11 *Example RTP Reception Hint Track sample.*

RTCP Reception Hint Tracks are used to maintain inter-stream synchronization especially when recording starts mid-stream or clock drift exists [4] [36]. RTCP Reception Hint Track is used for synchronization of corresponding RTP Reception Hint Track. This is done by recording RTCP sender reports shown in figure 2.5 as samples which enables using NTP timestamps to be used in sycnhronization. The relation of RTP and NTP timestamps can be used for synchronization of RTP timestamps within RTP Reception Hint Tracks to other RTP Reception Hint Tracks. Edit List ('elst') box makes possible synchronization with media tracks. RTCP Reception Hint Tracks refer to an RTP Reception Hint Track. Track reference box ('tref') is needed with reference type cdsc for RTCP Reception Hint Track to tell which RTP Reception Hint Track it is referring. Table 2.9 shows all the boxes related to RTP/RTCP Reception Hint Track implementation.

Table 2.9 *MP4 boxes related to RTP/RTCP Reception Hint Tracks.*

Four-cc	**Meaning**
rrtp	Received RTP Hint Sample Entry Box is used with RTP Reception Hint Track to describe characteristics of samples within track. It is shown in figure 2.10. Has three additional boxes: tims, tsro and tssy.
rtcp	Received RTCP Hint Sample Entry is identical to RTP Hint Sample Entry. It has no additional boxes.
tims	Timescale Entry Box tells the clock frequency of RTP timestamps. It is shown in figure 2.10.
tsro	Time Offset Box used in deriving value of RTP timestamp in equation 2.1. Structure is shown in figure 2.10.
tssy	Timestamp Synchrony Box tells whether timestamps of this track has been synchronized with other RTP Reception Hint Tracks. It is shown in figure 2.10.
udta	User Data Box is a container for user information boxes such as 'hnti'-box.
hnti	Movie Hint Information Box contains the movie part of SDP message shown in table 2.5. Movie part is all the lines before first media description.
hnti	Track Hint Information Box contains one media description of SDP message shown in table 2.7.
tref	Track Reference Box is a container box for track references such as 'cdsc'-box.
cdsc	Content Description Reference Box for telling which RTP Reception Hint Track RTCP Reception Hint Track is associated with.
elst	Edit List Box describes when the track is played.
hmhd	Hint Media Header Box contains general information of contained stream.
nmhd	Null Media Header Box may be used if other media header boxes are not applicable.

2.3.3 Multiplexing an MP4 file

Dictionary of Computer Science defines multiplexing as "The process of combining multiple messages simultaneously on the same physical or logical transmission medium" [37]. In case of a container multiplexing refers to the process of including several types of media to tracks inside the container. Creating a container usually happens after encoding so the data is in a compressed format.

L-SMASH is an MP4 writer and file-format library written in C. It is capable of creating an MP4 container and writing audio and video data into a file. It supports both multiplexing and demultiplexing. L-SMASH was started by Nakamura Yusuke in 2010 as an H.264/AVC multiplexer for MP4. Later, it has been expanded to support various standards such as ISOBMFF [4]. It is used through an API and has functions for operating the library and file creation.

The whole L-SMASH project encompasses around 50000 lines of code. It can create multiple tracks inside an MP4 file and samples are appended individually to a particular track. The MP4 track header data is inputted to library when creating a track and written to a file after the recording samples is finished. The samples are recorded into the file as they arrive. L-SMASH supports 55 codec identifiers for audio and 78 codec identifiers for video. It is used by video service Vimeo [38] and H.264/AVC encoder x264 [32].

2.4 Media playback

A media player is needed to play the contents of media container such as MP4. The media player has to support both the container format and the codec used for media. Popular open-source media players include VLC Media Player [7] and Media Player Classic - Home Cinema [39] of which the former is used in this Thesis.

2.4.1 VLC media player

VLC is an open-source media player written mostly in C and developed by the VideoLAN project. Its core is licensed in LGPL and the other parts in GPL or LGPL. VLC is capable of streaming media over the internet, saving it to a file and claims to support all formats. It can play one or more video tracks at the same time but only one audio track. Windows, Linux, MAC OS, Android and iOS versions of VLC exist. Bindings to several languages enable its usage in other programs.

VLC uses modular design and consists of VLC core and large number of modules. VLC core is responsible for loading the modules. Figure 2.12 shows the main modules needed in VLC file playback. A file is first read by an access module which in this case is the filesystem access module. After that, the stream module takes care of byte order. In case of an MP4 file, the demux module represents an MP4 demultiplexer. After demux module, a decoder module decodes the stream. An output module includes the video or audio filter modules and also the modules for displaying the GUI elements of VLC Media Player.

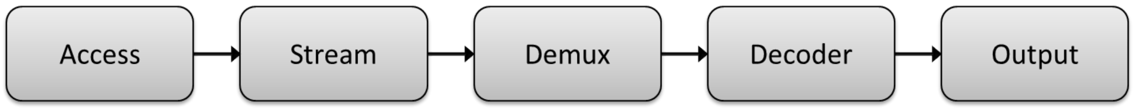

Figure 2.12 *VLC high level data flow from file to screen.*

2.4.2 Demultiplexing an MP4 file

Demultiplexing is the process of taking individual streams from a container and sending them, e.g., for decoding. A media player has to be able to demultiplex different containers. The demultiplexing process is taken into account when designing a specification for a container and demultiplexing process has to follow such specification or standard. Demultiplexing reads the data in container and uses it to facilitate the decoding and playback of the stream. The data in MP4 boxes shown in table 2.8 is readily available without the need to decode the bit stream.

VLC MP4 demux module supports wide variety of possible audio and video tracks, but it has no hint track support. VLC Demux module works in four steps: 1) the boxes are read from filestream and the box structure is recorded to VLC; 2) the VLC track is created from recorded boxes, VLC track holds data such as timescale and some of the boxes needed for facilitating demultiplexing; 3) the elemental stream module is created; and 4) the samples are demultiplexed one by one.

3. IMPLEMENTATION OF VIDEO CALL RECORDING

3.1 Media recording in Linphone

The purpose of this Thesis is to implement a video call recording and playback using RTP/RTCP Reception Hint Tracks. Storage of SRTP, SRTCP, and protected RTP are out of scope of this Thesis. None of the open-source video call software has video call recording implemented. Among existing alternatives, Linphone was chosen because it had existing audio recording feature which makes the needed modifications to UI minimal. Ekiga[22] and Homer Conferencing [23] were also considered, but Ekiga lacked any recording features and the development activity of Homer Conferencing was seen too low. Jitsi [24] was left out of consideration because it is written in Java.

The development of Linphone recording feature went alongside with updates to Linphone. The final Linphone version used is 3.8.3. This version was chosen because further version updates would have involved large modifications to RTP send and receive filters that would have required effort to reimplement the RTP header parsing. Linphone lacked support for multi-person video conferences, so recording for only two participants is implemented. L-SMASH was chosen for multiplexing of an MP4 file.

Linphone is extended to record 1) incoming audio/video RTP packets into RTP Reception Hint Tracks; 2) RTCP sender reports into RTCP Reception hint tracks; and 3) outgoing audio/video content into media tracks. The RTCP Reception Hint Tracks are recorded to provide synchronization information for their corresponding RTP Reception Hint Tracks. The incoming packets are assumed to originate from another Linphone running on a separate computer. GSM codec is chosen for audio and H.264 codec for video.

Section 3.1.1 shows the high level modifications to Linphone. More detailed implementation of file creation module, media recording filters, RTP recording filters, RTCP recording module are shown in Sections 3.1.2, 3.1.3, 3.1.4, 3.1.5 respectively. The rest of this Section shows modifications to Linphone outside Mediastreamer2 library. All modifications were primarily developed in MinGW [40] environment on Windows and modifications were tested also on Linux.

Figure 3.1 shows which parts of Linphone were modified. LibLinphone needed modifications for capturing the SDP message and for providing video stream data to recording. Recording is implemented in Mediastreamer2 library as filters in filter graph. Belle-Sip did not need modifications. L-SMASH was added to Linphone to facilitate multiplexing.

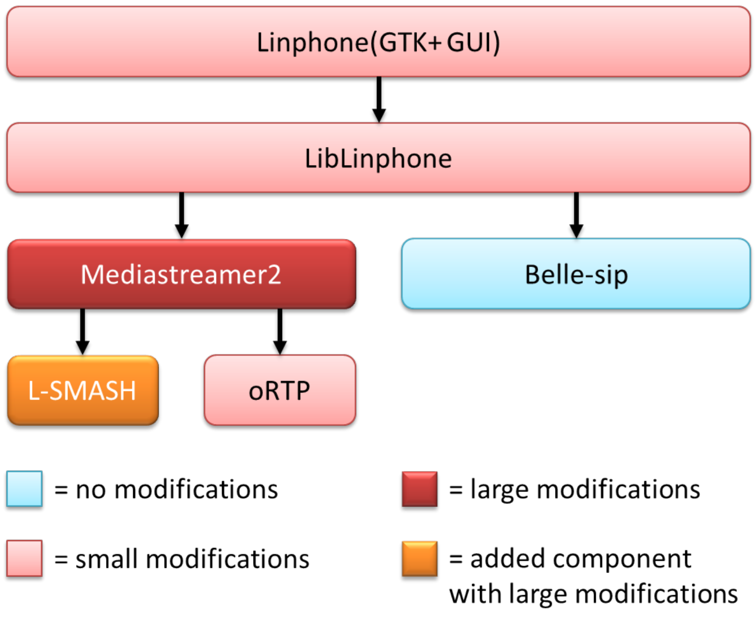

Figure 3.1 *Modified parts of Linphone.*

Linphone H.264/AVC stream has intra pictures every 10 seconds. This would cause a random delay of 0 to 10 seconds before the stream can be played during playback. To make the first intra picture come faster a full intra request (FIR) can be used or the video stream can be changed to all-intra. The FIR request is preferred to using all-intra because all-intra would cause bit rate increase. It would also require modifications to Encoder filter in video filter graph which would have an impact on bitstream and possibly have unpredictable consequences in other filters. The FIR request is sent by enabling AVPF in oRTP and in linphone configure file. Sending the request happens when recording starts, which results in intra picture arriving within few pictures when the recording button is pressed by the user.

3.1.1 Audio/Video filter graph modifications

Liblinphone calls directly recording functions in Mediastreamer2 API so the functionality of these functions is changed to control filters through a controlling module. These filters were inserted into Mediastreamer2 filter graphs to obtain the requested data. Figure 3.2 and figure 3.3 show the added filters and their locations in audio and video filter graphs of Mediastreamer2 library on Windows, respectively. The gray boxes are existing filters that did not require modifications whereas black boxes are new filters added by this work to facilitate different recording operations.

In figure 3.2 there are three new filters added and one filter was modified. Firstly a GSM recorder filter was added to record a GSM track. Secondly the same Tee filter which already exists in Mediastreamer2 was added to direct output from GSM encoder to GSM recorded. Thirdly, in order to get the RTP packets in a correct format, an RTP receiver filter was modified so that it does not strip the needed RTP header and outputs the whole RTP packet to a separate filter output. In both video and audio graphs, a new filter called RTP recorder was attached to this output.

Video filter graph modifications are shown in figure 3.3. Video recording follows the same principle as with audio recording expect GSM recorder has been replaced by H.264 recorder. Same RTP recorder is used in both filter graphs.

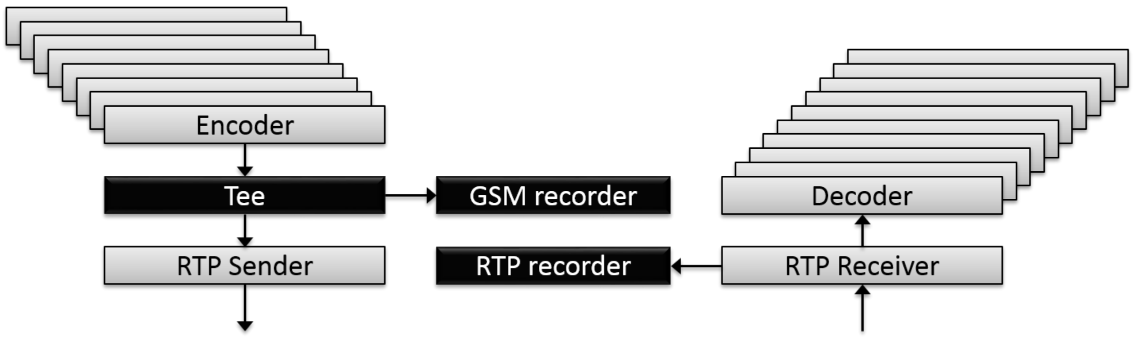

Figure 3.2 *Modifications done to Linphone audio filter graph.*

Initialization of recording and user actions for recording are handled in a separate

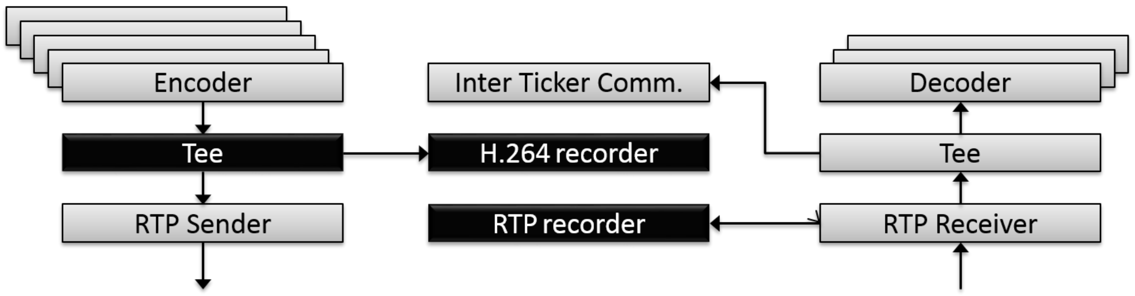

***Figure 3.3** Modifications done to Linphone video filter graph.*

control module. Whenever the user starts, pauses or ends the recording, the control module calls each filter consecutively to perform respective action. The filters have functions for initializing, opening, closing, uninitializing, and processing. Each new filter in Linphone creates an MP4 track. In addition there are two RTCP tracks that are created from RTCP sender reports captured. SDP message shown in table 2.5 has a media section for each transported media stream. The media sections are divided to their respective tracks and movie part of the message is handled by the file module presented in next Section. See 'hnti'-boxes in table 2.9 for where each part is stored within the MP4 file.

3.1.2 MP4 file creation

The file module is used to create the MP4 file through L-SMASH. The tasks of the file module include the creation of MP4 file header and opening/closing the file on filesystem. As L-SMASH is not thread safe this module takes care of multi-threaded access to a file by mutual-exclusion. Each filter asks to use L-SMASH specific structures and the file module uses a mutex to ensure only one filter uses L-SMASH at a time. The brands in 'ftyp'-box are set as isom and avc1, see table 2.8 for 'ftyp'-box. This also leaves out from the MP4 file boxes that are not part of ISOBMFF. Edit List Box ('edts') shown in table 2.9 was left out because no media player supported playing two audio tracks at the same time so possible desynchronization between GSM audio track and GSM RTP Reception Hint Track could not be detected in because the listener would only hear one at a time.

3.1.3 Media recording filters

GSM audio is chosen because it was supported by Linphone, L-SMASH, and VLC media player and because it does not require additional plugins. The MP4 track header and samples are created using L-SMASH. The samples are recorded into the file while the video call is taking place. Because RTP packet payloads are GSM samples inside MP4 GSM media track, they can be appended to track without modifications.

H.264/AVC codec is chosen for video recording, because it compresses the video more efficiently than any other video codec supported by Linphone, L-SMASH, and VLC. In Linphone, a separate plugin is needed for H.264/AVC. Linphone H.264/AVC stream uses slices that divide the picture to multiple RTP packets. When recording H.264/AVC stream to a file, the RTP packets have to be combined to a picture which is then recorded as sample to a H.264/AVC track inside an MP4 container. Video recording was done by combining the NALUs into a picture and removing start codes.

3.1.4 RTP Recording filter

RTP Recording filter records incoming RTP packets to corresponding RTP Reception Hint Track. The recording is done in two parts. First, the sample is constructed from one or more RTP packets and then it is appended to the track. RTP Reception Hint Tracks are not synchronized during recording but synchronization is done during playback using the recorded RTCP Reception Hint Tracks.

In case of audio, the necessary information from RTP header is copied to RTP packet. The sample is constructed as 1) sample header (see figure 2.7); 2) RTP Packet structure (see figure 2.8); 3) sample constructor (see figure 2.9); and 4) RTP packet payload (audio sample). The packet DTS is set as RTP timestamp of the recorded RTP packet minus the RTP timestamp of first recorded RTP packet. The RTP packet structure fields and their values are shown in table 3.1. The relative time field is set to 0 since it is used for traffic smoothing and there is no plan on resending the RTP stream from track.

Table 3.1 *Values of RTP packet structure in implementation.*

Field name	**Value**	**Meaning**
relative time	0	Not in use.
version (V)	2	RFC 3550 RTP
padding bit (P)	0	No padding.
extension bit (X)	0	No header extensions.
CSRC count (CC)	0	No CSRCs.
marker bit (M)	0 or 1	1 for last slice of a video picture and 0 for others.
payload type (PT)	3 or 96	3 for GSM and dynamic (96) for H.264/AVC.
RTPsequenceseed	-	Same as in sequence number in RTP packet.
reserved	0	Always 0 for RTP reception Hint track.
extra flag (E)	0	No TLV boxes included.
bframe (B)	0	Always 0 for RTP reception Hint track.
repeat flag (R)	0	Always 0 for RTP Reception Hint Track.
entrycount	1	One constructor.

In case of video, the sample constructing is more complicated. All slices belonging to the same picture must be recorded to one sample. This is done by gathering all RTP packets with the same RTP timestamps until a packet with different timestamp arrives. In addition, packetcount in RTPsample is set and RTP Packet structures, constructors for each packet, and the payloads are added. See figure 2.11 for an example of H.264/AVC RTP Reception Hint Track sample. Constructor is RTP-sampleconstructor shown in figure 2.9.

3.1.5 RTCP Recording module

RTCP recording is not done in a filter, but in terms of controlling the recording, it is treated like recording filters. RTCP recorder uses L-SMASH to create necessary header information and to append samples to a track. The RTCP recording module produces an RTCP Reception Hint Track in the MP4 file. In both audio and video mediastreams sender report RTCP messages are sent to RTCP recording module. Sender report timestamps are converted to host byte order and set as the DTS of the sample. The sample rate is the same as with the corresponding RTP Reception Hint Track.

3.2 Media multiplexing in L-SMASH

Linphone lacks the functionality to encapsulate media data into an MP4 file. Using an existing library for that saves the effort of implementing the MP4 file format. However, an open-source library with support for RTP/RTCP Reception Hint Tracks was not found, so the support for it had to be implemented. L-SMASH was chosen for multiplexing the MP4 file. This work extends L-SMASH with support for RTP/RTCP Reception Hint Tracks and for hint tracks in general. The modified version of L-SMASH library can also be used to record RTP/RTCP Reception Hint Tracks in other RTP applications. For a complete MP4 file all the mandatory boxes of table 2.8 and all the Hint Track related boxes shown in table 2.9 are needed. All the mandatory boxes and 'udta'-box, 'elst'-box and 'tref'-box of hint track functionality have been implemented. The rest of the hint track boxes were implemented.

The implementation of 'hmhd'-box was most challenging. The field maxBitrate of 'hmhd'-box indicates the largest bit rate of a sample and avgBitrate the average bit rate of all the samples of the track it belongs to. They were implemented by reusing existing fuction for btrt-box after all the samples have been recorded. The PDU sizes are calculated in runtime by reducing the sizes of RTPsample header, RTP packet structure, and constructor from sample size and adding the size of RTP header. The maxPDUsize calculation was implemented by comparing the sample PDU size to previous largest size and replacing if it was larger. For this it, was necessary to get the length of payload from constructor for calculation. The codec information to Sample Entry box for RTP Reception Hint Track ('rrtp') needed to be added including input fields to interface, which meant adding a necessary a new type of summary with needed fields. The data for additional boxes was implemented in codec specific data struct in L-SMASH.

4. MEDIA PLAYBACK IN VLC

4.1 MP4 demux module modifications

The recorded MP4 container file can be be played back by a player. In this work, the support for RTP/RTCP Reception Hint Track playback was implemented within the popular VLC media player since none of the existing players support RTP/RTCP Reception Hint tracks.

In the implementation, VLC media player reads the MP4 file containing RTP/ RTCP Reception Hint Tracks as well as media tracks and processes them according to figure 2.12. The MP4 demux module is chosen for demultiplexing. It demultiplexes RTP/RTCP Reception Hint Tracks to elemental streams and synchronizes them to each other using corresponding RTCP Reception Hint Tracks. In the proposed VLC extension, the playback of RTP Reception Hint Track is implemented as suggested in [4] and [36]. Modifications in VLC are limited to MP4 demuxer module so implementing RTP Reception Hint Track support is relatively easy. That is smaller part of program functionality has to be known before making the changes. The supported codecs are Speex and GSM for audio and H.264/AVC for video. The modifications were done in Ubuntu Linux using GCC compiler. Windows version of VLC software was created using MinGW Linux tool chain.

The first step of implementing supporting for the RTP Reception Hint Track is initializing data needed for playback of a track. An SDP message of table 2.5 is parsed to find out whether the RTP Reception Hint Track contains audio or video. Next, an elemental stream is created. It involves parsing the rest of the SDP message for media information. Table 4.1 shows an example of a Linphone SDP message with H.264/AVC and GSM payloads. The identifier rtpmap shows the mapping of payload types to codecs and the identifier rtcp-fb shows minimum interval of RTCP messages and additional RTCP feedback message types. Such message is recorded to an MP4 file as described in Section 2.2.5. VLC demux module parses the SDP message field at a time until it finds the codec type and sample rate used. For H.264/AVC track, it can be found in "a= & rtpmap:96 H264/90000" line, where H264 is the codec H.264/AVC and 90000 is the sample rate. For GSM, payload type is the number 3 in "m= & audio 7078 RTP/AVPF 3 101" as described in Section 2.2.5. Implementation is able to parse any SDP message which follows the format specified in [14].

Table 4.1 *Example Linphone SDP message with GSM and H.264 payloads.*

Symbol	Value
v=	0
o=	thinkpad 978 2342 IN IP4 192.168.0.3
s=	Talk
i=*	IN IP4 192.168.0.3
t	0 0
a=*	rtcp-xr:rcvr-rtt=all:10000 stat-summary=loss,dup,jitt,TTL voip-metrics
m=	audio 7078 RTP/AVPF 3 101
a=	rtpmap:101 telephone-event/8000
a=	rtcp-fb:* trr-int 5000
m=	video 9078 RTP/AVPF 96
a=	rtpmap:96 H264/90000
a=	fmtp:96 profile-level-id=42801F
a=	rtcp-fb:* trr-int 5000
a=	rtcp-fb:96 nack pli
a=	rtcp-fb:96 nack sli
a=	rtcp-fb:96 nack rpsi
a=	rtcp-fb:96 ccm fir

4.2 Demultiplexing RTP packets

The fields in samples and are converted from network byte order to host byte order using existing VLC functions. Demultiplexing a sample for GSM and Speex RTP packets involves stripping RTP sample header, RTP Packet, and constructor parts from sample data and leaving payload. The remaining RTP payload is sent forward to the elemental stream as a media sample. With H.264/AVC video, in addition to stripping, all RTP payloads of the same sample are combined to a single picture. The number of RTP packets in the sample is identified in the packetcount field of the header. The data from constructor is used to determine the location of each payload and the synchronization markers are removed from the picture data. These modifications result in a sample that can be passed on to elemental stream module for further processing such as decoding.

4.3 Synchronization of RTP Reception Hint tracks

The RTCP header and sender info blocks are read from sample data of RTCP Reception Hint Track. There is a random offset in the first samples in RTP Reception Hint Tracks due to the arbitrary starting point of the recording and random offset of RTP timestamps as described in Section 2.3.2. The offset is synchronized using the NTP timestamps of RTCP sender report so that both tracks start simultaneously. This was done by recording NTP and RTP timestamps of the first RTCP sender reports and calculating the RTP timestamp of the first RTP Reception Hint Tracks sample using equation 2.1. This RTP timestamp is then converted to NTP timestamp using the captured NTP and RTP timestamps from sender report. The offset is calculated between NTP timestamps of different tracks. This offset is converted to RTP timescale using sample rate and added to each DTS of each sample of the track which had larger NTP timestamp for first sample in effect delaying that track by calculated offset.

RTCP Reception Hint Track enables the synchronization of clock drift. However, this was not implemented because the effect of clock drift was found to be neglible. Testing clock drift correction would have required a special recording arrangement where clock drift is added to it. A quick attempt was made for clock drift correction, but it did not succeed and was determined not to be worth the effort.

5. THE OVERALL VIDEO CALL SYSTEM

5.1 System architecture

Figure 5.1 and 5.2 present two snapshots of the proposed video call system with two participants (A and B). The respective block diagram of this proof-of-concept architecture is shown in figure 5.3. The system starts with a video call taking place between two participants. One participant initiates the video call and the other answers.

In figure 5.1, the participant A is recording a two-way Linphone video call. The video call is being recorded into an MP4 file on computer A. Linphone creates the MP4 file using the L-SMASH library. The MP4 file includes RTP and RTCP Reception Hint Tracks for incoming GSM audio and H.264/AVC video and media tracks for outgoing GSM audio and H.264/AVC video. The tracks are played back by VLC media player as shown in figure 5.2 where only the playback functionality of the instance A is depicted. The playback of RTP Reception Hint Tracks is synchronized using RTCP Reception hint tracks. Participant A opens the MP4 file with modified version of the VLC Media player. VLC plays H.264/AVC video stream, RTP Reception Hint Track video stream and either GSM audio stream or RTP Reception Hint Track audio stream.

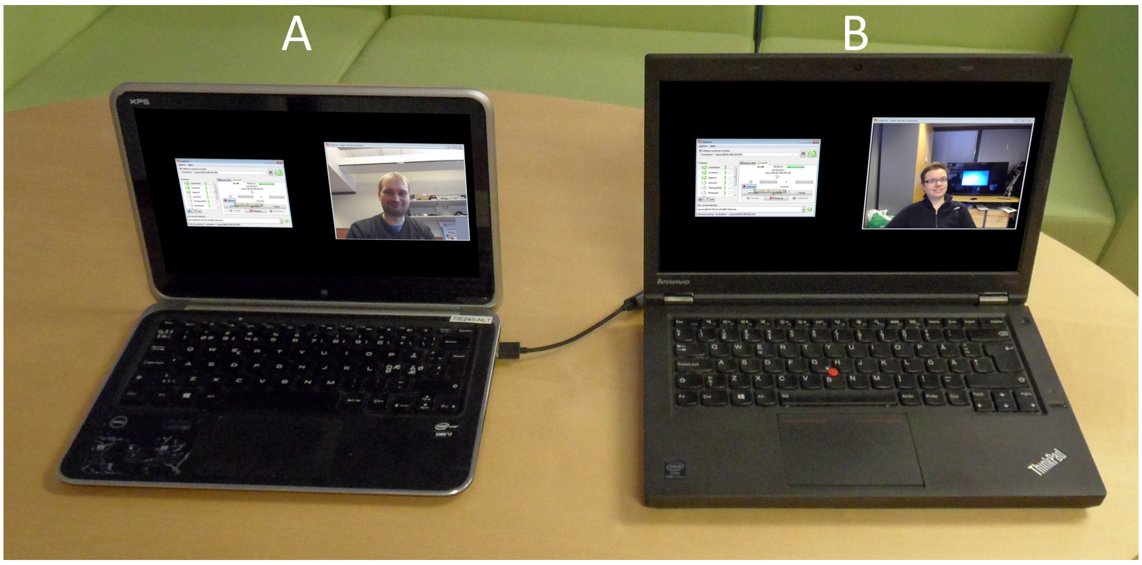

***Figure* 5.1** *Recording screenshot of Linphone.*

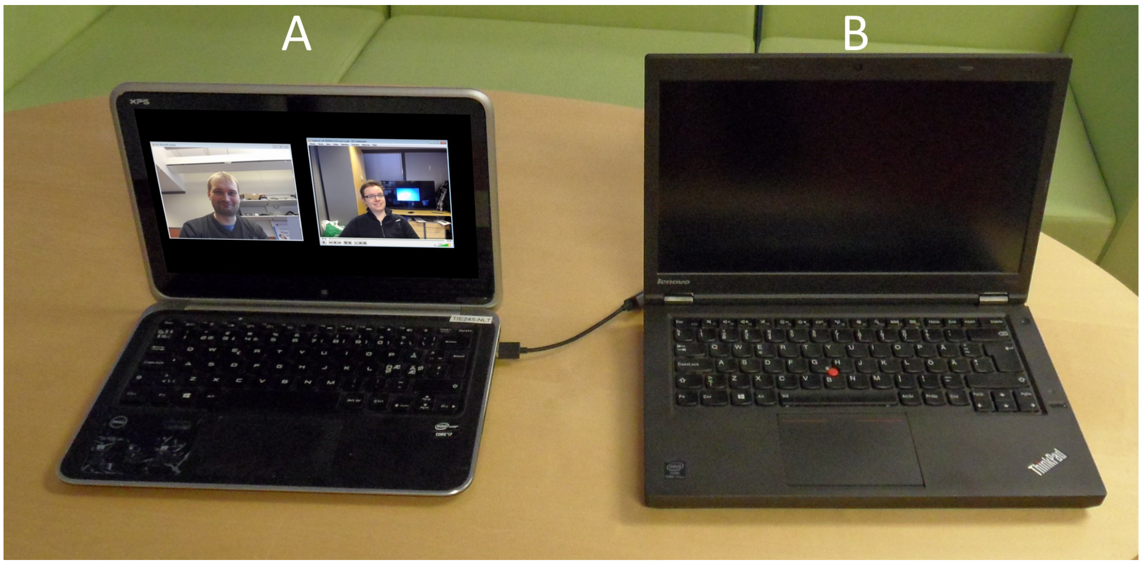

***Figure* 5.2** *Playback of recorded MP4 file using VLC.*

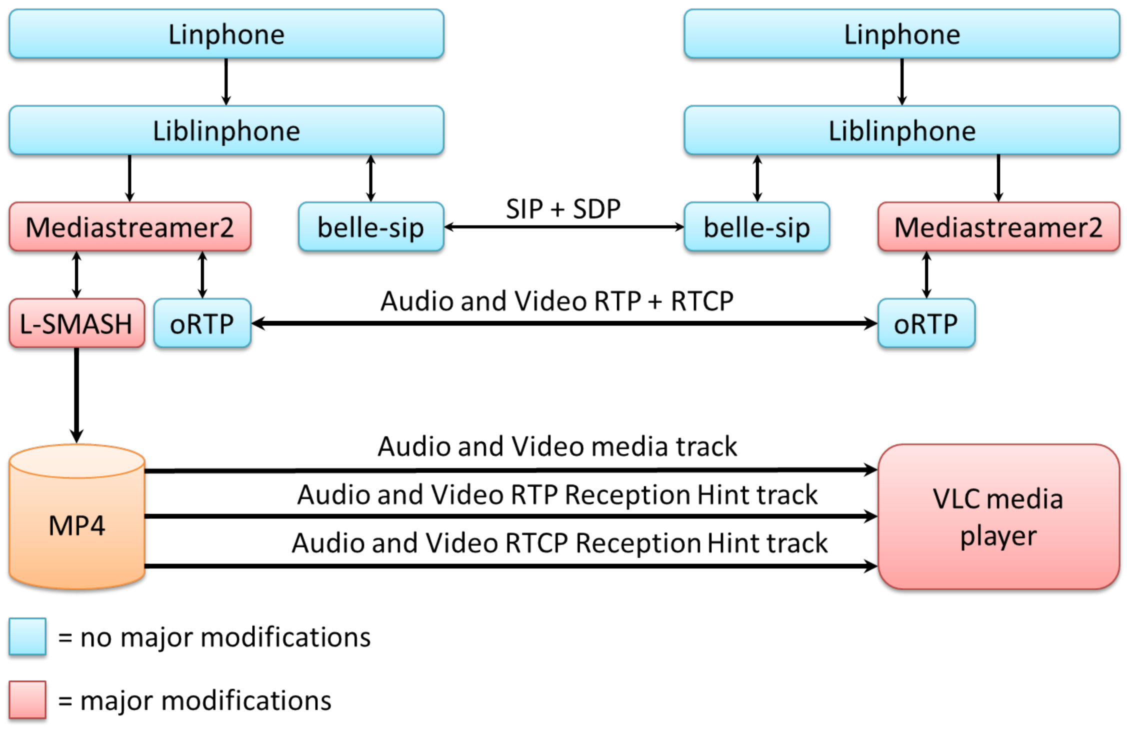

Figure 5.3 *The overall video call system with data flow shown.*

The file recording ends and header data is written when either of the participants presses the record button again or the video call ends. It is also possible to record multiple segments of the same video call, in which case the new segment is recorded into a separate file.

Figure 5.4 shows 'moov'-box (See table 2.8 and figure 2.6) with next level boxes of the created MP4 file. The four character codes for sample entry boxes of each track are shown in parenthesis. The final MP4 file has six tracks: two media tracks, two RTP Reception Hint Tracks, and two RTCP Reception Hint Tracks. GSM and H.264/AVC tracks include the recorded outgoing audio and video streams and RTP Reception Hint include the recorded incoming audio and video RTP streams. Incoming RTCP sender reports are recorded for both incoming audio and video as RTCP Reception Hint Tracks. 'udta'-box holding movie level 'hnti'-box with part of SDP message is also shown in the figure.

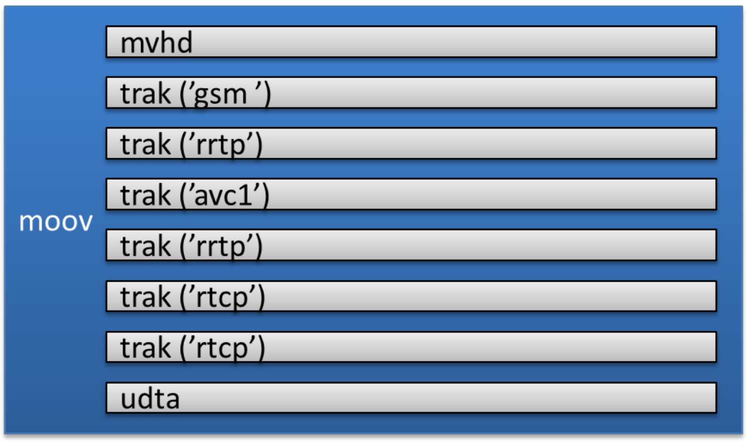

Figure 5.4 *Resulting MP4 moov-box contents.*

5.2 Recording modifications

Before modifications Linphone was not able to record video and the audio was recorded uncompressed. Now the audio is recorded as GSM compressed which takes less space. In the original version, the audio tracks were mixed together, but now the audio tracks are recorded as separate tracks, because of RTP Reception Hint Track can contain only one RTP stream. For recording to work, Linphone call settings have to be changed to use H.264/AVC video format and GSM audio format only. These are not the default options. For decoding of MP4 RTP Reception Hint Track playback in VLC to start immediately with FIR request, AVPF has to enabled in Linphone configuration file.

Linphone and L-SMASH modifications were tested to work on Windows and Linux operating without any major problems. Linphone encompassed about 2000 new lines of code and L-SMASH modifications were about 600 lines of code. The system is capable of generating MP4 files that follow ISOBMFF [4]. Therefore, the system is also used to generate a conformance file for RTP/RTCP Reception Hint Tracks. With these modifications, L-SMASH supports RTP/RTCP Reception Hint Tracks and hint tracks. All the modifications were submitted as a Pull Request to L-SMASH Github repository.

5.3 Playback modifications

VLC automatically recognizes when MP4 file has an RTP Reception Hint track and uses modified MP4 demultiplexer in playing the file. Playback is successful if recorded RTP stream is in GSM, Speex or H.264 format. VLC implementation is tested to work in both Linux and Windows operating systems.

VLC modifications consist of approximately 700 new lines of code. The modified VLC code was updated to VLC nightly version and submitted to VLC main repository as a patch. Patch was accepted and is part of the current VLC nightly version 3.0.0. The inclusion of RTP Reception Hint track in VLC also dissaminates awareness of RTP Reception Hint Tracks.

6. ANALYSIS

6.1 Performance

The performance of RTP Reception Hint Track recording filters were compared to that of audio and video track recording filters in terms of CPU time. Table 6.1 tabulates the results of six runs of Linphone recording using the implemented system (Chapter 5). Recording took place on both computers, i.e., two set of test results were generated from one video call. In total, 3 calls were made, a 1 minute call, a 10 minute call, and a 60 minute call. Both computers record similar video feeds using different cameras (see figure 2.3 and 5.3).

In summary, recording RTP Reception Hint Track takes 1.8 to 2.8 times as much processing power as recording just media tracks. On computer A, the relative CPU usage is larger than that on computer B, because video encoding takes more time on computer B due to more complex video data. The RTP recording filters are slower compared to media recording filters on computer B. This difference could mean that RTP recording filters do better when the media data is less complicated.

Table 6.1 *Comparison of recording filter CPU usage of program CPU usage.*

Computer	Time	Media (GSM & H.264/AVC)	RTP (GSM & H.264/AVC)	
		CPU time	CPU time	Slowdown
A	1 min	0.33 %	0.59 %	1.8x
B	1 min	0.22 %	0.50 %	2.3x
A	10 min	0.36 %	0.87 %	2.4x
B	10 min	0.25 %	0.71 %	2.8x
A	60 min	0.39 %	0.84 %	2.1x
B	60 min	0.24 %	0.56 %	2.4x

RTP or media recording filters are not optimized for speed so no definite conclusion on comparison between media recording and RTP Reception Hint Track recording can be drawn. However, the CPU usage times are less than 1% of total CPU time for both recording filter types so there is no meaningful penalty for recording as RTP Reception Hint Track. A performance log for the first test case is shown in Appendix A. It can be seen that about 82% of program CPU usage comes from encoding H.264/AVC video. Hence, the choice of codec, encoding software and encoding parameters is much more potential source of performance increase than the type of MP4 track chosen for recording.

Using RTP Reception Hint Tracks increases the sample size compared to media tracks. The increase is constant for audio. For video increase is dependent on the number of RTP packets in one video sample. The added sample for RTP Reception Hint Track can be calculated as in equation 6.1. The packetcount field is one for audio samples and is the number of slices for video samples. headersize = 4 B, RTPpacketsize = 12 B and constructorsize = 16 B can be calculated from figures 2.7, 2.8, and 2.9 respectively where each row is four bytes. For GSM audio, there is always one RTP packet per sample increasing the RTP Reception Hint Track sample structure to 4 B + 12 B + 16 B = 32 B. Compared to GSM media track sample size of 33 B, this is an 97% increase so RTP Reception Hint Tracks nearly double the sample size.

$$RTPsamplesize = headersize + (RTPpacketsize + constructorsize) * packetcount \tag{6.1}$$

The video samples are usually so large that the effect of recording as RTP Reception Hint Track is negligible. The size calculations of an example of a video picture containing one inter video picture goes as follows: It has six RTP packets of sizes 1116 B, 1124 B, 1124 B, 1132 B, 1126 B, and 382 B totaling 6004 B. In this case, the offset of RTP Reception Hint Track samples is: 4 B + (12 B + 16 B) * 6 = 172 B increasing the size of this sample by 3%.

For 18 frames per second video excluding intra frames, the bit rate will be 864.6 kbit/s. In Linphone Intra pictures arrive every 10 seconds and were excluded because their combined effect on bitrate is small. The bit rate of GSM stream is 13.2 kbits/s so the the total bit rate of the video call is 864.6 kbit/s + 13.2 kbit/s = 877.8 kbit/s. Since there are 50 audio frames/s in GSM the RTP Reception Hint Track structures account for 18 * 172 B + 50 * 32 B = 37.6 kbit/s, i.e., they increase bit rate by 4.3%. Considering the storage capacity of modern hardware, 4.3% is not a large increase expect in cases where only audio streams are recorded and storage space is an issue.

6.2 Discussion

This Section discusses the process of implementation, merits, and shortcoming of video call system and evaluation of RTP/RTCP Reception Hint Track when compared to plain media tracks.

6.2.1 Process of implementing a video call recording

The development of the video call system took about one year to complete (11 person months). Linphone modifications took around 80% of total development time where as L-SMASH and VLC parts took the remaining 20%. Linphone seemed to suffer from code bloat. This is possibly due to Linphone being old and not having enough refactoring done to it over the years. VLC is also an old project, but has a much more modular design with a separation between VLC core and modules. An estimate of created code would be about 2000 lines of C code for Linphone, 600 lines of C code for L-SMASH, and about 700 lines of C code for VLC.

MinGW was not an ideal platform, because it lacked a visual debugger. Porting Linphone to Visual studio was tried unsuccessfully with a moderate effort. When testing with virtual machine, it was concluded that compiling Linphone with Ubuntu was much easier, faster, and worked without any additional steps. This is an indication that Linphone was developed under Ubuntu or a similar operating system and is therefore most tested to work on it. The compilation of VLC media player was tried on Windows, but the instructions were out of date and Windows compilation was not supported properly. Again, compiling within Linux operating system was much easier. It can be concluded that when making modifications to an existing open-source software, most convenient development environment is the one primarily used by the development team of the software.

6.2.2 Video call system evaluation

Recording a video call enables returning to the session, which can be essential, e.g., in job interviews and business meetings. Furthermore, the implmemented video call system could be improved to provide a better user experience through the following improvements: 1) Implementing SRTP and SRTCP Reception Hint Tracks would enable recording of encrypted video calls. They would make the recording system more applicable to usage scenarios where privacy or security is concerned during the call. 2) The sound quality of the video call could be improved, because the chosen GSM codec is not up to the same standards as modern sound codecs such as Opus [41]. 3) FIR request can be used to determine the point at which the other participant starts recording the video call. This can also be an unwanted side effect. A solution would be to disable FIR request and either use all-intra H.264/AVC stream or leave out the support for immediate video recording.

From user point of view, the lack of video conferencing can be considered as a major shortcoming of the implemented system since other applications such as Skype and Jitsi provide support for it. Implementing a video conferencing system with recording would require one of the following modifications: 1) Implement video conferencing to Linphone; 2) implement video call recording in some VoiP software capable of video conferencing; or 3) implement a complete new video conferencing system capable of recording. The Option 1 would be problematic because of code bloat within Linphone. As of version 3.8.2, Linphone audio conferencing features are implemented as a separate module without any consideration for video conferencing. For the option 2 only suitable open-source software is Jitsi, but it is written in Java, so L-SMASH could not be used without implementing new language bindings for L-SMASH. Hence the option 3 is the recommended one for expanding the video call system to video conferencing system with recording.

Another drawback of the implemented video call system is the ability to only play one audio track at a time. Thus, a convenient usage of the system would require switching between audio tracks every time another person starts talking. Two solutions for that are possible: 1) Implement simultaneous playback of two or more audio tracks within a player; or 2) mixing all audio tracks to one. The solution 2 would require less effort than solution 1, since it was already existing in Linphone and there is no problem adding a third audio track to MP4 file. RTP Reception Hint Tracks are not compatible with the solution 2.

6.2.3 RTP Reception Hint Track evaluation

The main advantage of using RTP/RTCP Reception Hint track is that more information is retained with regards to timestamps during a video call. The following benefits were discovered in usage of RTP/RTCP Reception Hint Tracks: 1) Both resending and playback of the stream are possible; 2) Start offset synchronization and clock drift synchronization are enabled; and 3) A support is provided to all codecs that can be transported with RTP stream.

Different codecs were tested during the development. In addition to GSM codec, the Speex codec was tested and RTP Reception Hint Track recording worked without any modifications to Linphone or L-SMASH, but VLC needed small modifications to recognize a different codec from the SDP message. Hence, new codecs could be added in the proposed system by modifying playback only. For additional codec support, the following steps need to be implemented in VLC: 1) recognize RTP payload type from within SDP message. 2) demultiplexing of sample. For audio, this step is not necessary, but for video slices may need to be combined to a picture.

There are some disadvantages to using RTP/RTCP Reception Hint Tracks. In addition to small size increase the tracks cannot be mixed together at recording stage which can create a problem if media player does not support simultaneous playback. This is the case, e.g., with VLC media player. For better evaluation of RTP/RTCP Reception Hint Tracks, a demo setup that resends recorded RTP/RTCP Reception Hint Tracks should be constructed. Resending would enable such features as remote playback of video calls. For an answering machine could be implemented for VoIP applications.

7. CONCLUSIONS

This Thesis showed an end-to-end video call system that deploys RTP and RTCP Reception Hint Tracks in video call recording and playback. The proposed setup uses Linphone and L-SMASH to record the video call in MP4 format and VLC media player to play the recorded MP4 file. The MP4 file has six tracks, two RTP Reception Hint Tracks, two RTCP Reception Hint Tracks, and two media tracks. GSM was chosen as audio codec and H.264/AVC as video codec. The video call system works on both Windows and Linux operating systems. The new features implemented in this work include recording filters to Linphone as well as support for RTP and RTCP Reception Hint Tracks for L-SMASH and for VLC media player. The changes to VLC media player were included to official VLC nightly repository.

The usage of RTP Reception Hint Tracks in a video call system is feasible in terms of performance and size requirements. The CPU usage of RTP Reception Hint Track recording stayed under 1% of the total CPU usage of the program and the bit rate increase over traditional media tracks was 4% in a typical video call. Video encoding accounts for over 80% of Linphone CPU time.

The proposed proof-of-concept system validates the feasibility and benefits of the RTP/RTCP Reception Hint Track feature, so augmenting a video call capture with them is recommended in RTP based traffic, provided that a compatible player is available. Adding support for playback of RTP/RTCP Reception Hint track is recommended in media players to enable wider usage of RTP Reception Hint track in RTP stream recording. RTP Reception Hint Tracks also meet particularly well the needs of multi-codec implementations where a video call recording and playback can be done with different audio and video codecs.

In the future, the proposed system could be extended with five new features. Firstly, the RTP Reception Hint Tracks could be used to detect possible packet losses during recording. Secondly, an answer machine for video calls could be implemented by allowing a sender to re-create and send the original RTP streams. Thirdly, the

RTCP Reception Hint Tracks could be used to correct potential clock drift during playback by using the NTP timestamps in RTCP sender reports. The listener could then choose whether to play the file as it was received or as it was sent. Fourthly Linphone could be extended to include video conferencing. Finally RTP Reception Hint Track could be switched to SRTP Reception Hint Track and RTCP Reception Hint Track to SRTCP Reception Hint Track for enabling the usage of encryption within a video call.

BIBLIOGRAPHY

[1] S. Casner, R. Frederick, V. Jacobson, and H. Schulzrinne, RFC 3550, RTP: A Transport Protocol for Real-Time Applications, Network Working Group, IETF, July 2003, 104 p, [Online]. Available: https://tools.ietf.org/html/rfc3550

[2] Skype, Skype Technologies, Microsoft Corporation, [Online]. Available: http://www.skype.com/

[3] How can I record my Skype calls? Skype Help, [Online]. Available: https://support.skype.com/en/faq/FA12395/how-can-i-record-my-skype-calls

[4] ISO Base Media File Format, document ISO/IEC 14496-12 and ISO/IEC 15444-12, ISO/IEC, Jul 2012.

[5] Linphone: Open-source VOIP project, Belledonne communications, [Online]. Available: http://www.linphone.org/

[6] L-SMASH, [Online]. Available: http://l-smash.github.io/l-smash/

[7] VLC media player, VideoLAN, [Online]. Available: http://www.videolan.org/vlc/index.html

[8] WhatsApp, WhatsApp Inc., Facebook, [Online]. Available: https://www.whatsapp.com/

[9] Edison's Telephonoscope (TRANSMITS LIGHT AS WELL AS SOUND), PUNCH'S ALMANACK FOR 1879, Terramedia, December 1878, [Online]. Available: http://www.terramedia.co.uk/Chronomedia/years/Edison_Telephonoscope.htm

[10] A Missing Link in the History of the Videophone, VSee, June 2011, [Online]. Available: https://vsee.com/blog/a-missing-link-in-the-history-of-the-videophone/

[11] Television over the Telephone Sends Images of Speakers, Modern Mewchanix, October 1938, [Online]. Available: http://blog.modernmechanix.com/television-over-the-telephone-sends-images-of-speakers/

[12] S. Lingfen, I.-H. Mkwawa, E. Jammeh, and E. Ifeachor, Guide to Voice and Video over IP: For Fixed and Mobile Networks, Springer, 2013.

[13] G. Camarillo, M. Handley, A. Johnston, J. Peterson, J. Rosenberg, H. Schulzrinne, and R. Sparks, RFC 3261, SIP: Session Initiation Protocol, Network Working Group, IETF, 269 p, [Online]. Available: `http://tools.ietf.org/html/rfc3261`

[14] M. Handley, V. Jacobson, and C. Perkins, RFC 4566, SDP: Session Description Protocol, Network Working Group, IETF, July 2002, 269 p, [Online]. Available: `https://tools.ietf.org/html/rfc4566`

[15] L. Peterson and B. Davie, Computer Networks: A System Approach, 4th edition, Morgan Kaufman, 835 p, April 2007.

[16] Skype has over 300 million monthly active users, Microsoft announces at Build 2016, Windows Report, [Online]. Available: `http://windowsreport.com/skype-number-of-users/`

[17] Skype of Niklas Zennström and Janus Friis, History of Computers, [Online]. Available: `http://history-computer.com/Internet/Conquering/Skype.html`

[18] WebRTC, The WebRTC initiative, [Online]. Available: `https://webrtc.org/`

[19] Real-Time Communication in WEB-browsers (rtcweb), IETF, [Online]. Available: `https://datatracker.ietf.org/wg/rtcweb/documents`

[20] WebRTC 1.0: Real-time Communication Between Browsers, W3C, May 2016, [Online]. Available: `https://www.w3.org/TR/webrtc/`

[21] Empathy, The GNOME project, [Online]. Available: `https://wiki.gnome.org/Apps/Empathy`

[22] Ekiga, [Online]. Available: `http://www.ekiga.org/`

[23] Homer Conferencing, Integrated Communication Systems Group, Ilmenau University of Technology, [Online]. Available: `http://www.homer-conferencing.com`

[24] Jitsi, Jitsi Team, [Online]. Available: `https://jitsi.org/`

[25] Advanced video coding for generic audiovisual services, 02/2014, ITU-T, 2014, 766 p, [Online]. Available: http://www.itu.int/rec/T-REC-H.264-201402-S

[26] ISO/IEC 14496-10, Information technology - Coding of audio-visual objects - Part 10: Advanced Video Coding, 2014.

[27] Digital cellular telecommunications system (Phase 2+)(GSM); Full rate speech; Transcoding, ETSI, November 2000, 64 p, [Online]. Available: http://www.etsi.org/deliver/etsi_en/300900_300999/300961/08.01.01_60/en_300961v080101p.pdf

[28] GSM 06.10 lossy speech compression, [Online]. Available: http://www.quut.com/gsm/

[29] The GTK+ project, The GTK+ Team, [Online]. Available: http://www.gtk.org/

[30] M. Baugher, E. Carrara, D. McGrew, M. Naslund, and K. Norrman, RFC 3711, The Secure Real-time Transport Protocol (SRTP), Network Working Group, IETF, march 2004, 56 p, [Online]. Available: https://tools.ietf.org/html/rfc3711

[31] B. Burman, U. Chandra, S. Wenger, and M. Westerlund, RFC 5104, Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF), Network Working Group, IETF, February 2008, 64 p, [Online]. Available: https://tools.ietf.org/html/rfc5104

[32] x264, VideoLAN, [Online]. Available: https://www.videolan.org/developers/x264.html

[33] Data Encapsulation, Protocol Data Units (PDUs) and Service Data Units (SDUs), The TCP/IP Guide, [Online]. Available: http://www.tcpipguide.com/free/t_DataEncapsulationProtocolDataUnitsPDUsandServiceDa.htm

[34] R. Even, R. Jesup, T. Kristensen, and Y.-K. Wang, RFC 6184, RTP Payload Format for H.264 Video, IETF, May 2011, 101 p, [Online]. Available: https://tools.ietf.org/html/rfc6184

[35] C. Burmeister, J. Ott, J. Rey, N. Sato, and S. Wenger, RFC 4585, Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF), Network Working Group, IETF, February July 2006, 51 p, [Online]. Available: https://tools.ietf.org/html/rfc4585

[36] S. Döhla, M. Hannuksela, and K. Murray, The DVB file format, IEEE Signal Processing Magazine, volume 29, no. 2, March 2012, pp. 148-153.

[37] A. Butterfield and G. Ngondi, "multiplexing", A Dictionary of Computer Science, Oxford University Press, 2016, [Online]. Available: `http://www.oxfordreference.com/view/10.1093/acref/9780199688975.001.0001/acref-9780199688975-e-3394`.

[38] Vimeo: Make life worth watching, IAC/InterActiveCorp, [Online]. Available: `https://vimeo.com/`

[39] Media Player Classic - Home Cinema, [Online]. Available: `https://mpc-hc.org/`

[40] MinGW: Minimalist GNU for Windows, MinGW Project, [Online]. Available: `http://www.mingw.org/`

[41] T. Terriberry, JM. Valin, and K. Vos, RFC 6716, Definition of the Opus Audio Codec, IETF, September 2012, 326 p, [Online]. Available: `http://tools.ietf.org/html/rfc6716`

APPENDIX A. LINPHONE FILTER STATISTICS AFTER 1 MINUTE VIDEO CALL

```
============================================================
                  FILTER USAGE STATISTICS
Name                Count     Time/tick (ms)      CPU Usage
------------------------------------------------------------
MSX264Enc           2350      39,7414             81,8703
MSDrawDibDisplay    4425      2,20493             8,55137
MSH264Dec           13862     0,412104            5,00605
MSWinSndWrite       3476      0,684211            2,08461
MSRtpSend           27766     0,0341412           0,83069
MSRTPRec            9388      0,0719991           0,592349
MSGsmEnc            3476      0,112166            0,34174
MSVideoRec          2350      0,110166            0,22695
MSRtpRecv           27766     0,00504196          0,122676
MSAudioRec          2488      0,0470068           0,102522
MSPixConv           2350      0,0455125           0,0937593
MSSpeexEC           7804      0,0094811           0,0648429
MSAudioMixer        27808     0,000934949         0,0227826
MSGsmDec            6909      0,00274964          0,0166488
MSTee               10655     0,00168919          0,0157726
MSJpegWriter        4829      0,00351967          0,0148963
MSVolume            10386     0,00154039          0,0140201
MSWinSndRead        13904     0,000791082         0,00963881
MSDsCap             13862     0,00064921          0,0078863
MSDtmfGen           14185     0,000422952         0,00525753
MSFilePlayer        14185     0,000211476         0,00262877
MSSizeConv          2350      0,000425351         0,000876255
MSGenericPLC        13904     7,19166e-005        0,000876255
MSEqualizer         6910      0,000144697         0,000876255
MSResample          0         0                   0
```