SCALABLE WIRELESS VIDEO STREAMING OVER REAL-TIME PUBLISH SUBSCRIBE MIDDLEWARE

BY

MOHAMMED ABDULJALIL MOHAMMED AL-SAEDI

A Thesis Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER NETWORKS

OCT 2013

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **MOHAMMED ABDULJALIL MOHAMMED AL-SAEEDI** under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER NETWORKS**.

Thesis Committee

[illegible]
Dr. Basem Al-madani (Adviser)

(Co-adviser)

[illegible]
Dr. Tarek Sheltami (Member)

[illegible]
Dr. Zubair Baig (Member)

Dr. (Member)

[illegible]
Dr. Basem Al-madani
Department Chairman

[illegible]
Dr. Salam A. Zummo
Dean of Graduate Studies

عمادة الدراسات العليا
DEANSHIP OF GRADUATE STUDIES

29/10/13
Date

DEDICATION

To my parents who have been my constant source of inspiration

ACKNOWLEDGMENTS

Apart from the efforts of myself, the success of any project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project. I would like to show my greatest appreciation to Dr. Basem Al-madani. I can't say thank you enough for his support and help. Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Tarek Sheltami and Dr. Zubair Baig, for their encouragement and insightful comments. I would also like to thank the Real-Time Infrastructure Software company (RTI) for their support with all tools that I have used in this thesis work. Spacial thanks are given to the Departement of Computer Engineering of King Fahd University of Petroleum and Minerals. The guidance and support received from all the members who contributed and who are contributing to this project, was vital for the success of the project. I am grateful for their constant support and help.

Last but not the least important, I owe more than thanks to my family for their support and encouragement throughout my life

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

RTPS	Real-Time Publish Subscribe
RTP	Real-Time Protocol
DDS	Data Distribution Service
CORBA	Common Object Request Broker Architecture
OMG	Object Management Group
DCPS	Data-Centric Publish-Subscribe
GDS	Globle Data Space
QoS	Quality of Service
AVC	Advance Video Codec
SVC	Scalable Video Codec
NAL	Network Abstract Layer
VCL	Video Coding Layer
GOP	Group of Pictures
SNR	Signal to Noise Ratio
PSNR	Peak-to-Peak Signal-to-Noise Ratio
TFRC	TCP-Friendly Rate Control

FEC	Forward Error Correction
VBR	Variable Bit Rate
ARQ	Automatic Repeat Request
VoD	Video on-Demand
CODEC	COder-DECoder or COmpressor-DECompressor
ME	Motion Estimation
MCP	Motion-Compensated Prediction
MPEG	Moving Pictures Expert Group
UEP	Unequal Error Protection
ULP	Unequal Layer Protection
IDR	Instantaneous Decoding Refresh
MIMO	Multi-Input Multi-Output
MCMI	Multi-Channel Multi-Interface
MAD	Mean Absolute Difference
SSIM	Structural SIMilarity index
MSE	mean Squared Error
MOS	Mean Opinion Score

THESIS ABSTRACT

NAME:	Mohammed Abduljalil Mohammed Al-saeedi
TITLE OF STUDY:	SCALABLE WIRELESS VIDEO STREAMING OVER REAL-TIME PUBLISH SUBSCRIBE MIDDLEWARE
MAJOR FIELD:	COMPUTER NETWORKS
DATE OF DEGREE:	Oct 2013

Enabling Real-Time video streaming over wireless networks is a challenging task due to the time-varying channel conditions and the limited network resources. The instability of wireless networks leads to problems such as limited and time-varying bandwidth, and unexpected traffic congestion when transmitting a burst of video streams. As a result, the transmitted video packets are exposed to be delayed or dropped. However, in Real-Time video streaming, each frame must be delivered and decoded by its playback time. In other words, any frame that is retransmitted due to loss in transmission or late arriving is considered a useless frame if its decoding and display deadline is too late to be displayed. Therefore, efficient Real-Time video streaming requires an efficient video quality of service (QoS) transmission control mechanism to adapt to the time-varying network changes. Traditional ap-

proaches of Real-Time video streaming focused on adapting the video encoder bit rate to the available network resources. Other approaches focused at the level of network protocols and link layer adaptation to the source video streaming rate. Recently, layer coding (LC) has enabled Real-Time and scalable video streaming to clients of heterogeneous capabilities by dropping upper enhancement layers without the need of re-encoding. However, layer coding still facing unfair layer protection problem in which packets from the base or lower layers might be dropped while there is a chance to drop packets from the upper enhancement layers. Moreover, the reception of the base layer bitstream is always required for at least decoding the base quality. Thus, loosing packets from the base layer can significantly affect the delivered video quality and sometimes lead to an interruption especially in error-prone networks such as wireless networks. Architectural solutions at the middleware level introduce higher flexibility, more efficiency in development time and more QoS control. In this research, I investigate the behaviour of video streaming over Real-Time publish-subscribe based middleware. I propose and develop an unequal layer protection mechanism for Real-Time video streaming based on the Data Distribution Service (DDS) middleware, and show the performance of my approach over IEEE 802.11g WLAN networks. My approach shows a graceful degradation of video quality while maintaining a robust video streaming free of visible error or interruptions.

ARABIC ABSTRACT

ملخص الرسالة

الاسم:	محمد عبدالجليل محمد السعيدي
عنوان الرسالة:	تدفق الفيديو المتحجم لاسلكيا في الوقت الحقيقي باستخدام وسيط النشر والاشتراك
التخصص:	شبكات الحاسب الآلي
تاريخ التخريج:	اكتوبر ٢٠١٣

تمكين تدفق الفديو في الوقت الحقيقي عبر الشبكات اللاسلكية هي مهمة صعبة نظظرا لظظروف قناة الارسال المتفاوتة خلال الوقت وكذلك موارد الشبكة المحدودة . عدم استقرار الشبكات اللاسلكية يؤدي إلى مشاكل مثل عرض النطاق الترددي يكون محدود و متفاوت خلال الوقت، و الازدحام الغير متوقع عندما يتم ارسال وابلا من الفيديو المتدفق. ونتيجة لذلك ، تتعرض حزم الفيديو المنقولة للتأخير أوالفقدان . ومع ذلك ، فان من اجل تدفق الفديو في الوقت الحقيقي، يجب أن يتم توصيل كل صوره وفك ترميزها وعرضها خلال الوقت المخصص لعرضها عند المستقبل . وبعبارة أخرى ، أي صوره تم اعادة ارسالها بسبب فقدانها خلال الارسال او وصولها بشكل متاخر تعتبر عديمة الفائدة اذا كان الموعد النهائي لفك التشفير و العرض الخاص بها بعد فوات الأوان . لذلك، فان كفاءة تدفق الفيديو في الوقت الحقيقي يتطلب وجود نوعية خدمة فعالة لارسال الفديو كما يتطلب وجود آلية تحكم للتكيف مع التغيرات المتفاوتة في الشبكة خلال الوقت. ركزت الطرق التقليدية لتدفق الفيديو في الوقت الحقيقي على تكييف ترميز معدل بت للفيديو إلى موارد الشبكة المتوفرة . كما ركزت الطرق الأخرى

على مستوى بروتوكولات الشبكة و تعديلها لتتكيف مع معدل تدفق الفديو من المصدر . في الآونة الأخيرة ، طبقات الترميز (ظال سي) استطاعت ان تمكن تدفق الفديو في الوقت الحقيقي للمستقبلين ذوي الامكانيات غير المتجانسة من خلال إسقاط بعض طبقات التعزيز العليا من دون الحاجة إلى إعادة ترميز. ومع ذلك ،فان طبقات الترميز لا تزال تواجه مشكلة عدم حماية الطبقات الاكثر اهمية حيث انه يمكن إسقاط حزم من القاعدة أو الطبقات السفلى في حين أن هناك فرصة لإسقاط حزم من طبقات التعزيز العليا الاقل اهمية. وعلاوة على ذلك ، هناك حاجة دائمة لاستقبال طبقة القاعدة الاساسية على الأقل لفك ترميز فديو باقل جودة . وبالتالي ، فقدان الحزم من الطبقة الأساسية يمكن أن يؤثر تأثيرا كبيرا على جودة الفيديو المرسل و قد يؤدي في بعض الأحيان إلى انقطاع الفديو وخاصة في الشبكات المتعرضة للاخطاء مثل الشبكات اللاسلكية . الحلول على مستوى الوسيط توفر أعلى مرونة ، وأكثر كفاءة من حيث الوقت اللازم للتطوير و يوفر مزيد من التحكم بجودة الخدمة . في هذا البحث ، يتم التحقيق عن سلوك تدفق الفيديو في الوقت الحقيقي عبر استخدام وسيط النشر والاشتراك .حيث تم اقتراح وتطوير آلية حماية غير متكافئة لطبقات الفيديو لتدفق الفديو في الوقت الحقيقي على أساس استخدام خدمة توزيع البيانات (دي دي اس) الوسيطة ، واختبار أداء الطريقة المقترحة عبر شبكات الواي فاي من نوع ٨٠٢.١١ جي . وقد بينت النتائج انحسار رشيق في جودة الفيديو مع الحفاظ على تدفق مستمر للفيديو خالي من الاخطاء المرئية أو الانقطاع.

CHAPTER 1

INTRODUCTION

The flexibility and low infrastructure requirements that wireless networks offer to customers increase its popularity among users. More demands have been recently turned into delivering Real-Time video over wireless networks. However, Real-Time video streaming over wireless networks has been addressed with data delivery and communication challenges.

The time-varying changes in the wireless channel that can occur due to interference, fading, and mobility make video streaming over such networks a challenging task [8]. Therefore, provisioning of video streaming end-to-end Quality of Service (QoS) is required for maintaining a continuous video playback in Real-Time multimedia applications, e.g., video conferencing. Video streaming is often described as *bursty* since video is basically a collection of frames sequence transmitted in a particular frame rate. The video frame cannot be decoded or played out at the receiver side until all or most of its transmitted packets are received on time [9]. Although, several schemes have been proposed to adapt video encoder bit rate to the available wireless network resources, or to adapt the wireless network architecture and protocols to the generated video encoder bit rate, the ability of these schemes to deliver a continuous Real-Time video streaming is limited.

1.1 Problem Description

In Real-Time video streaming, each frame must be delivered and decoded by its playback time. Therefore, any data that is retransmitted due to loss in transmission or late arriving is considered as useless if its decoding and display deadline is too late to be displayed. Real-Time video streaming over wireless networks is difficult because the transmitted packets are exposed to the time-varying bandwidth, delay jitter, or high packet loss rate. Reliable delivery of high-quality video over wireless networks which are dealing with unknown and dynamic bandwidth, delay jitter and loss rate is a wide research area in Real-Time video streaming.

The bandwidth available between two points in a wireless network is generally unknown and time-varying. If the sender transmits faster than the available bandwidth, this will result in congestion that may lead to a severe drop in video quality due to packets loss. If the sender transmits slower than the available bandwidth then the receiver produces sub-optimal video quality.

The end-to-end delay that a packet experiences may fluctuate from packet to packet leading to a delay jitter problem. However, the receiver must receive, decode, and display frames at a constant rate, and any late frames resulting from the delay jitter can produce problems in the reconstructed video, e.g. jerks. This problem is typically addressed by including a playout buffer at the receiver but with additional delay which is not appropriate for Real-Time applications.

In addition to the delay jitter, packet loss is considered as a fundamental problem in Real-Time video streaming over error-prone networks. Wireless channels are typically afflicted by bit errors or burst errors. The packet-loss problem may lead to serious video quality degradation, which affects not only the quality of current frame, but also leads to error propagation to subsequent frames due to the use of the Motion Compensation Prediction (MCP) mechanism [10, 11]. Moreover, a single bit error in a video bitstream with a variable-length coding (VLC) may cause the decoder to loss the synchronization, and consequently the successive correctly received bits become useless [11].

Real-Time video streaming over a time-varying bandwidth wireless channel requires to accurately estimate the available bandwidth and meanwhile adapt the transmitted encoded video bit rate to the estimated channel bandwidth. It also requires to solve this problem in a multicast situation where a single sender streams data to multiple receivers of different available bandwidth capacities. Traditional approaches to Real-Time video streaming focused on adapting the video encoder bit rate to the available network resources. Other approaches focused at the level of network protocols and link layer adaptation (e.g., cross-layer approach) to the source video streaming rate. Recently, layer coding (LC) or as formaly called

Scalable Video Coding (SVC) [12, 13] has enabled Real-Time and scalable video streaming to clients of heterogeneous capabilities by dropping upper enhancement layers without the need of re-encoding. However, layer coding still faces unfair layer protection problem in which packets from the base or lower layers might be dropped while there is a chance to drop packets from the upper enhancement layers. Moreover, bitstream layers are not fully independent since a particular layer requires the presence of all lower layers and the reception of the bit-stream's base layer is always required for at least decoding the base quality. Thus losing packets from the base layer can significantly affect the delivered video quality and sometimes lead to an interruption especially in error-prone networks. Architectural solutions at the middleware level introduce higher flexibility, more efficiency in development time and more QoS control.

In this research I investigate the behaviour of video streaming over Real-Time publish-subscribe based middleware. I propose and develop an unequal layer protection mechanism for Real-Time video streaming based on the Data Distribution Service (DDS) middleware [5], and show the performance of my approach over IEEE 802.11g WLAN networks. My approach shows a graceful degradation of video quality while maintaining a robust video streaming free of visible error or interruptions. The rest of the research is organized as follows. Chapter 2 presents an overview background about video streaming and publish-subscribe communication model. Chapter 3 presents the literature review. Chapter 4 presents the design and implementation of my proposed scalable RTPS-based video streaming approach. Chapter 5 demonstrates the experimental setup and evaluation results discussion. Finally, my conclusion is given in Chapter 6.

CHAPTER 2

BACKGROUND

2.1 Basics of Video Streaming

Video streaming is a way of transmitting video content in a compressed form over the network to be displayed by the viewer on-demand or as a live show. The media is sent in a continuous stream of data to let users play the video as it arrives instead of waiting for transmitting the entire video file. Video streaming is classified into two types [14]:

On-demand streaming: Video streams are often saved in a permanent storage for extended amount of time, so users can play a video at their own pace and may seek new positions during the playback.

Live streaming: All the users play a video simultaneously with the video source and download the video at its playback rate.

Live video streaming is restricted to the playback deadline of each packet and therefore it requires to transmit video packets with a delivery rate as high as possible and within the delay constraints. The playback delay should not exceed a few seconds for an application to be called live streaming. In live video streaming, source video should be captured, encoded, transmitted, and decoded in Real-Time which requires more computational processes and network resources. Interactive applications are one example of applications that require Real-Time video streaming, e.g., video conferencing, videophone, or interactive games.

On the other hand, on-demand video streaming allows users to download the video at a rate higher or less than the video's playback rate since it does not require Real-Time encoding constraints. In on-demand video streaming, the video source is pre-encoded and stored for later viewing. Therefore, it enables more efficient encoding such as multi-pass encoding that is typically performed for DVD content. However, pre-encoded video provides limited flexibility in terms of adapting to channels that support different bit rates or clients that support different display capabilities than those used in the original encoding devices. Moreover, it is not suitable for live video streaming since it requires more delay for the pre-encoding process. Video-on-demand (VoD), and video streaming over the Internet (e.g., RealNetworks and Microsoft Windows Media), are two examples of on-demand video streaming applications.

The basic purpose of video streaming is to overcome the problems associated with file download since video generally has very large files that require long download times and large storage spaces. The basic idea of video streaming is to split the video into parts, transmit these parts in succession, and enable the receiver to decode and playback the video as these parts are received; without having to wait for the entire video to be delivered. Video streaming conceptually consists of three main processes: first, partition the compressed (encoded) video into packets, then start delivery of these packets, and finally decode and playback those partitions at the receiver while the video is still being delivered. Video streaming provides a number of benefits including low delay before start of video play, and low storage requirements since only a small portion of the video is stored at the client at any point in time [14].

2.2 Video CODEC

Video CODEC (COder-DECoder or COmpressor-DECompressor) is hardware or software that compresses (encodes) digital data (e.g., video or audio) and uncompresses (decodes) the data back to its original form. Video compression is used for reducing the size of data that is going to be transmitted in order to save network bandwidth and storage space. The compression of a video is achieved by exploiting the similarities or redundancies that exist in a typical video signal or frames. For example, consecutive frames in a video frame sequence show some redundancy since they typically contain the same objects, especially in less motion videos. Moreover, video compression is used to only code video features that are perceptually important and not to waste valuable bits for information that is not perceptually important or irrelevant [14].

An ordinary video stream consists of a sequence of video frames or images. Each frame may be coded as a separate image, for example by independently applying JPEG-like coding to each frame. However, since consecutive video frames are typically similar, much higher compression can be achieved by exploiting the similarity between frames. Currently, the most effective approach to exploit similarity between frames is by coding a given frame by first predicting it based on a previously coded frame, and then coding the error in this prediction. Consecutive video frames typically contain the same imagery, however possibly at different spatial locations because of motion. Therefore, to improve the predictability it is important to estimate the motion between the frames and then to form an appropriate prediction that compensates for the motion. The process of estimating the motion between frames is known as Motion Estimation (ME), and the process of forming a prediction while compensating for the relative motion between two frames is referred to as Motion-Compensated Prediction (MCP).

There are three basic common types of coded frames: (1) intra-coded frames, or I-frames, where the frames are coded independently of all other frames, (2) predictively coded, or P-frames, where the frame is coded based on a previously coded frame, and (3) bi-directionally predicted frames, or B-frames, where the frame is coded using both previous and future coded frames. Figure 2.1 illustrates the different coded frames and prediction dependencies for an example MPEG Group of Pictures (GOP). The selection of prediction dependencies between frames can have a significant effect on video streaming performance, i.e. in terms of compression efficiency and error resilience. Currently there are two families of video compres-

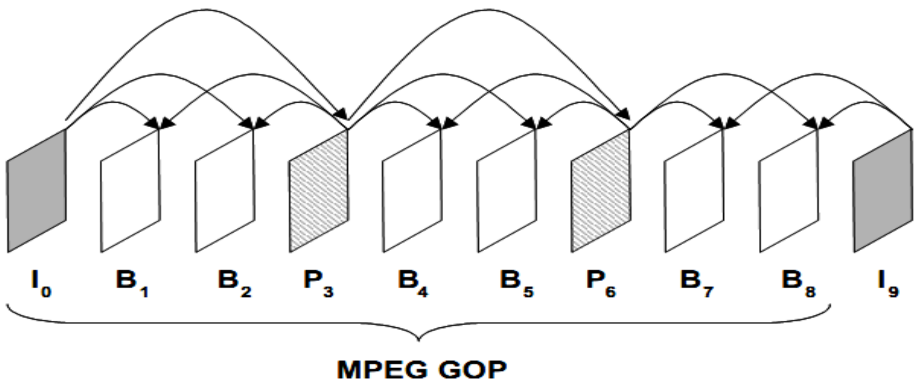


Figure 2.1: Example of the prediction dependencies between frames.

sion standards, performed under the sponsor of the International Telecommunication Union-Telecommunications (ITU-T) and the International Organization for Standardization (ISO). The first video compression standard to gain widespread acceptance was the ITU H.261 [15], which was designed for video conferencing over the integrated services digital network (ISDN). The video compression part of the standard is H.263 and its first phase was adopted in 1996 [16]. Continuous enhancement on H.263 was accomplished till a completely new algorithm originally referred to as H.26L which is currently being finalized as H.264/AVC [17]. Another standard called Moving Pictures Expert Group (MPEG) was established by the ISO to develop a standard for compression moving pictures (video) and associated audio on digital storage media (CD-ROM). Four versions of this standard MPEG-1, MPEG-2, MPEG-3, and MPEG-4 have been published. MPEG-4 was designed to provide improved compression efficiency and error resilience features, as well as increased functionality, including object-based processing, integration of both natural and synthetic (computer generated) content, and content-based interactivity [18]. Currently, the video compression standards H.264/MPEG-4 Part

10 AVC/SVC are primarily used for video communication and video streaming and have gained wide acceptance.

2.3 IEEE 802.11 Networks

To support communication over Wireless Local Area Networks (WLANs), the IEEE 802.11 standard provides specifications for both the physical layer and the Medium Access Control (MAC) sublayer [19, 20, 21, 22, 23]. Different versions of the standard are varied in terms of their choice of modulation and frequency bands at the physical layer. 802.11a and 802.11g use Orthogonal Frequency-Division Modulation (OFDM) and use spectrums centered at 5 GHz and 2.4 GHz respectively [19, 20]. On the other hand, 802.11b uses Direct Sequence Spectrum Spreading (DSSS) and operates at the 2.4 GHz Industrial, Scientific, and Medical (ISM) band [22].

Due to the increasing popularity of Real-Time voice and video traffic over wireless LANs, the 802.11e protocol has been developed to address the growing need for Quality-of-Service (QoS) support [23]. The Enhanced Distributed Channel Access (EDCA) scheme allows traffic classification at the MAC layer, and serves different traffic categories differently according to their priority levels by tuning their channel access parameters: CWmin (minimum Contention Window) and AIFSN (Adaptive Inter Frame Space Number). Despite the enhancements introduced by 802.11e, supporting QoS over 802.11 networks remains a challenging problem.

New standardization efforts for 802.11n are devoted to increasing both the date rate and throughput in wireless LANs [22]. The IEEE 802.11n amendment promises transmission rates up to 600 Mbps by applying Multiple-Input-Multiple-Output (MIMO) technology across multiple antennas and bonding multiple frequency channels for transmission. The amendment is also designed to reduce MAC-layer overhead by aggregating transmissions of multiple consecutive packets, thereby improving throughput of payload data.

2.4 Video Streaming Scalability Overview

Scalable video streaming are classified into two streaming approaches: switching among multiple pre-encoded non-scalable bit-streams and streaming with a single scalable bit-stream. Unlike single scalable video streaming, pre-encoded non-scalable video streaming is not suitable for Real-Time applications since it introduces more encoding delay. The idea of scalable video streaming has been proposed to overcome the problems of unknown and time-varying channel bandwidth, delay jitter, and packet loss that have been addressed in video streaming. The idea of scalable video streaming aims to adjust the amount of data to be transmitted according to the time-varying channel bandwidth [2].

According to the above scalable video streaming classification, there are two methods to compress the video signals: pre-encoded non-scalable video coding and scalable video coding. In the pre-encoded non-scalable video coding, the video content is encoded independent of the actual channel characteristics to a variety of bit-streams. The main problem with this method is that it is difficult to adaptively stream non-scalable video contents to an unknown and time-varying bandwidth channel with heterogeneous client terminals. On the other hand, video in the scalable video coding needs to be encoded only once. Then simply transmit a certain stack of layers (sub-stream) truncated from the main encoded bit-stream. Lower qualities, spatial resolutions and/or temporal resolutions could be obtained adaptively to the actual channel characteristics. As an ultimate goal, the scalable representation of video should be achieved without impact on the coding efficiency. That is, the truncated scalable stream (at lower rate, spatial and/or temporal resolution) should produce the same reconstructed quality as a single-layer bit-stream in which the video was coded directly under the same conditions and constraints, notably with the same bit-rate. However, practically all scalable video coders suffer loss in compression efficiency relative to state-of-the-art non-scalable coders [2].

2.4.1 Non-Scalable H.264/MPEG4 AVC Video Coding

H.264/MPEG4 AVC is a video coding standard that was developed by the Joint Video Team (JVT) of the ITU-T Visual Coding Experts Group (VCEG) and the ISO/IEC Moving Pictures Experts Group (MPEG) [24]. The structure of H.264/AVC constructs of two main layers: Video Coding Layer (VCL) which is designed to efficiently represent the video content, and Network Abstraction Layer (NAL) which formats the VCL representation of the video and provides header information in a manner appropriate for mapping VCL data to a variety of transport layers such as RTP/IP or storage media [25] (see Figure 2.2). In H.264/AVC

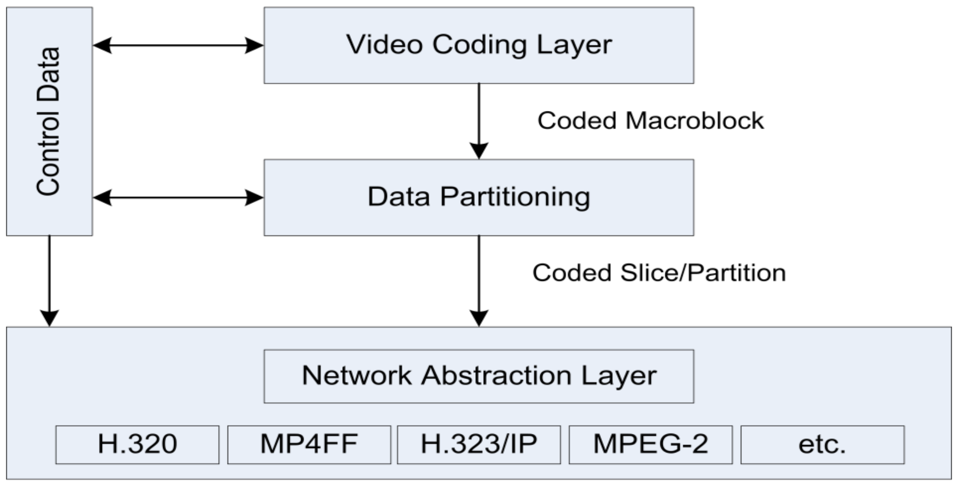


Figure 2.2: Structure of H.264/AVC video encoder [2].

the coded video data is organized into NAL units, which is basically a packet of integer numbers of bytes. The first byte of each NAL unit is a header byte that contains an indication of the type of content data, and the remaining bytes contain payload data of the type indicated by the header. The NAL unit is used for encapsulating the encoded video data and allows tagging each packet with an identifier that can later be used by the delivery scheme for enforcing unequal error protection (UEP) or preferential treatment [26]. A set of successive NAL units with specific properties is referred to as an access unit. One access unit is decoded exactly to one decoded picture. Thus, a set of consecutive access units with certain properties is referred to as an encoded video frame sequence. A coded video frame sequence represents an independently decodable part of a NAL unit bit-stream. This bit-stream always starts with an instantaneous decoding refresh (IDR) access unit, which signals that the IDR access unit and all following access units can be decoded without decoding any previous pictures of the bit-stream.

For streaming NAL units, H.264/AVC specification defines a byte stream format in which each NAL unit is prefixed by a specific pattern of three bytes called a start code prefix. The start code prefix helps in identifying the boundaries of the NAL unit. In some cases, streaming the NAL unites can be done without using the start code prefix and instead they can be carried in data packets framed by the system transport protocol as in RTP-based systems [2].

The way pictures are partitioned into smaller coding NAL units in H.264/AVC follows the rather traditional concept of subdivision into macroblocks and slices. Each picture is partitioned into macroblocks that each covers a rectangular picture area of 16X16 luma samples and, in the case of video in 4:2:0 chroma sampling format, 8X8 samples of each of the two chroma components. The samples of a macroblock are either spatially or temporally predicted, and the resulting prediction residual signal is represented using transform coding. The macroblocks of a picture are organized in slices, each of which can be parsed independently of other slices in a picture. Depending on the degree of freedom for generating the prediction signal, H.264/AVC supports three basic slice coding type: I slice, where intra-picture predictive coding using spatial prediction from neighboring regions, P slice, where intra-picture predictive coding and inter-picture predictive coding with one prediction signal for each predicted region, B slice, where intra-picture predictive coding, inter-picture predictive coding, and inter-picture bi-predictive coding with two prediction signals that are combined with weighted average to form the region prediction [4].

2.4.2 Scalable H.264/SVC Video Coding

The Scalable Video Coding (SVC) is an extension of H.264/AVC standard. Unlike single-layer H.264/AVC, SVC provides network-friendly scalability at a bit-stream level with a little bit increase in decoding delay. Scalability in video streaming is supposed to provide functionalities such as bit rate, format, and power adaptation to the varying terminal capabilities or network conditions as shown in Figure 2.3. In H.264/SVC scalability refers to the removal of parts of the video bit-stream in order to adapt it to the various preferences of end users as well as to heterogeneous terminal capabilities or network conditions [4].

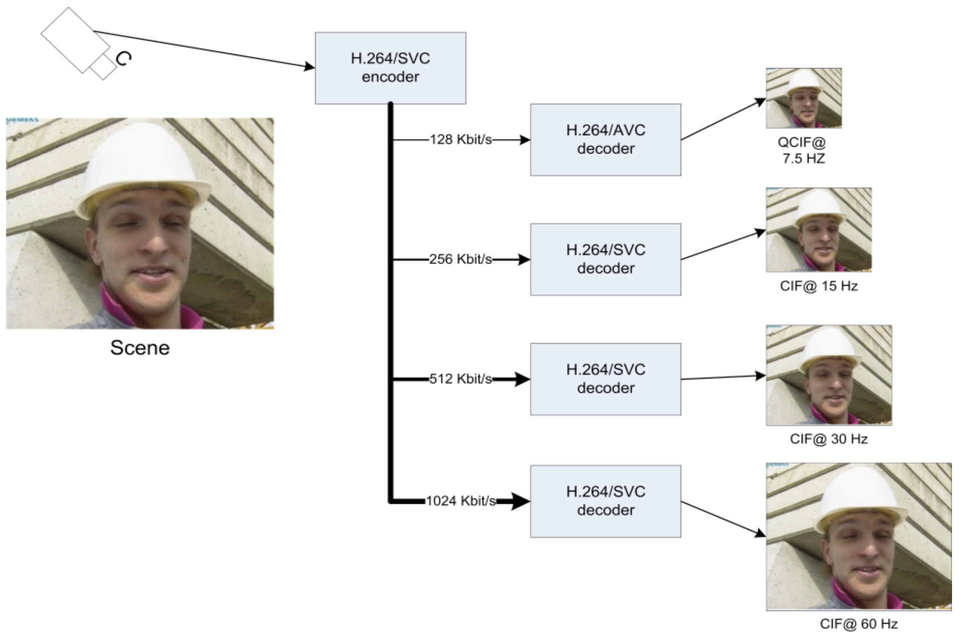


Figure 2.3: The principle of scalable video coding [3].

SVC has achieved a significant improvement in coding efficiency and scalability in comparison to the relative scalable profiles of prior video coding standards. It enables on-the-fly adaptation to certain application requirements and network time-varying transmission conditions. Due to this reason and the continuous evolution of receiving devices and the increasing usage of transmission systems that are characterized by a widely varying connection quality, the desire for scalable video coding has been rapidly increased. Video transmission over the Internet or wireless networks is exposed to variable transmission conditions, which can severely affect the pre-configured video streaming features and might lead to interrupt the video transmission session. Furthermore, video content is delivered to a variety of decoding devices with heterogeneous display and computational capabilities. In such heterogeneous environments, flexible adaptation of once-encoded content is desirable to avoid the overhead of re-transmitting a new encoded content especially for Real-Time applications. Meanwhile, it enables interoperability of encoder and decoder products from different manufacturers [4].

In H.264/SVC, video bit-stream is called scalable when parts of the stream can be removed in a way that the resulting sub-stream forms another valid bit-stream for some target decoder. The remains sub-stream represents the new source content for reconstructing a video with a less quality than that of the complete original bit-stream but with a high quality when comparing to the lower sub-stream. Video stream scalability is classified to three main types: temporal, spatial, and quality scalability (combination of those types can also be used). In spatial scalability, the subsets of the bit-stream have a reduced picture size (spatial resolution) while in temporal scalability they have a reduced frame rate (temporal resolution), see Figure 2.4. However, in quality scalability, the sub-stream provides the same spatio-termporal resolution as the complete bit-stream, but with a lower fidelity or as informally referred to as signal-to-noise ratio (SNR) [3].

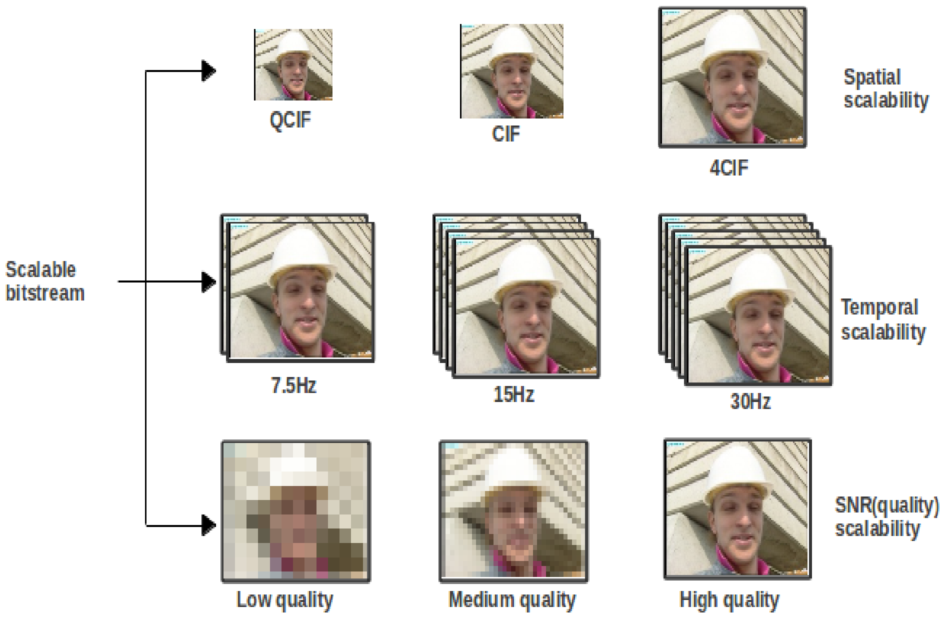


Figure 2.4: Types of scalability in video coding.

A bit-stream provides temporal scalability when the set of consecutive NAL units with specific properties (access units) can be partitioned into a temporal base layer and one or more temporal enhancement layers with the following property. Let the temporal layers be identified by a temporal layer identifier T starting from 0 for the base temporal layer and is increased by 1 for every added temporal layer. Then for each natural number n, the bit-stream that is obtained by replacing all access units of all temporal layers with a temporal layer identifier T greater than n forms another valid bit-stream for the given decoder.

For hybrid video codecs, temporal scalability can generally be enabled by restricting motion-compensated prediction to reference pictures with a temporal layer identifier less than or equal to the temporal layer identifier of the picture to be predicted. The prior video coding standards MPEG-1, H.262/MPEG-2 Video, H.263, and MPEG-4 visual all support temporal scalability to some degree. H.264/AVC provides a significantly increased flexibility for temporal scalability because of its reference picture memory control. It allows the coding of picture sequences with arbitrary temporal dependencies, which are only restricted by the maximum usable Decoded Picture Buffer (DPB) size. Hence, for supporting temporal scalability with a reasonable number of temporal layers, no changes to the design of H.264/AVC were required. The only related change in H.264/SVC refers to the signaling of temporal layers.

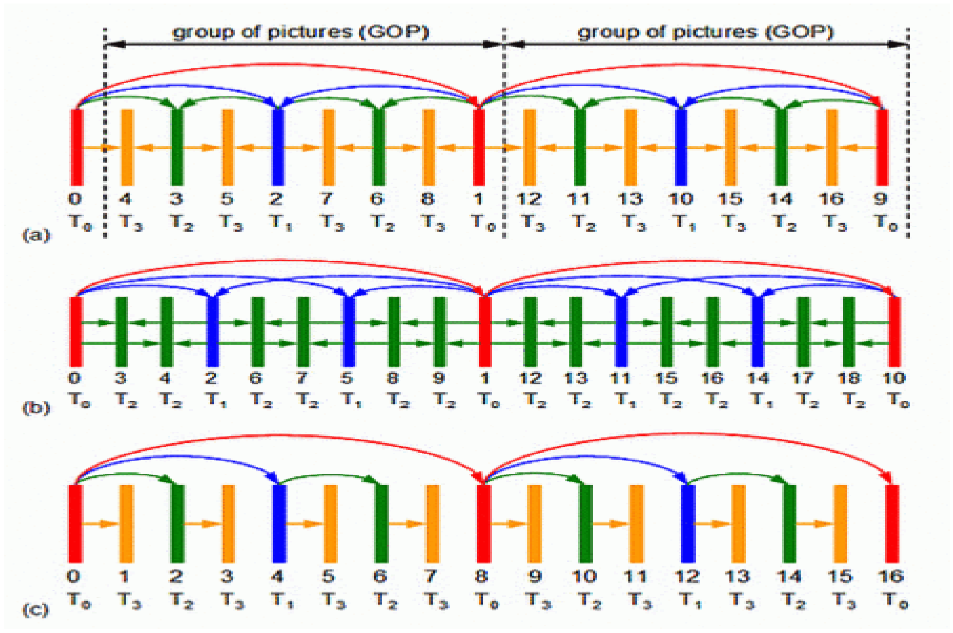


Figure 2.5: Hierarchical prediction structures for temporal scalability [4].

Temporal scalability with dyadic temporal enhancement layers can be very efficiently provided with the concept of hierarchical B or P pictures as illustrated in Figure 2.5a. The enhancement layer pictures are typically coded as B pictures; where the reference picture lists 0 and 1 are restricted to the temporally preceding and succeeding picture, respectively, with a temporal layer identifier less than the temporal layer identifier of the predicted picture. Since backward prediction is not necessarily coupled with the use of B slices in H.264/AVC, the temporal coding structure of Figure 2.5a can also be realized using P slices. Each set of temporal layers (T_0,...,T_n) can be decoded independently of all layers with a temporal layer identifier $T > T_n$. The set of pictures between two successive pictures of the temporal base layer together with the succeeding base layer picture is referred to as a group of pictures (GOP) [3].

Although the described prediction structure with hierarchical B or P pictures provides temporal scalability and also shows excellent coding efficiency, it represents a special case. In general, hierarchical prediction structures for enabling temporal scalability can always be combined with the multiple reference picture concept of H.264/AVC. This means that the reference picture lists can be constructed by using more than one reference picture, and they can also include pictures with the same temporal level as the picture to be predicted. Furthermore, hierarchical prediction structures are not restricted to the dyadic case. As an example, Figure 2.5b illustrates a non-dyadic hierarchical prediction structure, which provides two independently decodable sub-sequences with 1/9-th and 1/3-rd of the full frame rate. It is further possible to arbitrarily adjust the structural delay between encoding and decoding a picture by restricting motion-compensated prediction from pictures that follow the picture to be predicted in display order. As an example, Figure 2.5c shows a hierarchical prediction structure, which does not employ motion-compensated prediction from pictures in the future. Although this structure provides the same degree of temporal scalability as the prediction structure of Figure 5a, its structural delay is equal to zero compared to 7 pictures for the prediction structure in Figure 2.5a.

For supporting spatial scalable coding, SVC follows the conventional approach of multi-layer coding, which is also used in H.262/MPEG-2 Video, H.263, and MPEG-4 Visual. In each spatial layer, motion-compensated prediction and intra prediction are employed as for single-layer coding. In addition to these basic coding tools of H.264/AVC, SVC provides what so-called inter-layer prediction methods (see Figure 2.6), which allow an exploitation of the statistical dependencies between different layers for improving the coding efficiency (reducing the bit rate) of enhancement layers [3].

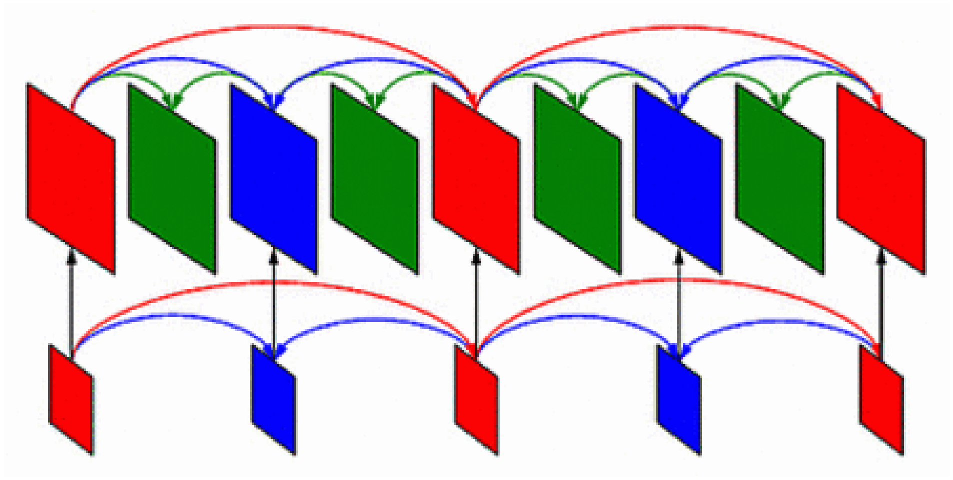


Figure 2.6: Multi-layer structure with additional inter-layer prediction [4].

In H.262/MPEG-2 Video, H.263, and MPEG-4 Visual, the only supported inter-layer prediction methods employ the reconstructed samples of the lower layer signal. The prediction signal is either formed by motion-compensated prediction inside the enhancement layer, by upsampling the reconstructed lower layer signal, or by averaging such an upsampled signal with a temporal prediction signal. Although the reconstructed lower layer samples represent the complete lower layer information, they are not necessarily the most suitable data that can be used for inter-layer prediction. Usually, the inter-layer predictor has to compete with the temporal predictor, and especially for sequences with slow motion and high spatial detail, the temporal prediction signal typically represents a better approximation of the original signal than the upsampled lower layer reconstruction. In order to improve the coding efficiency for spatial scalable coding, two additional inter-layer prediction concepts have been added in SVC: prediction of macroblock modes and associated motion parameters and prediction of the residual signal. All inter-layer prediction tools can be chosen on a macroblock or sub-macroblock basis allowing an encoder to select the coding mode that gives the highest coding efficiency [3]. Quality scalability can be considered as a special case of spatial scalability with identical picture sizes for base and enhancement layer. This case, which is also referred to as coarse-grain quality scalable coding (CGS), is supported by the general concept for spatial scalable coding as described above. The same inter-layer prediction mechanisms are employed, but without using the corresponding upsampling operations. When utilizing inter-layer prediction, a refinement of texture information is typically achieved by re-quantizing the residual texture signal in the enhancement layer with a smaller quantization step size relative to that used for the preceding CGS layer. As a specific feature of this configuration, the

deblocking of the reference layer intra signal for inter-layer intra prediction is omitted. Furthermore, inter-layer intra and residual prediction are directly performed in the transform coefficient domain in order to reduce the decoding complexity.
The CGS concept only allows a few selected bit rates to be supported in a scalable bit-stream. In general, the number of supported rate points is identical to the number of layers. Switching between different CGS layers can only be done at defined points in the bit-stream. Furthermore, the CGS concept becomes less efficient, when the relative rate difference between successive CGS layers gets smaller. Especially for increasing the flexibility of bit-stream adaptation and error robustness, but also for improving the coding efficiency for bit-streams that have to provide a variety of bit rates, a variation of the CGS approach, which is also referred to as medium-grain quality scalability (MGS), is included in the SVC design. The differences to the CGS concept are a modified high-level signalling, which allows a switching between different MGS layers in any access unit, and the so-called key picture concept, which allows the adjustment of a suitable trade-off between drift and enhancement layer coding efficiency for hierarchical prediction structures [3].
Drift describes the effect that the motion-compensated prediction loops at encoder and decoder are not synchronized, e.g., because quality refinement packets are discarded from a bit-stream. Figure 2.7 illustrates different concepts for trading off enhancement layer coding efficiency and drift for packet-based quality scalable coding.

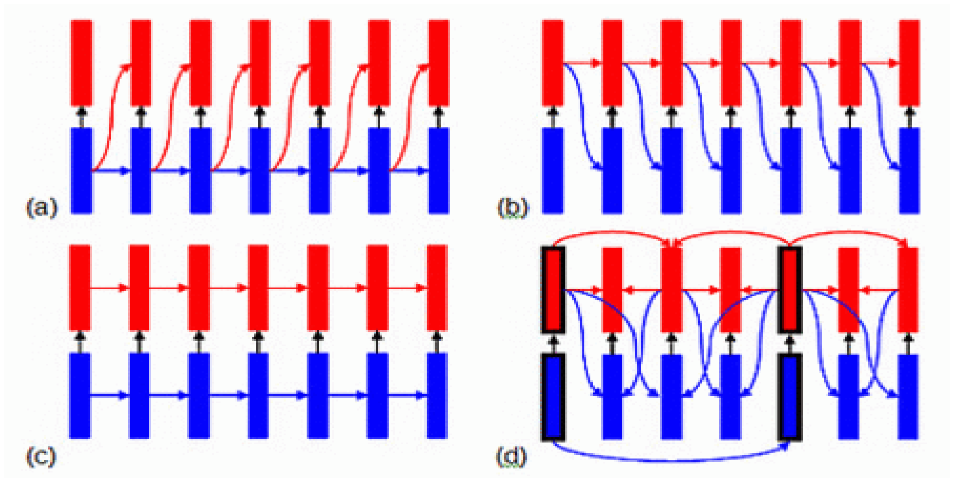


Figure 2.7: Various concepts for trading off enhancement layer coding efficiency and drift for packet-based quality scalable coding [4].

2.5 Real-Time Transport Protocol (RTP)

The Real-Time Transport Protocol (RTP) is a transport protocol that provides end-to-end network transport functions for transmitting Real-Time data, such as interactive audio and video. Services that are provided by RTP include payload type identification, sequence numbering, time stamping and delivery monitoring. RTP time stamping service allows placing the incoming audio and video packets in the correct time slot order. Normally RTP is placed on top of the User Datagram Protocol (UDP) and linked to the Real-Time Control Protocol (RTCP) to provide a mechanism for reporting feedbacks on the transmitted Real-Time data. In addition, RTP can be used in multicast audio conferencing as well as audio and video conferencing scenarios [2]. For more information about RTP for Real-Time application, audio and video conferencing, and H.264 video payload format, check RFC 3550 [27], RFC 3551 [28] and RFC 3984 [29].

2.5.1 Real-Time Control Protocol (RTCP)

The Real-Time Control Protocol (RTCP) is a data transport protocol used in conjunction with RTP for transporting Real-Time media streams. It includes functions such as supporting synchronization between different media types (e.g., audio and video) and providing the streaming applications with information about network quality, number of viewers, identity of viewers, etc. RTCP gives feedback to each participant in an RTP session which can be used to control streaming performance. These feedbacks include reception reports, number of lost packets and jitter statistics. The RTCP feedback messages can potentially be used by the higher application layer to modify the transmission mechanism [2].

2.5.2 Real-Time Streaming Protocol (RTSP)

The Real-Time Streaming Protocol (RTSP) is an application-level protocol for the control of Real-Time multimedia data. It acts as an auxiliary protocol for controlling video, audio, and multimedia sessions (e.g. play, pause and stop). Unlike the protocols that are responsible of the delivery of the video signals (e.g., RTP), RTSP allows these signals to be controlled by the user. Like a dispatcher for a delivery service, RTSP acts like a dispatcher for a delivery service since it does not actually deliver packages, instead it controls when and how packages are delivered by other protocols such as RTP. Basically, RTSP acts as a remote control for the media server [2].

2.6 Real-Time Publish Subscribe Communication Model

The Publish-Subscribe architecture is a data centric design permitting direct control of information exchange among different nodes in the architecture [30]. It is a sibling of the message queue paradigm, and is typically one part of a larger message-oriented middleware system. It generally relies on asynchronous message-passing, as opposed to a request-response architecture. It connects anonymous information producers with anonymous information consumers. The property of decoupling publisher and subscriber in time (data when you want it), in location (publisher and subscriber can be located anywhere) and in platform (connect any set of systems) makes the publish-subscribe communication model more appropriate for large scale and loosely coupled distributed Real-Time systems than traditional models such as client-server models [31].

Client-server communication drawbacks, e.g., server bottleneck, single points of failure and high bandwidth load in many-to-many communication are resolved by publish-subscribe communication model [32]. Unlike client-server communication model, data in publish-subscribe communication model is pushed by the publishers to *topics* or named logical channels where subscribers will receive all messages published to the topics to which they subscribe immediately after the data is produced without the need of request, and thus subscribers can access the data in Real-Time. In addition, publish-subscribe architecture frees the data sender (publisher) from waiting for an acknowledgement by the receiver (subscriber). As a result, the publisher can quickly move on to the next receiver within deterministic time without any synchronous operations which is desirable for a large scale distributed Real-Time systems [31]. Recently, the publish-subscribe communication model has become popular in different middlewares such as Java Message Service (JMS), Microsoft Component Object (COM+) and Data Distribution Service (DDS).

DDS is a high performance middleware standardized by the Object Management Group (OMG) for QoS-enabled publish-subscribe communication aimed at distributed Real-Time and embedded systems [5]. At the core of DDS is the Data-Centric Publish-Subscribe (DCPS) layer that is targeted towards the efficient delivery of the proper information to the proper recipients for applications running on heterogeneous platforms [5]. DCPS builds on a Global Data Space (GDS) by which applications or participants running on heterogeneous platforms can share information by publishing data under one or more topics of interest to other participants. On the other hand, applications or participants can use the global data space to declare their intent to become subscribers and access data of interested topics. Each topic represents a logical channel for connecting publishers to all interested subscribers. Figure 2.8 represents the dissemination of data from one or more publishers to interested subscribers in the DDS middleware.

Moreover, DDS is a publish-subscribe standard with a diverse set of Quality of service (QoS) that ensures high performance and low delay of data transmission [5].

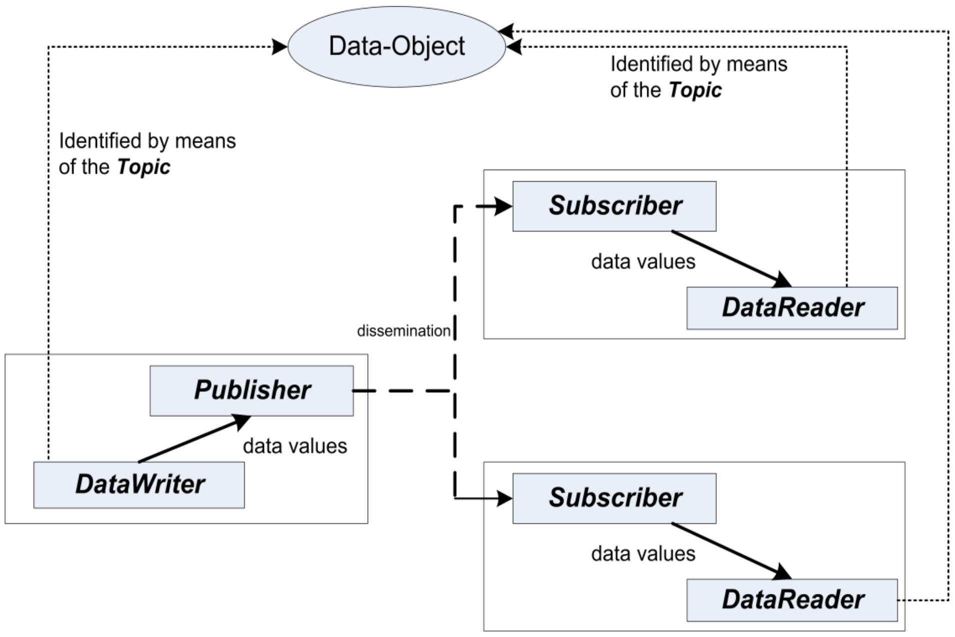


Figure 2.8: Overview of publish-subscribe using DDS. [5].

CHAPTER 3

REAL-TIME VIDEO STREAMING TECHNIQUES

Due to the rapid growth of wireless communication, video streaming over wireless networks has gained a great amount of research. However, Real-Time video streaming over wireless networks still suffer from the problem of time-varying changes in wireless conditions due to changing distance between the enabled devices, signal fading, noise interference, and network congestion, leading to time-varying packet loss rate and fluctuating effective bandwidth. Therefore, the provisioning of end-to-end QoS in wireless communications is a very challenging and demanding task.

The literature of Real-Time video streaming has concentrated on the avoidance of network congestion since it severely affects the performance of video streaming. This is performed by adapting the source video bit rate to the network channel bit rate. On the other hand, if frame's packet loss occurs due to unavoidable errors, many papers have proposed error concealment and resilience mechanisms for preventing decoders from discarding the entire frame and breaking of the continuity of the video streams. In this chapter, I classify the papers that address the Real-Time video streaming area into five major approaches; video rate adaptation approach, channel assignment approach, cross-layer design approach, error resilient approach, and middleware approach.

3.1 Video Rate Adaptation

Congestion is a common phenomenon in network communications that occurs when the offered load exceeds the designed limit, causing degradation in network performance such as throughput. Useful throughput can be decreased for a number of reasons. For example, it can be caused by collision in multiple access networks, or by increased number of retransmissions in reliable systems. Besides a decrease in useful throughput, the network traffic might be exposed to other problems such as packet losses, higher delay or delay jitter. To avoid such undesirable consequences of congestion, control procedures are often employed to limit the amount of network load. Such control procedures are called rate adaptation or congestion control.

In video streaming over wireless networks, a proposal of adapting the bit rate of the video encoder based on verified network status (e.g., available bandwidth) has been widely adopted. Five main approaches have been proposed under the concept of rate adaptation: *rate control*, *transcoding*, *bit-stream switching*, *packet pruning*, and *scalable coding.* Transcoding and bit-stream switching [33] approaches might not be suitable for Real-Time live video streaming. Bitstream switching requires pre-encoded video contents at different rates and quality levels which introduce more delay and not practical for live video show. Transcoding referes to the conversion of one encoding data to another when the receiver does not support the format or has limited storage capacity. However, transcoding is commonly a lossy process, introducing video quality loss. It requires some computational cost and in some cases it may be necessary to decode the content and re-encode according to end-user restrictions. Some literatures have proposed error resilient transcoding approaches [34, 35, 36] to solve the loss problem in order to be suitable for real-time video streaming, but still restricted to the prestored videos not a live one.

Unlike transcoding, the idea of the rate control approach is to change the bit rate of the video encoder without changing video formats. This change is according to the negative feedback of the available network resources by using some QoS indicators e.g., packet loss, packet deadline, etc., while maintaining a reasonable video quality and avoiding any modifications to the network infrastructure. On of the early and widely accepted rate control approach is the TCP-friendly Rate Control (TFRC) [37]. TFRC is an equation-based congestion control for unicast multimedia traffic based on the TCP Reno's throughput equation. In TFRC, the sender adjusts its sending rate as a function of the measured rate loss, where a loss consists of one or more packets dropped within a single round-trip time. However, TFRC is mainly proposed for wired networks especially for the Internet and when applied to the wireless networks, it suffers from performance degradation [38]. This is because TFRC method assumes perfect link quality and considers the network congestion as the only packet loss reason while most of packet loss in wireless networks is due to error at the physical layer. As a result, literatures such as [38, 39, 40, 41, 42, 43, 44] proposed a new optimized TFRC-based mechanisms to support error-prone networks as wireless networks. In addition, authors in [45, 46, 8, 47, 48, 49, 50] proposed different rate control schemes to adapt video stream to the available network resources. As instance [8] proposed an algorithm for verifying network status by using MAC-layer parameters implying PHY-layer information and then correspondingly adjusting the target bit rate. Despite the success of the rate control approach in Real-Time video streaming, it may fail under the multicast video streaming scenarios over multi-rate wireless LANs. In multicast video streaming, rate control mechanism requires to send a specific video transmission rate for every user of different wireless network channel conditions. This may involve the estimation of every users channel resources and the generation of multiple video transmission bit rates which introduce an overhead.

Recently, the idea of multi-layer scalable video stream start to be dominant in the field of video streaming. Unlike the single-layer rate adaptation approaches (e.g., rate control, transcoding, bit-stream switching), multi-layer video stream consists of a base layer and other enhancement layers which are independent of each other. That is, dropping an enhancement bit-stream layer will not severally affect the whole quality of the decoded video. Scalable (multi-layer) video streaming overwhelms non-scalable (single-layer) video streaming in heterogeneous and error-prone networks. It allows data rate adaptation without re-encoding, just by dropping bit-stream packets. This property eliminates the overhead of transcoding and bit-stream switching to adapt video rate to the available network resources. The Scalable Video Coding (SVC) is an implementation of multi-layer scalable video stream. H.264/SVC provides network-friendly scalability at the bitstream level with a moderate increase in decoding delay.

Scalability in video streaming is supposed to provide functionalities such as bit rate, format, and power adaptation to the varying terminal capabilities or network conditions. In H.264/SVC, scalability refers to the removal of parts of the video bit stream in order to adapt it to the various preferences of end users as well as to varying terminal capabilities or network conditions [4]. However, scalable video coding as H.264/SVC still requires end-to-end QoS for maintaining the priority of which frames packets of which layer are supposed to be dropped first in order to control video traffic congestion and deliver better video quality [51]. Forward Error Correction (FEC) mechanism also proposed for the scalable video streaming by allocating different amount of FEC codes to different layers according to their priority to achieve graceful degradation [52, 53, 54]. In [53] an unequal layer error protection has been proposed in a DVB-H transmission of layered video on response to packet losses. A research work as in [54] proposed an unequal error protection for SVC base and enhancement layers while considering the transmission of on-demand scalable variable-bit-rate (VBR) video and the existence of receiver playback video buffer. However, this research work is different in which it consider the real-time video transmission.

On the other hand, a good amount of researches have been conducted in a cross-layer schemes as a proposed solution for delivery of scalable video over multirate wireless networks as 802.16 or IEEE 802.11 [26, 55, 56, 57, 58]. As instance in [26], authors have proposed a cross-layer design to optimize the link adaptation scheme that configures the PHY and MAC layers, and treat SVC enhancement layers differently in a way that the highest possible video quality is achieved by avoiding dropping layers and without adding to the traffic load of the WLAN. Link adaptation optimization is used to determine the number of video layers permitted, and the PHY transmission mode assigned to each video layer. However, such proposed approaches are complex and only exclusive for those wireless networks which employ a variable rate PHY and a link adaptation mechanism as in 802.11n multi-input multi-output (MIMO), 802.11a and 802.16. Some other literatures proposed a real-time and scalable delivery of SVC-based video over MIMO networks as in [59, 60].

3.2 Video Error Control Coding

Data packets may be lost or corrupted due to either traffic congestion or bit errors due to impairment of the physical channel as in wireless networks. Retransmission techniques as Automatic Repeat Request (ARQ) [61, 62] have been used as a solution to such problem but with unacceptable delay for real-time applications. In broadcast application as in real-time video streaming, retransmission techniques are not used completely due to the playback constraints of every video packet and the network flooding considerations. Forward error correction (FEC) techniques could also be employed to reduce the effects of errors on the decoded video quality [63]. However, in the Scalable Video Streaming (SVC) when burst packet loss occurs and a certain layer is lost, the received packets in the higher layers will become useless [63]. As a result, literatures as in [52, 53, 54] comes to propose the idea of allocating different amount of FEC codes to different layers according to their priority to achieve graceful degradation. In [53] an unequal layer error protection has been proposed in a DVB-H transmission of layered video on response to packet losses. A research work as in [54] proposed an unequal error protection for SVC base and enhancement layers while considering the transmission of on-demand scalable variable-bit-rate (VBR) video and the existence of receiver playback video buffer. However, this research work is different in which it consider the real-time video transmission. In fact, packet loss or corruption is inevitable and therefore using encoding/decoding schemes that can make the compressed bit-stream resilient to transmission errors is required in real-time video applications to avoid a significant degradation in video quality [11].

Video bit-stream error resilience techniques can be categorized into three main groups: error resilient encoding technique which make the compressed bit-stream more resilient to potential errors; decoder error concealment technique which is used at the decoder side to estimate the missing or corrupted video samples based on the surrounding received samples, by making use of the inherent correlation among spatially and temporally adjacent samples; and adaptive error resilient technique in which the encoder can adapt its operation based on the loss condition detected at the decoder. Unlike error resilient encoding, decoder error concealment technique is not employing any additional bit rate, but adds computational complexity to the decoder [11]. The problem of error control approach is that it implies adding redundancy or more bits to the encoded bit-stream. Therefore, the encoding efficiency will decrease due to the increase size of every video sample. Moreover, error control approach can lead to a significant delay when a large number of video packets are corrupted, in which discarding such packets might be a better decision for real-time video streaming.

3.3 Channel Assignment

Multi-Channel Multi-Interface (MCMI) wireless ad hoc networks have received amount of interest, especially under the context of Real-Time video streaming since multi-channel can provide higher performance than single channel. Unlike single channel, multiple interfaces and multiple channel technology offers a concurrently multiple transmissions/receptions by using multiple interfaces equipped on each node. Moreover, neighboring links assigned to different channels can carry traffic free of interference by exploiting multiple available channels, thereby multiplying the network throughput and reducing the link-layer delay dramatically.

Channel Assignment (CA) is a key technique to relieve signal interference and to increase network capacity in MCMI networks. Streaming video over MCMI networks requires an efficient channel assignment strategy that guarantees an efficient use of available channel resources for Real-Time video streaming. In this aspect, different channel assignment mechanisms have been proposed in [64, 65, 66, 67, 68], which can be further divided into three main categories [69], i.e., static, dynamic, and hybrid channel assignment. A statistic link load based hybrid channel assignment strategy (SLL-HCA) [70], is one of the channel assignment solutions for Real-Time video streaming that based on a representative hybrid channel assignment strategy (HMCP). This strategy is presented to obtain a better channel assignment metric than in HMCP. SLL-HCA is based on the HMCP protocol and adopts the statistical link load metric to ensure load balancing in a two-hop neighborhood, and to prevent both the hidden node problem and the exposed node problem. SLL-HCA and its advanced version VE-SLL-HCA are proposed mainly for reserving lower interference of routing path for video-streaming traffic than other non-video traffic and for improving the QoS support of video-streaming over multi-channel ad hoc networks.

No doubt that channel assignment approach has a significant improvement in real-time video streaming but it is specified to a certain networks that support MCMI.

3.4 Cross-layer Design

The idea of cross-layer approach is to exchange information between different layers to support Real-Time video streaming over wireless networks. This approach allows some layers to use the information from other layers to make a better strategic decision.

A number of cross-layer design schemes aims at Real-Time video streaming have been recently proposed [71, 72, 73, 74, 75]. A new adaptive cross-layer mechanism (ACMRV) [71] is proposed for Real-Time video streaming over MCMI wireless networks. It includes both an efficient channel assignment (CA) and adaptive Foreword Error Correction (FEC) mechanism. Both packet queue length from PHY layer and the available bandwidth calculated from MAC layer are used for selecting a better channel for forwarding video streaming packets, and added redundant packets of adaptive FEC for unavoidable packet errors.

In [73, 75], different adaptive cross layer techniques have been proposed to optimally enhance the QoS of wireless video transmission in an IEEE 802.11e WLAN. These techniques make use of two analytical models: a video distortion model and channel throughput estimation model for predicting the video quality in term of average PSNR of all decoded video frames and, the channel throughput and packet loss rate of each MAC layers queue. The estimated parameters are then fed into the analytical models in order to optimize the selection of the two IEEE 802.11e EDCA MACs channel access parameters; CWmin and AIFSN. These mechanisms basically aim at the selection of appropriate CWmin and AIFSN for reducing the impact of MAC contention and improve the QoS transmission of video traffic.

The authors in [74] have proposed a link adaptation strategy for IEEE 802.11 WLAN that estimates the received perceptual video quality at the current and adjacent PHY rates. The PHY rate that produces the best perceptual quality is chosen for each Group of Pictures (GOP). A subset of codec-related parameters and network-related parameters is chosen to be cross-layer information about the system. This information is then input to a module that estimates a video quality metric, e.g., Peak Signal-to-Noise Ratio (PSNR), Mean Absolute Difference (MSAD), Structural Similarity Index (SSIM), VQM, and Mean Squared Error (MSE).

Although Cross layer design approach optimally uses the wireless channel under different conditions for transmitting the video streams, it has some drawbacks as follows: i) It is not a general solution for Real-Time video streaming since it depends on the wireless network architecture, for example, cross layer design approaches that proposed for 802.11n MCMI based networks are different from those proposed for 802.11e EDCA based networks; ii) The exchanged information between different layers may introduce additional overhead and thus affects the performance of video streaming; iii) It is very complex approach since the development of a video streaming system based on cross-layer mechanism requires deep network understanding.

3.5 Middleware

Distributed system based on middleware technologies have been recently proposed as a practical solution for the integration of distributed control systems (DCS) [76, 77]. Middleware technology provides interoperability among heterogeneous DCS with an acceptable Real-Time transmission QoS of huge amount of data. Industrial applications have become more sophisticated. Most of these applications requires a human assisted decision based on video processes to control their traditional physical processes. In this section I summarize the most relevant research work on the topic of middleware based Real-Time video streaming.

Authors in [76] proposed that a CORBA middleware based implementation can be used to offer Real-Time video streaming. They have applied a CORBA based Real-Time video surveillance system in a non-wireless network to a country-wide DCS integration problem in order to see the effects of their actions in remote hydroelectric power plants. Although CORBA is a very complete technology that introduces a big number of interfaces for almost any type of required middleware functionality, it is a complex architecture that introduces implementation overheads, in particular if compared with other lighter weight technologies such as ICE (Internet Communications Engine) [78], DDS (Data Distribution Service for Real-Time systems) [5], or some specific Real-Time Java based solutions [79]. Therefore, existing approaches can be improved to offer appropriate support to the Real-Time nature of video transmission. In addition, using new standard middleware introduces flexibility for video transmission in two ways. First, compared to direct implementation over the network level, the utilization of a middleware is already more flexible. Second, utilizing middleware solution offers QoS management which allows to appropriately initiate Real-Time support for video transmission. Another work in [80] presents an architecture of a Real-Time and QoS-based video surveillance. This work shows how the decoupled interaction paradigm of the DDS[5] middleware can be used for the development of higher complexity surveillance systems. A prototype video transmission surveillance system is presented by sending a single-layer video stream. This video stream adapts to luminosity conditions by using the QoS resource management component that may require the compression factor to be altered in order to fit the stream to the new requirements. This way of adaptation introduces more encoding latency which is not appropriate for Real-Time video streaming applications.

Detti et al., [81] demonstrate and evaluate a technique for streaming H.264/SVC video over a DDS[5] middleware in an 802.11 wireless scenario. Authors in this work developed a receiver-driven rate control mechanism to maximize the quality of the received video based on the built-in DDS[5] functionality. This mechanism estimates the available network transfer capacity to compute the highest video sub-stream that the subscriber can receive. The available channel capacity is estimated by configuring the video publisher to send a fixed number of aggregated NAL unit packets periodically. Then calculate the available capacity as a ratio between the number of bits of the aggregation set and its duration. However, this aggregated set of the NAL units adds more delay and increase the bursty of the transmitted traffic which should not be considered in Real-Time video streaming architecture.

CHAPTER 4

DESIGN & IMPLEMENTATION OF SCALABLE RTPS-BASED VIDEO STREAMING

This chapter covers the architecture and implementation of the proposed scalable video streaming system over Real-Time Publish Subscribe Protocol (RTPS). The RTI Data Distribution Service (RTI-DDS) [5] is used for implementing the proposed architecture.

4.1 System Architecture

The architecture of the proposed scalable RTPS-based video streaming as shown in Figure 4.1 consists of five main components: the Video Encoder, Video Decoder, Video Publisher, Video Subscriber, and Video Streaming QoS.

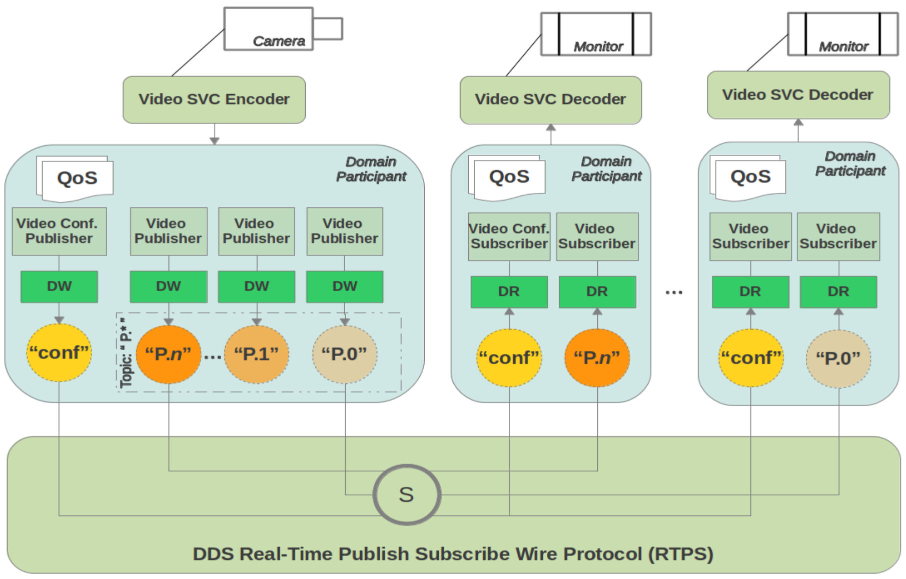


Figure 4.1: Scalable RTPS-Based Video Streaming Architecture.

4.1.1 Video Encoder

In the video encoder module, both H.264 Advance Video Coding (H.264/AVC) and H.264 Scalable Video Coding (H.264/SVC) can be used based on the system requirements. H.264/AVC can be used in a condition of small system participants having less scalability and complexity requirements. On the other hand, H.264/SVC can be used in a condition of large system participants with higher scalability requirements. For implementing H.264/AVC, X264 [82] encoder is used while JSVM [83] library is used for implementing H.264/SVC.

Video encoder receives the captured frame data of YUV420 type from the camera. And then it encodes every frame depending on the encoder configuration and type. The results are the Abstract Network Layer data (NAL) units which provide network friendliness by enabling simple and effective customization of the use of video coding data for a broad variety of systems. Every frame might be in one NAL unit or slices in more than one NAL unit. Every scalability layer (sub-stream) in a video stream encoded by an SVC encoder contains a sequence of NAL units. Each NAL unit contains the encoding bits (payload) and is encapsulated by a header to identify the sub-stream (scalability layer) that it belongs to. The NAL unit header identifying parameters are used in the process of assigning NAL unit to a publishing video streaming partition track.

4.1.2 Video Publisher

Video publisher is the part where the NAL units (video payload) and other Meta data (e.g., frame number, SVC layer number) are published. Two main topics are configured in this proposed architecture: video configuration topic "$conf$" and video NAL units topic "$P.*$". The video configuration topic enables video subscribers to join the video stream session and configure the SVC decoder based on the published encoding parameter set. The video NAL units topic are partitioned at run-time into a number of partitions depending on the number of encoded scalable sub-streams. The configured video partition QoS connects publishers/subscribers to a video partitions list which might also contain wildcards, e.g. "$P.*$". That is, video partition of a topic "$P.n$" enables the subscriber to join the complete video bitstream with the best quality while "$P.n - 1$" joins the subscriber to a sub-stream with lower quality.

Every publisher has a DataWriter (DW) for the dissemination of the video data to the middleware global data space (GDS) under a certain video topic. In order to design the proposed architecture, the Interface Description Language (IDL) data type that represents the packet data structure of video configurations and NAL units video publisher DataWriters is defined to contain the following fields:

```
const long MAX_NAME_LEN = 32;
const long MAX_PAYLOAD_SIZE = 1024;

struct VideoStream {
string <MAX_NAME_LEN> sender;    //@key
long frameSize;        //frame size
long frameno;       //frame number
char frametype;        //frame type (e.g., I-frame, P-frame, B-frame)
octet tid;          //NAL temporal layer id
octet did;          //NAL spatial layer id
octet qid;          //NAL quality layer id
char framepayload[MAX_PAYLOAD_SIZE];  //NAL payload};
```

4.1.3 Video Subscriber

Video subscriber is the part where the data sequence is received in order to be encoded to the original video frames. A participant subscriber requires to register its interest to receive a video data of a certain Topic from a specific video partition in order to begin receiving the published video samples of the corresponding video partition. A valid QoS contract between the video publisher's DataWriter (DW) and subscriber's DataReader (DR) is also required for initiating the transmission process. By default, every subscriber is initially configured to receive from the publisher video partition with the highest video quality. Meanwhile, the video subscriber can dynamically switch to read from other video partitions. The decision of switching among partitions is handled by the subscriber in response to the feedback coming from the configured user QoS adapting the transmission data to the network channel limitations (e.g., available bandwidth). Every video frame that has been received by the subscriber's DataReader is directly sent to the video decoder module to be processed.

4.1.4 Video Decoder

Video decoder receives the ordered sequence of video frames partition from the subscriber's DataReader which gathers data from a buffer of Group Of Pictures (GOP) size. It then decodes every frame based on the decoder video quality configuration. The results are video images or frames that are immediately and properly rendered in every playback amount of time (normally 30ms) by an image processing engine. The Open Source Computer Vision (OpenCV) [84] image processing library is used in the implementation part for capturing frames from the camera and to playback the encoded pictures. It is necessary that the decoder must be initialized prior to the subscription process. This is because of the unnecessary delay caused by initializing the decoding configuration process while the publisher and the subscriber are already communicating.

4.1.5 Video Streaming Quality of Service

A set of quality of services (QoS) are configured to be suitable for Real-Time video streaming over lossy networks as shown in Table 4.1. Two main points were taking into account while configuring those QoS parameters. First, the delivery of each video frame must be within the permitted playback duration so the client is able to watch the live video. The next point is the scalability of the system which means the ability of the system to deliver the proper video quality to the proper receiver taking into account the time varying network channel bandwidth and the receiver limited resources in a heterogeneous system of various resources and capabilities. Some quality of services requires the publisher and subscriber QoS to be matched for communication. The DDS QoS policies has request-offer semantics: the publisher must offer a level of service (QoS) that is greater than or equal to the level of service requested by the subscriber. However, in some QoS levels of service, the subscriber request level of service must be matched with the publisher offered level of service (e.g., Reliablility QoS). In Table 4.1 if R×O is assigned to Y, it means that the actual subscription will not be established unless publisher and subscriber QoS are matched.

4.1.5.1 Reliability

Controls the reliability between the publisher's DataWriter and the subscriber's DataReader. When the reliability QoS is set to RELIABLE, the system will at-

Table 4.1: Scalable RTPS-based Video Streaming QoS.

	Publisher	Subscriber	R×O
Reliability	BEST_EFFORT	BEST_EFFORT	Y
History	KEEP_LAST (depth = 1)	KEEP_LAST (depth = GOP)	N
Durability	VOLATILE	VOLATILE	Y
Partition	"$P.*$"	"$P.*$"	N
Deadline	Playback deadline	Playback deadline	Y
Time Based Filtering	Not applicable	<Playback deadline	N/A
Lifespan	Playback deadline	Not applicable	N/A
Presentation	Publish order	Publish order	Y

tempt to repair samples that were not successfully received. Therefore, reliability is also controlled in conjunction with other QoS policies, such as History and ResourceLimits, to determine which data remains relevant and therefore eligible for repair.

In Real-Time video streaming, frames are supposed to be delivered with minimum latency, subject to their playback deadlines. Therefore, reliability between publisher's DataWriters and subscriber's DataReader is not an issue in Real-Time video streaming.

Reliability QoS of the video payload NAL contents is set to BEST_EFFORT value, which means that the system will not use any resources to guarantee that the data sent by a DataWriter is received by a DataReader. Best effort deleivery is the fastest, most efficient, and least resource-intensive (CPU and network bandwidth) method of getting the newest/latest value for a topic from DataWriters to DataReaders but with the cost of no guarantee to receive data. Best effort delivery is suitable for Real-Time video streaming over lossy networks such as wireless, since it is more efficient and losing some video frames will affect the quality of the encoded video but not corrupting the whole video data. On the other hand, video configuration data publisher's DataWriter that is responsible for initiating the decoder at the subscriber side is set to RELIABLE to gurantee that the configuration data will be delivered to the subscriber's DataReader successfully.

4.1.5.2 History

Controls how the system manages frames payload sent by a publisher's DataWriter or received by a subscriber's DataReader. It helps tune the reliability between publishers and subscribers.

In Real-Time video streaming reliability is not a matter, so there is no need for keeping history of the recent published data for retransmission. The history QoS is set to KEEP_LAST value with one Group of Pictures (GOP) of depth for the subscriber's DataReaders. The decoder at the subscriber side requires that the received frames should be in the same publishing order for decoding reference frames. Moreover, this buffer helps in reducing the video delay jitter problem but within the playback deadline.

4.1.5.3 Durability

Durability controls whether or not new subscribers get data which was published by publisher's DataWriters earlier, to increase system tolerance to failure conditions. It is obvious that in live video streaming, the new joining participants should follow the live show while the previous show events are useless. However, decoding frames like P or B-frames are based on prediction compensation algorithms. For example, P-frame can only be decoded with reference information from previous I or P-frames. In addition, B-frame can only be decoded with reference information from the previous and successive I or P-frames [25]. Due to this fact, durability QoS can be set to TRANSIENT value which means that the frame which has been already sent may be relevant to late-joining subscribers and subjected to any history depth, lifespan, and content or time-based filters defined. Data will be cached with the DataWriter that originally produced it. Durability QoS of video payload NAL contents is set to VOLATILE since the reliablity QoS is defined as BEST-EFFORT unreliable connection, which means that late-joining subscribers will not receive any previous video frames. On the other hand, video configuration publisher's DataWriter, which is responsible for initiating the decoder at the subscriber side is set to TRANSIENT_LOCAL to gurantee that late-joining subscriber will be able to get the decoder initiating configuration set.

4.1.5.4 Partition

The partition QoS provides another way to control which DataWriters will match and communicate with which DataReaders. Normally, DataWriters are matched to DataReaders of the same Topic. However, by using the partition QoS policy, additional criteria can be used to decide if a DataWriters data is allowed to be sent to a DataReader of the same topic. One or more strings can be added to the DataWriters publisher or DataReaders subscriber parent topic. In such a case the DataWriter is only matched to a DataReader for the same topic only if their publisher and subscriber have a common partition.

Partition QoS has some key fesures that play a main role in this proposed scalable RTPS-based Real-Time video streaming architecture. First, subscription to a certain video partition can be dynamically changed at runtime. This is used in the proposed scalable architecture to quickly control the subscribers DataReader to begin receiving lower video quality from another DataWriter when a network degradation is introduced. Second, partition QoS policy can dynamically configure the connection topology without stopping/starting or destroying/re-creating publishers, subscribers or even a participant. As a result, there is no spawning and killing of threads or allocation and deallocation of memory when publishers and subscribers add or remove themselves from partitions. This property is appropriate to Real-Time applications since keeping a low latency is a critical issue. In this proposed scalable architecture, every video sub-stream is assigned to a certain video partition. By default, the video subscriber subscribes to the highest video partition, so it reads video stream from the matched DataWriter of the highest video quality. It adaptively subscribes to a lower video partition (lower video quality) when network degradation is notised (e.g., low bandwidth, congestion, etc). Thus, subscriber's DataReader immediately receives video frames from the matched and proper publisher's DataWriter.

4.1.5.5 Deadline

Deadline period is set to a specific value that estimates when the frame packets should be received at the subscriber side. It is also used as an indicator of network performance degradation. When frames packets fail to reach within the estimated deadline period, it means that the system is performing improperly and the subscriber should switch to begin receiving videos of lower quality from another publisher DataWriter's partition. Partition QoS is configured to the maximum frame playback deadline; normally 150ms.

4.1.5.6 Time Based Filtering

Time based filtering QoS policy controls the rate of data samples that should be delivered to a DataReader within the permitted deadline. Data samples for a DataReader can be filtered out using the TIME_BASED_FILTER QoS by setting the minimum separation time. Once a data sample for an instance has been received, the middleware will accept but drop any new data samples for the same instance that arrives within the time specified by "minimum_separation". Minimum separation time should be less than the deadline time. Simply, time based filter QoS allows receiving the data samples within a period of time (deadline) but after the time specified by minimum_separation, as shown in Figure 4.2.

In this proposed video streaming architecture, time-based filter QoS is used to optimize resource usage (CPU and possibly network bandwidth) by only delivering the required amount of video samples (NALs) to different DataReaders, and filtering out samples that arrive faster than a specified rate (period of time between video samples arrival).

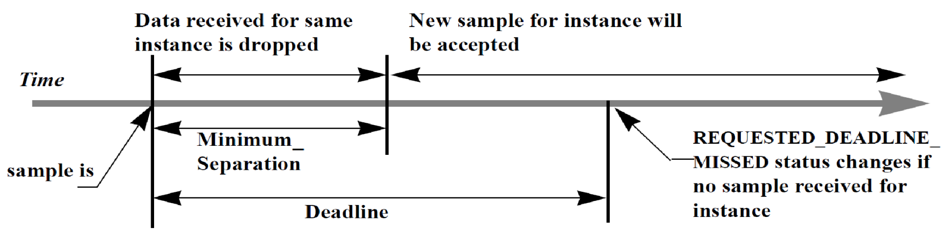

Data received for same instance is dropped
New sample for instance will be accepted
Time
sample is
Minimum_ Separation
Deadline
REQUESTED_DEADLINE_ MISSED status changes if no sample received for instance

Figure 4.2: Time Based Filter QoS [6].

4.1.5.7 Lifespan

Lifespan QoS Specifies how long the system should consider data sent by a publisher to be valid. It is used to timestamp all data sent and received. In this proposed video streaming system, lifespan QoS is set to a period equal to the maximum frame playback time. The DataReaders receiving queue is continuously checked out to see how long the frames packets have been stored before playout by comparing their timestamp to the current time. Video sample that has exceeded its lifespan duration will be removed from the DataReaders receive queue. Therefore, ensure that the system doesnt receive or act on video data that are too old and have expired.

4.1.5.8 Presentation

Presentation QoS policy controls the order of data received by DataReaders. Usually DataReaders will receive data in the order that they were sent by a DataWriter. In some conditions data might arrive out of order, for instance when using a reliable connection. In such conditions, data will be buffered until all previous samples arrive and presentation QoS will play a role in how to present those samples to the DataReader. Moreover, a set of data for the same topic sometimes is needed to be presented to the receiving DataReader only after all of the elements of the set have been received, but not before, or in a different order than what was received. Thus, presentation QoS policy allows the user to specify different scopes of presentation, within a topic, across instances of a topic, and across different topics of a publisher.

In video streaming applications, frames or video samples should be retrieved in the same order as were originally sent. The Presentation QoS is used in this proposed RTPS-based Real-Time video streaming architecture to guarantee that the video NAL units are retrieved by the subscriber's DataReader as were originally sent by the publisher's DataWriter.

4.2 System Scalability and Behaviour

Scalability in my proposed approach means the ability of the system to adaptively serve different video subscribers (clients) with the appropriate video quality that is proportional to the time-varying wireless channel conditions (e.g., time-varying bandwidth), or limited computational resources. The approach uses the Data Distribution Service (DDS) [5] middleware which contains a built-in Real-Time Publish Subscribe Protocol (RTPS) in order to stream Real-Time video. My scalability technique adopted the passive or non-intrusive technique for estimating the available bandwidth and user's capabilities. Based on the existence traffic in the network, DDS data QoS is used to estimate the potential congestion and packet loss occurrence and thus control the transmitted video NAL unit packets. Two scalability architectures are proposed in this research work: one for the single-layer video streaming H.264 AVC and the other for multi-layer video streaming H.264 SVC. Re-encoding, retransmission, pre-encoded bit-stream switching and transcoding are not considered because the aims and scope of this research is the scalable streaming of live video through an error-prone lossy networks such as IEEE 802.11.

The proposed scalability mechanism is based on unequal layer protection to protect the most important video NAL's packets with the cost of dropping the less important video NAL's packets. The parameter set (e.g., frame type, sub-stream layer in SVC, etc) in every NAL unit header are used to assign every NAL unit packet to a specific video partition track. Single-layer encoded video stream as in H.264/AVC can be treated as a temporal scalable video stream, if the encoded bit-stream has the correct properties. For example, have a correct temporal scalable video picture sequence. That is, reference pictures as I and P-frames both are considered as the base layer while the subsequence of the hierarchical B-frames are considered as the next enhancement layers. Figure 4.3 shows my proposed temporal scalability for single-layer video stream using the DDS middleware over wireless networks.

Every encoded frame is assigned in the publisher side to one partition P or more depending on the temporal layer it belongs to. By default the subscriber side subscribes to the maximum video stream quality which are assigned to the highest partition P_n of topic V_n; where V_n refers to the topic string "$P.n$" as shown in

Figure 4.3. Every partition in the publisher side has a DataWriter (DW) with a certain QoS. All DWs have the same Best-Effort reliability QoS to maintain a fast transmission for Real-Time video streaming. In the subscriber side, only one built-in DataReader is considered and using the partitioning QoS to switch among different temporal streams. When the subscriber side DR's deadline Qos detects NAL's packets exceeding their playout deadline in a history buffer of GOP size, it directly updates its subscription to another partition $P = P_{n-1}$ just by simply altering the subscribtion topic string wildcard from the default "$P.n$" to "$P.n - 1$" and without the need to send a feedback traffic. As a result, the only partition traffic that is supposed to be transmitted is the one that the subscriber is subscribed to.

Figure 4.4 shows an example of encoded single-layer video with a temporal scalability property. A source video of a CIF resolution (352×288) is encoded to AVC stream of ($IBBBPBBBI$...) GOP sequence. In this example, the group of picture size is 8 and the bit-stream consists of hierarchial B-pictures. The lowest temporal T_0 which only contains I-frames (II...) can be considered as the temporal base layer and assigned to the lowest partition P_0. The lowest temporal plus the next enhancement layer of p-frames which contain I and P-frames (IPI...) are considered as the next temporal layer with better video quality and assigned to partition P_{0+1}. The highest partition P_3 which contains the whole frames ($IBBBPBBBI$...) is considered as the highest temporal layer with the maximum video quality. The result will be four partitions for streaming four different temporal streams.

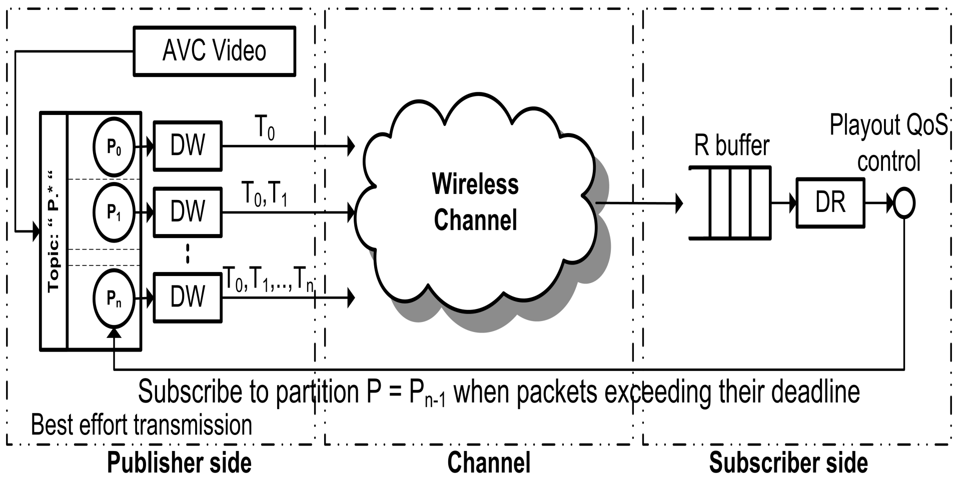


Figure 4.3: RTPS-based video streaming of single-layer video stream with temporal scalability of H.264/AVC.

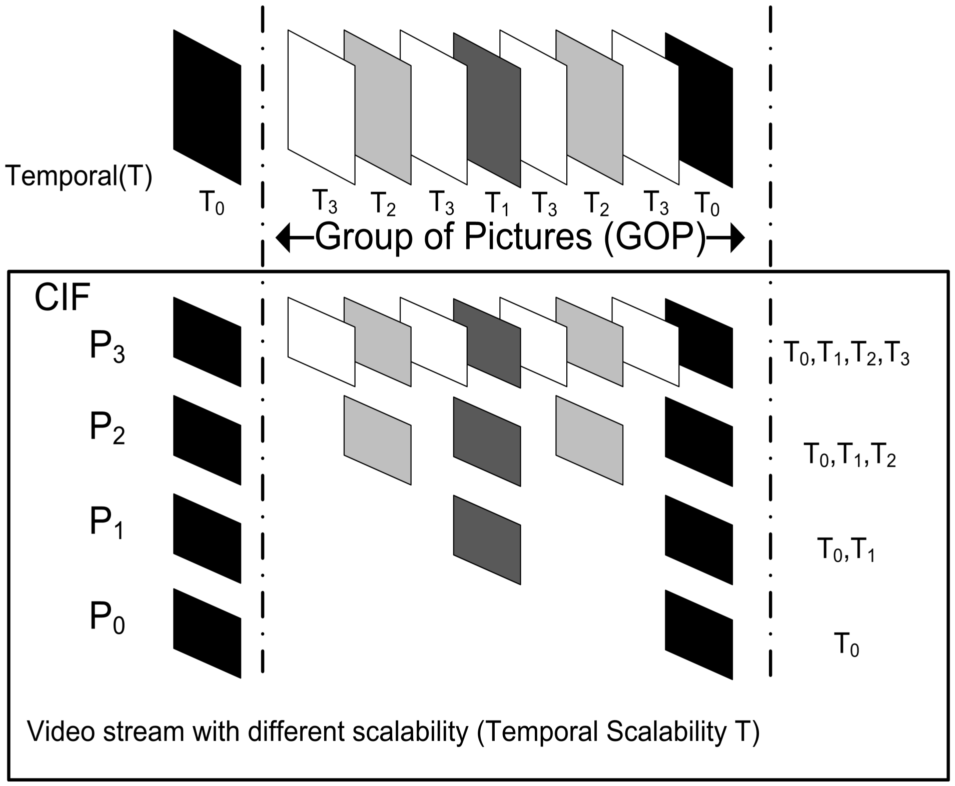


Figure 4.4: Single-layer video stream partitioning example.

The same idea for scalable (multi-layer) video stream H.264 SVC but with supporting various scalabilities, i.e., temporal, spatial, and quality scalabilities. Beside the scalability nature of the SVC video stream, the proposed RTPS-based video streaming scalability enables more control on which layer should be dropped first by using the DDS rich QoS without the need for relay node or dropping packets at the receiver side. As shown in Figure 4.5, all encoded SVC sub-layers are assigned to a certain partition. Figure 4.6 shows an example of multi-layer encoded video with a combined (temporal, spatial and quality) scalability property. In this example, the lowest two sublayer are both with the same temporal and spatial (352×288) properties while the quality is changed from Q_0 to Q_1. Also, the next two sub-layers are both with the same temporal and spatial (176×144) properties but differ than those of the first two sub-layers, and with different qualities Q_0 to Q_1 same as those of the first two sublayers.

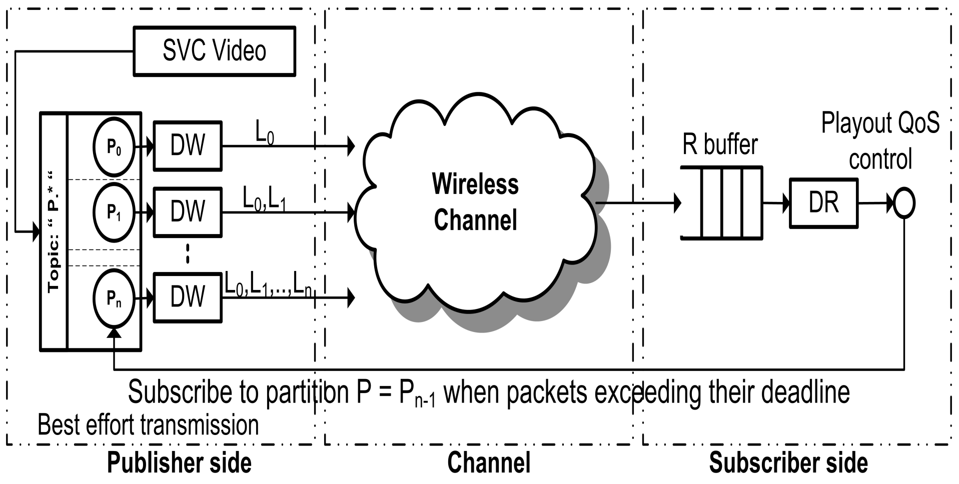


Figure 4.5: RTPS-based video streaming of multi-layer video stream with combined scalability of H.264/SVC.

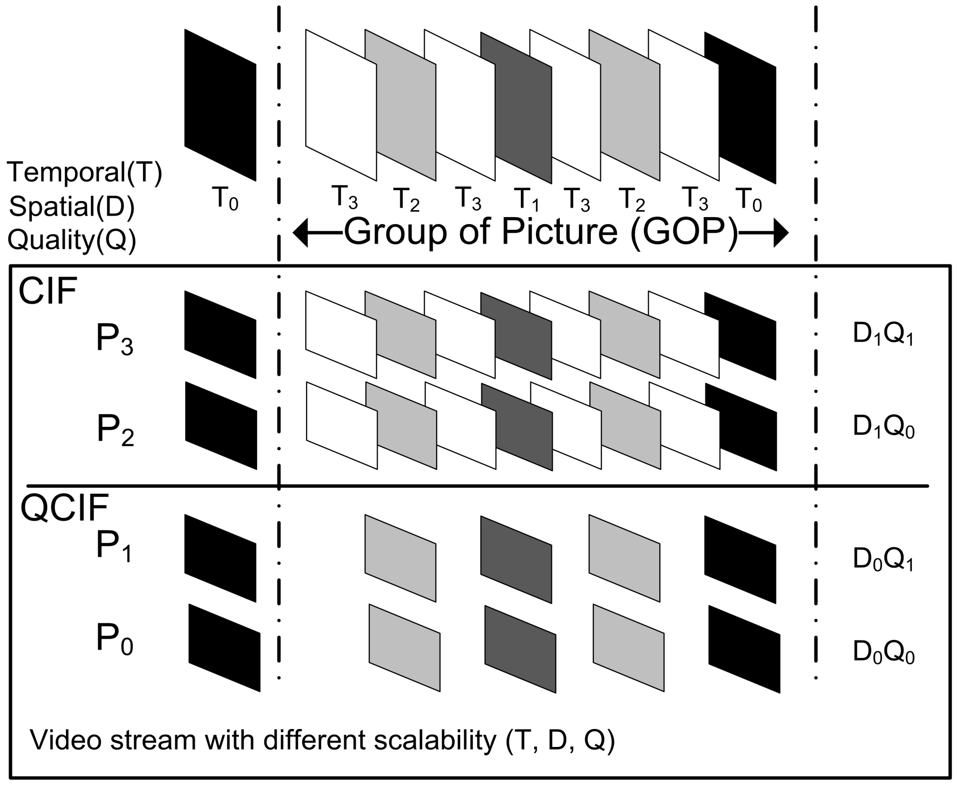


Figure 4.6: Multi-layer video stream partitioning example.

The activity diagrams of both video stream publication and subscription modules in my proposed scalable RTPS-based Real-Time video streaming architecture are shown in Figures 4.7 and 4.8 respectively.

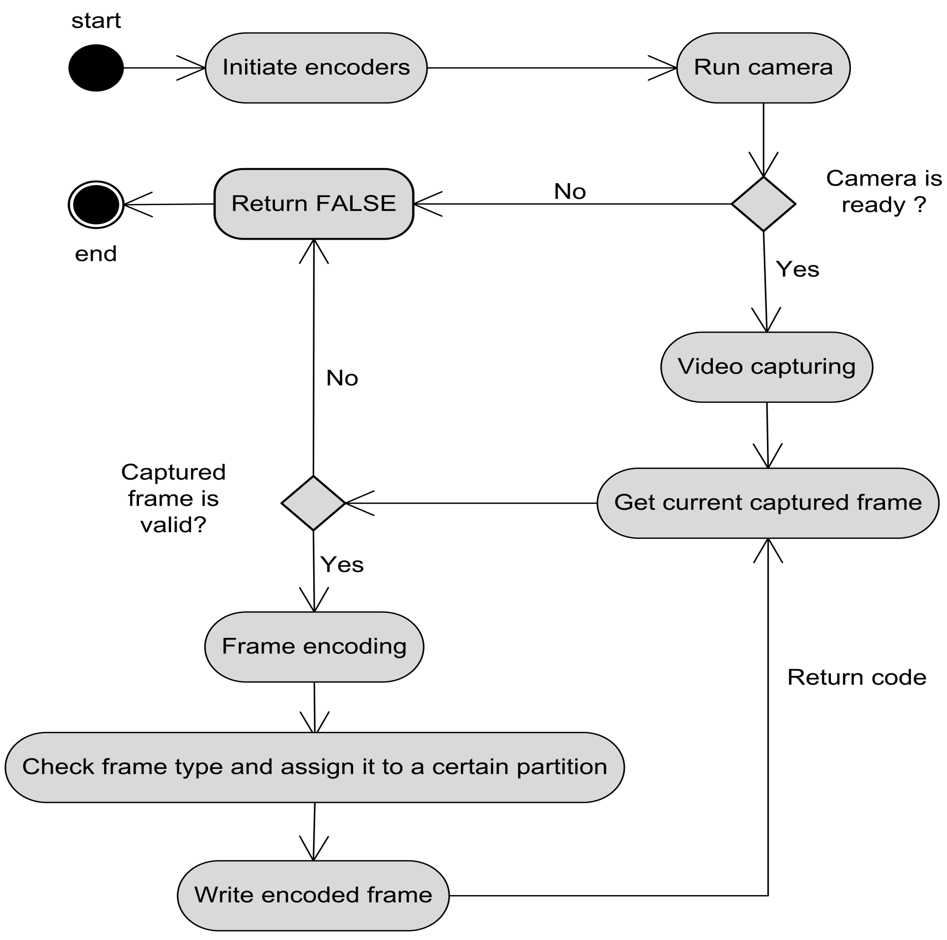


Figure 4.7: Publication activity diagram.

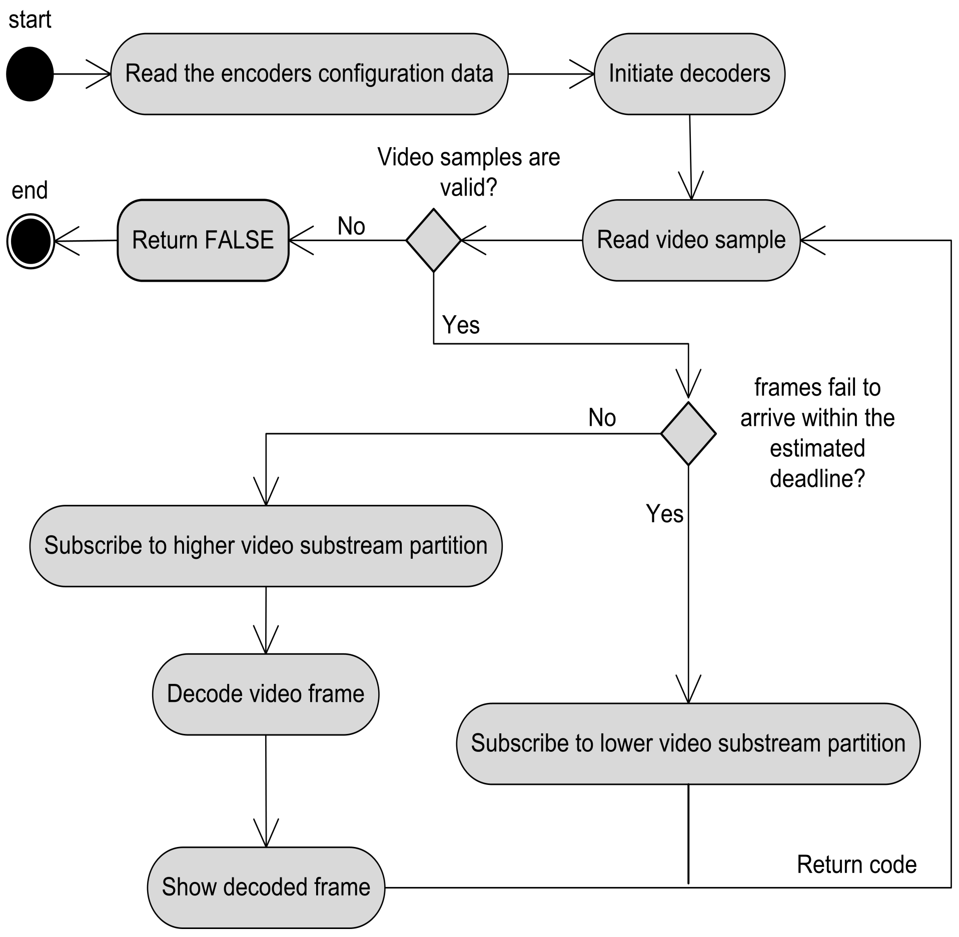

start
Read the encoders configuration data
Initiate decoders
Video samples are valid?
end
Return FALSE
No
Read video sample
Yes
frames fail to arrive within the estimated deadline?
No
Yes
Subscribe to higher video substream partition
Decode video frame
Subscribe to lower video substream partition
Show decoded frame
Return code

Figure 4.8: Subscription activity diagram.

CHAPTER 5

EXPERMINTAL SETUP & PERFORMANCE EVALUATION

This chapter presents the performance and scalability results obtained from transmitting a Real-Time video stream over the Data Distribution Service (DDS)[5] middleware by using the proposed scalable RTPS-based Real-Time video streaming system architecture through IEEE 802.11g WLAN.

5.1 Experimental Setup

The experimental setup involves a test-bed framework of five nodes A to E with a wireless adapter of 54Mbps in each one connected by an access point at 54Mbps as shown in Figure 5.1. The experiment setup is performed indoor to study the effects of transmitting live video in simple conditions. A source video sample is encoded by H.264/SVC encoder into a temporal scalability stream compatible with AVC decoders and transmitted over the proposed scalable RTPS/UDP based video streaming implementation. For comparison purposes, the same source video sample is transmitted over an RTP/UDP based Real-Time video streaming application. The video participant node (A) acts as the video streaming publisher to the other wireless subscriber's participant nodes (B, C, D, and E). Both the publisher and subscriber participants are configured with the packet monitoring tools, Tcpdump and wireshark, in addition to the RTPS and RTP based video streaming applications.

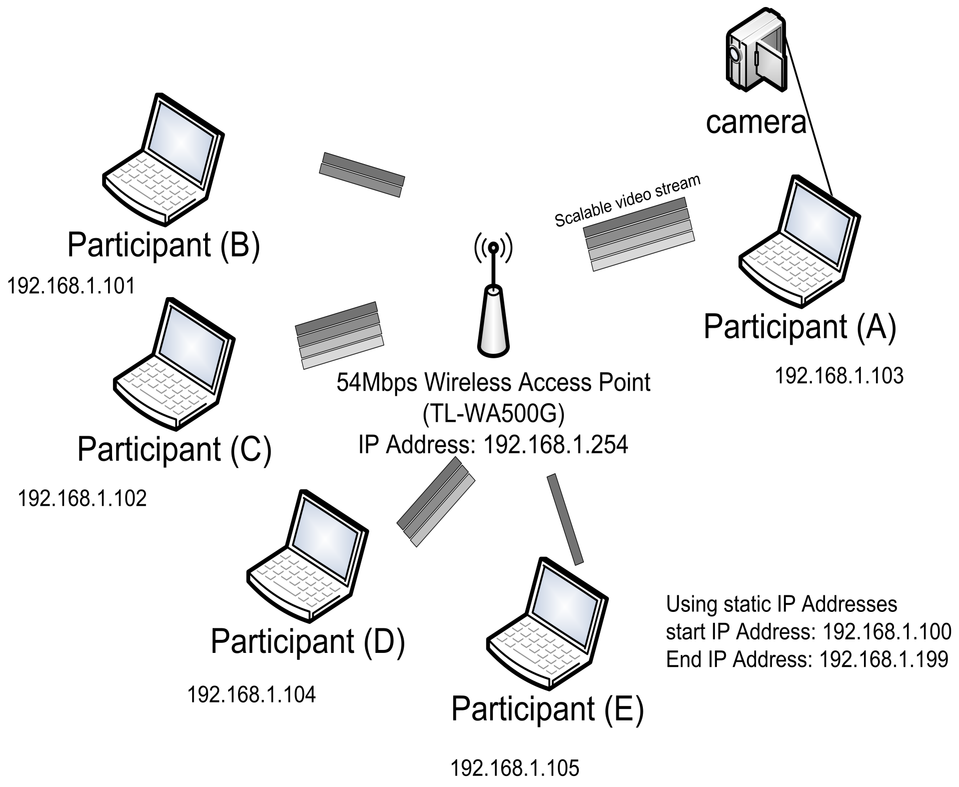


Figure 5.1: Experimental setup topology.

5.1.1 Source video sample

The commonly used *foreman* video test sequence in the 4:2:0 YUV format is used to evaluate the proposed scalable RTPS-based video streaming system. Two foreman video sequences of different Common Intermediate Format (CIF) and frame size are used as shown in Table 5.1.

Table 5.1: Source video samples.

Video sample name	Format	Number of frames
foreman	CIF (352×288)	300
foreman	QCIF (176×144)	300

5.1.2 Evaluation Framework

The Evaluation Video (EvalVid) tool-set [1] is used for evaluating the proposed scalable RTPS-based video streaming architecture. EvalVid enables networking operatives to evaluate the effects of real video streams on proposed network protocols. It basically evaluates the received quality of the transmitted video in a real or simulated network environment. One of the drawbacks of EvalVid is that it only supports single layer video codec like H.264/AVC. That is, a scalable video codec like H.264/SVC is not supported. Fortunately, EvalSVC [7] comes to fill this gap and enabling the evaluation of a scalable video coding. It is capable of evaluating the enhanced features such as: spatial, temporal, SNR, and combined scalability of SVC bitstreams transmited over real or simulated networks. Thus, both evaluation tool sets can be used to evaluate the proposed scalable RTPS-based video streaming system. EvalVid and EvalSVC framework structures for evaluating the transmited AVC and SVC bitstreams over the proposed scalable RTPS-based video streaming system are depicted in Figures 5.2 and 5.3 respectively.

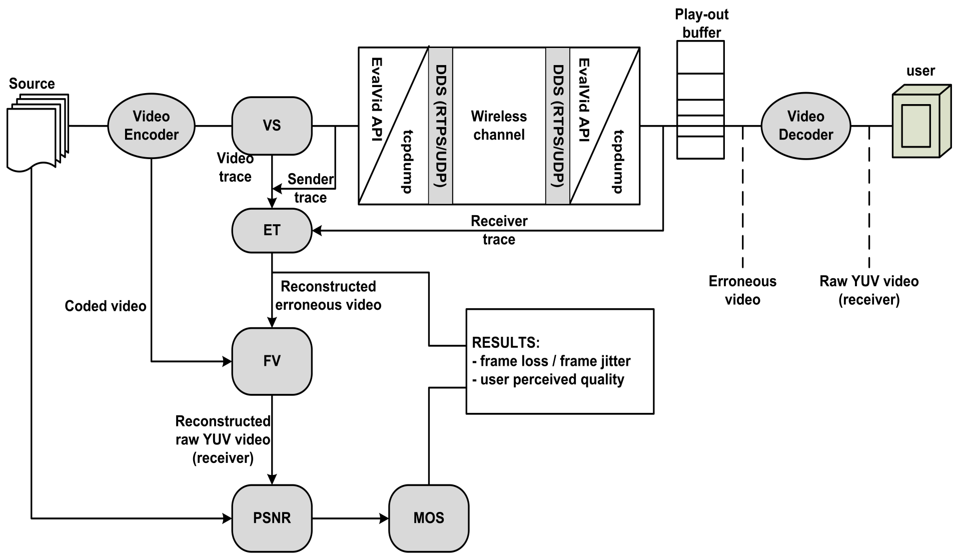


Figure 5.2: EvalVid evaluation framework structure for streaming H.264/AVC bitstream [1].

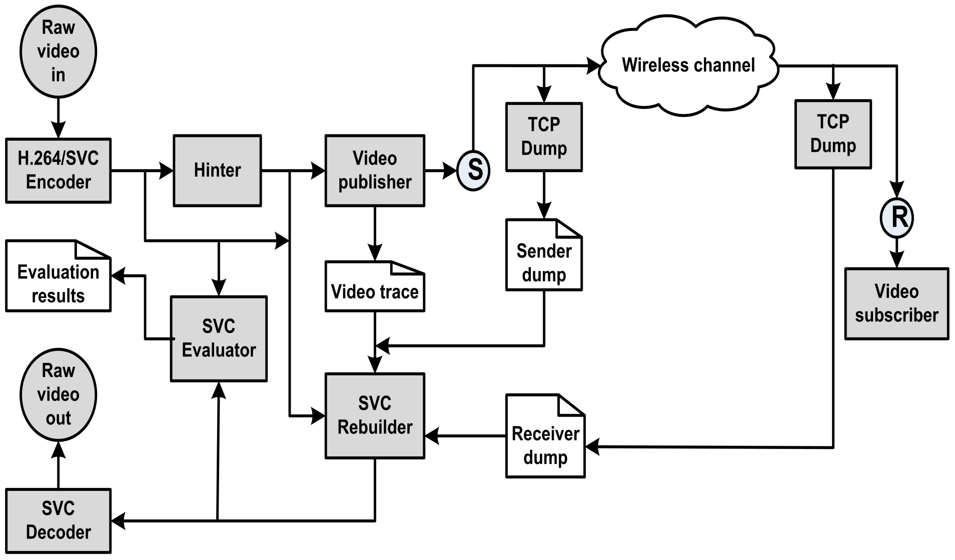


Figure 5.3: EvalSVC evaluation framework structure for streaming H.264/SVC bitstream [7].

In order to evaluate the proposed approach, the following procedures have been conducted in each experimental test:

- An input source video of YUV format is used. YUV video format is acceptable by H.264 AVC and SVC encoders as well as common video capturing devices.

- The source video is encoded by H.264/AVC encoder using X264 [82] and H.264/SVC encoder using JSVM [83].

- The encoded bitstream is encapsulated into MP4 container by using the mp4box tool of the GPAC library [85]. Both H.264 AVC and SVC bitstreams format are supported by mp4box.

- Use the mp4trace tool of EvalVid[1]/EvalSVC[7] for transmitting the encoded and encapsulated H.264 AVC/SVC bitstream over RTP/UDP and using the RTPS-based video publisher to publish the same bitstream over RTPS/UDP protocol out to the network. The output of this step is the sending trace file that contains each frame number, frame size in bytes, and transmission timestamp.

- The Tcpdump network monitoring tool is used to trace the real network traffic at both ends and to form the sender's and receiver's dumping files.
- Rebuilding the transmitted encoded video bitstream from the sender's and receiver's dumping files, video transmission trace file and hinted file. In EvalVid, etmp4 tool is used to rebuild the H.264 AVC bitstream while in EvalSVC, etmp4_SVC is used to rebuild the H.264 SVC bitstream. The video reconstructor is taking into account the missing packets or frames, and have to option for rebuilding such corrupted bitstream. It can truncate the H.264 AVC/SVC video frame or fill that frame with zero (default value). Other QoS measurements of the network such as frame end-to-end delay, frame jitter, frame/packet loss, sender's and receiver's bit-rate will also be generated by etmp4/etmp4_SVC tools.
- Decoding the received and reconstructed H.264 AVC/SVC by the appropriate decoder. For H.264 AVC bitstream, ffmpeg [86] decoder is used to decode such non-scalable single-layer video coding. On the other hand, Joint Scalable Video Model (JSVM) [83] is used to decode the scalable video bitstream of multi-layer video encoding.

- After decoding the received H.264 AVC/SVC video bitstream, both EvalVid and EvalSVC use two video quality evaluation tools: the objective and subjective quality evaluation Peak Signal-to-Noise Ratio (PSNR) and Mean Opinion Score (MOS). Both quality evaluation tools are used to calculate the quality of the decoded H.264 AVC/SVC bitstream by comparing it to the original decoded bitstream.

5.1.3 Performance evaluation metrics

Different QoS measurement metrics of the network such as end-to-end delay, jitter, loss rate, sender's and receiver's bit-rate are going to be measured in order to see the performance of the proposed approach for streaming video over 802.11g WLAN networks. In addition, the video quality measurement metrics such as Peak Signal-to-Noise Ratio (PSNR) and Mean Opinion Score (MOS) are going to be measured.

5.1.3.1 End-to-End delay

Frame end-to-end delay involves the one-way delay at the source encoder, channel transmission and propagation delay, and source decoder delay at the receiver endpoint. The encoding and decoding delay (processing delay) have been excluded since they are out of this research work scope. Encoding and decoding processes are performed separately and in non-real time. Therefore, the measured end-to-end delay is only for the channel transmission and propagation delay of every successive transmitted frame within its playback time constraint from the publisher (sender) endpoint to the subscriber (receiver) endpoint.

5.1.3.2 Jitter

Frame jitter refers to the one-way frame delay variation measurements over time of a series of transmitted frames across the network. Frame jitter is caused by network congestion, time varying network bandwidth, interferences, etc. It severely affects the quality of streaming video. A network with constant latency has no variation (jitter). Packet jitter is expressed as an average of the deviation from the network mean latency.

Based on RFC3550-RTP[87], the cumulative frame jitter is calculated. The cumulative jitter measurement helps to clearly represent the increase in frames jitter over the transmission period. As shown in the equation below, jitter J for frame i is calculated by estimating the difference D for that frame and the previous frame $i-1$ in order of arrival (not necessarily in sequence), according to the following equations:

$$D_{i,j} = (R_j - R_i) - (S_j - S_i) = (R_j - S_j) - (R_i - S_i) \tag{5.1}$$

$$J_i = J_{i-1} + \left(|D_{i-1,i}| - J_{i-1}\right)/16 \tag{5.2}$$

Where S_i is the frame sending timestamp of frame i, and R_i is the time of arrival in timestamp units for frame i. S_j and R_j refere to the next frame sending and receiving timestamps respectively.

These equations are the optimal first-order estimator and the gain parameter

1/16 gives a good noise reduction ratio while maintaining a reasonable rate of convergence.

5.1.3.3 Frame loss rate

Packet loss can be caused by a number of factors including signal degradation over the network channel due to multi-path fading or packet drop because of channel congestion. Frame loss rate refers to the percentage loss of frames that are dropped by such factors or intentionally by the proposed solution for scalability purpose.

5.1.3.4 PSNR

PSNR stands for Peak Signal-to-Noise Ratio. It computes the peak signal-to-noise ratio, in decibels, between two images. This ratio is often used as a quality measurement between the original and a compressed image. The higher the PSNR, the better the quality of the compressed, or reconstructed image.

The Mean Square Error (MSE) and the Peak Signal to Noise Ratio (PSNR) are the two error metrics used to compare image compression quality. The MSE represents the cumulative squared error between the compressed and the original image, whereas PSNR represents a measure of the peak error. The lower the value of MSE, the lower the error.

To compute the PSNR, it first calculates the mean-squared error using the following equation:

$$\mathrm{MSE} = \frac{\sum_{j=1}^{N}\left(\sum_{i=1}^{M}(X_{i,j} - Y_{i,j})^2\right)}{MN} \tag{5.3}$$

In the previous equation, M and N are the number of rows and columns in the input images, respectively. Then the block computes the PSNR using the following equation:

Peak Signal-to-Noise Ratio (PSNR):

$$\text{PSNR} = 10 \log \frac{R^2}{\text{MSE}} \tag{5.4}$$

In the previous equation, R is the maximum signal value that exists in the original "known to be good" image. For example, if the input image has a double-precision floating-point data type, then R is 1. If it has an 8-bit unsigned integer data type, R is 255, etc.. Prior to transmission, it is possible to compute a reference PSNR value sequence on the reconstruction of the encoded video as compared to the original raw video. After transmission, the PSNR is computed at the receiver for the reconstructed video of the possibly corrupted video sequence received. The individual PSNR values at the source or receiver do not mean much, but the difference between the quality of the encoded video at the source and the received one can be used as an objective QoS metric to assess the transmission impact on video quality at the application level [1].

5.1.3.5 MOS

Mean Opinion Score (MOS) is a subjective metric to measure digital video quality at the application level. This metric estimates the human quality impression on a scale that ranges from worst to best. The PSNR evaluation results can be used to estimate the approximated MOS scale using the mapping shown in 5.2.

Table 5.2: PSNR to MOS conversion [1].

PSNR[dB]	MOS
>37	5 (Excellent)
31 37	4 (Good)
25 31	3 (Fair)
20 25	2 (Poor)
<20	1 (Bad)

5.2 Performance Evaluation

For evaluating the proposed scalable RTPS-based video streaming system over error-prone wireless networks, the H.264/SVC encoder is used for producing a temporal scalable video bitstream and then transmits this stream by both RTP/UDP and RTPS/UDP protocols separately in Real-Time. The source video file (foreman) of YUV format in CIF resolution (352×288) and 300 frames is encoded by using JSVM [83] encoder into 30 frames per second and a GOP size of 8 frames. The first frame is intra-coded IDR frame and represents a special GOP while every other GOP consists of a key frame followed by a hierarchically predicted B-frames. The number of temporal scalability levels that can be generated is dependent on the specified GOP size. In this experiment, the generated bitstream provides 4 temporal scalability levels as shown in Table 5.3. The encoded video bitstream is hinted by MP4 container using MP4Box tool of GPAC [85] framework in order to packetize the frames for the transport with RTP and enable the encoded video playout at the receiver side. The maximum transmission packet unit (MTU) is assigned to 1KB.

Table 5.3: Example of supported Temporal Scalability Layers for single-layer coding.

Layer	Resolution	Frame Rate (fps)	Bit Rate	DTQ	In Partition
0	352×288	3.7500	182.4971	(0,0,0)	P_0,P_1,P_2,P_3
1	352×288	7.5000	230.3008	(0,1,0)	P_1,P_2,P_3
2	352×288	15.0000	276.8912	(0,2,0)	P_2,P_3
3	352×288	30.0000	327.0816	(0,3,0)	P_3

For transmitting the encoded video bitstream over RTP/UDP, the mp4trace tool from EvalSVC[7] is used to send the hinted mp4 file to a unicast and multicast destination IP in four different scenarios; one publisher to one subscriber, 6 subscribers, 12 subscribers and 18 subscribers. On the other hand, the proposed RTPS-based video streaming implementation is used to transmit the same encoded and hinted video sample over RTPS/UDP protocol using the same scenarios.

In each scenario, the tcpdump network monitoring tool is used to trace the IP packets at the sender and receiver during the transmission process. The result are sender and receiver trace files to be used later by EvalSVC[7]. In every experimental scenario five main files are used for the evaluation purpose; the YUV source file before and after the encoding, the encoded and encapsulated mp4 video file, the sender and receiver UDP packets trace file, and the transmission sender trace file which contains information about the frame type, packet size, packet space (segmentation), and transmission timestamp in milliseconds.

To obtain the PSNR values I compare the encoded video at the sender side with PSNR values of the receiver side. The results show that RTP and RTPS have nearly the same performance in 1 to 1 unicast scenario as well as in 1 to 6 multicast scenario as shown in Figures 5.4 and 5.5. However, when the numbers of receivers (subscribers in RTPS) increased as shown in Figures 5.6 and 5.7, RTP shows a continuous degradation in video quality, while RTPS was able to maintain a stable level of video quality with no interruptions. This is because subscribers in the proposed scalable RTPS-based video streaming approach are adaptively subscribing to lower video partition when network degradation is detected and vise versa. Thus, RTPS-based approach intentionally drops some upper enhancement layer packets to protect video stream from a severe packet loss for maintaining a continuous video stream. Figure 5.8 shows video snapshot of the received video in every video transmission session over both RTP/UDP and RTPS/UDP protocols, while Table 5.4 shows the Mean Openion Score (MOS) for every video transmission session.

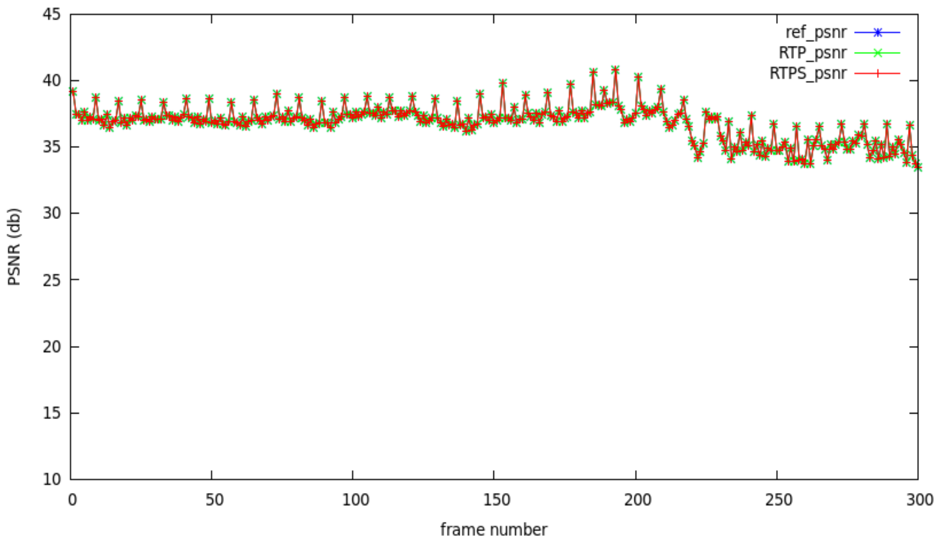


Figure 5.4: PSNR (1 to 1).

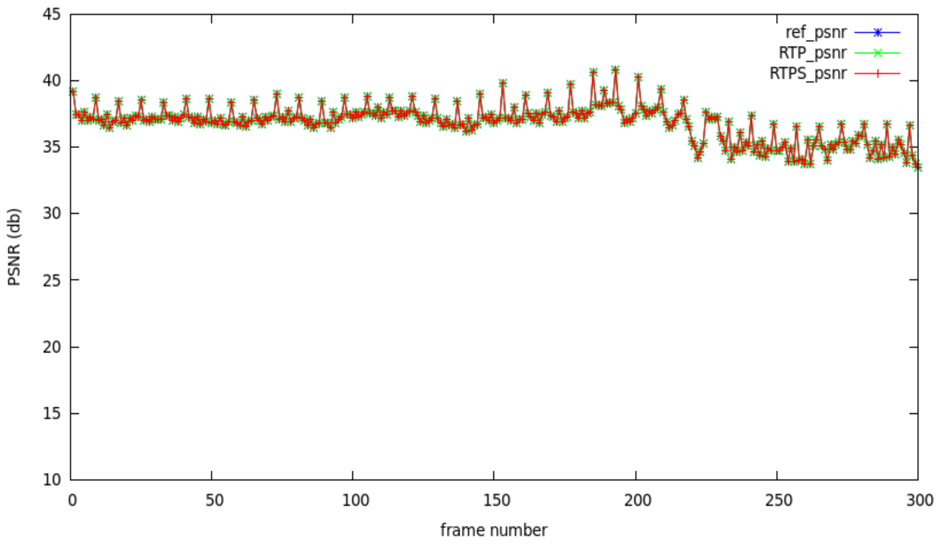


Figure 5.5: PSNR (1 to 6).

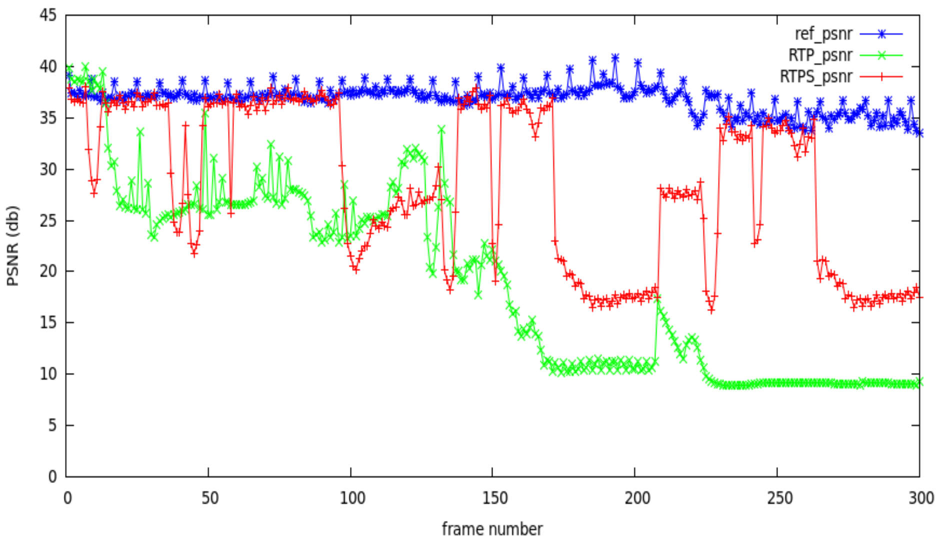


Figure 5.6: PSNR (1 to 12).

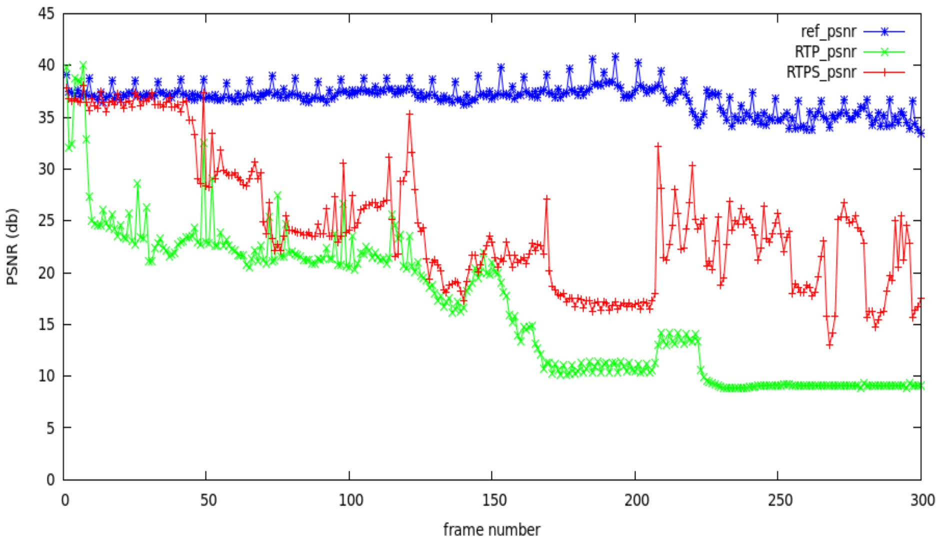


Figure 5.7: PSNR (1 to 18).

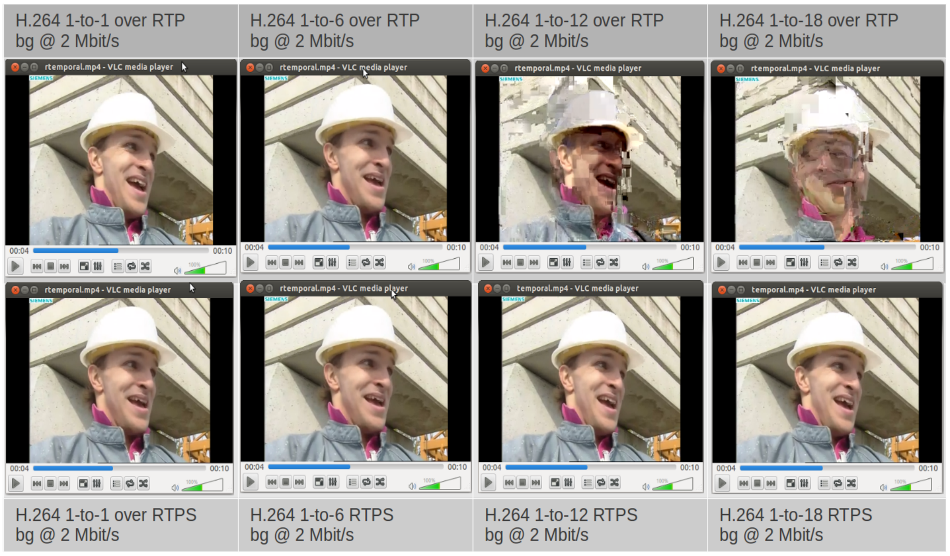


Figure 5.8: Video snapshots with different background traffic throughput.

Table 5.4: Mean Openion Score.

Type	Pub/Sub	PSNR[dB] Average	MOS (5-1)
RTP	1-to-1	36.85	4 (Good)
	1-to-6	36.85	4 (Good)
	1-to-12	23.22	2 (Poor)
	1-to-18	19.87	1 (Bad)
RTPS	1-to-1	36.85	4 (Good)
	1-to-6	36.85	4 (Good)
	1-to-12	31.07	3 (Fair)
	1-to-18	26.34	3 (Fair)

End-to-end delay for both RTP-based/RTPS-based 1 to 1 and 6 scenarios show a low video frame latency as shown in Figures 5.9 and 5.11. However, RTP-based video streaming records a little bit latency increase in some video frames which reflects an increase in the accumulative jitter as demonstrated in Figures 5.10 and 5.12. This increase in frames latency became worse when the number of subscribers increased to 12 and 18 subscribers as represented in both Figures 5.13 and 5.15. Consequently, RTP-based video streaming shows a highly increase in its cumulative jitter due to the burst packet loss from the upper and lower temporal enhancement layers as shown in Figures 5.14 and 5.16. Unlike RTP-based video streaming, the scalable RTPS-based video streaming reconstructed video is almost clear of significant jerks. It is worth here to mention that for any dropped frame the latency value is filled out by zero for representing the droped/lost packets in the plotted charts.

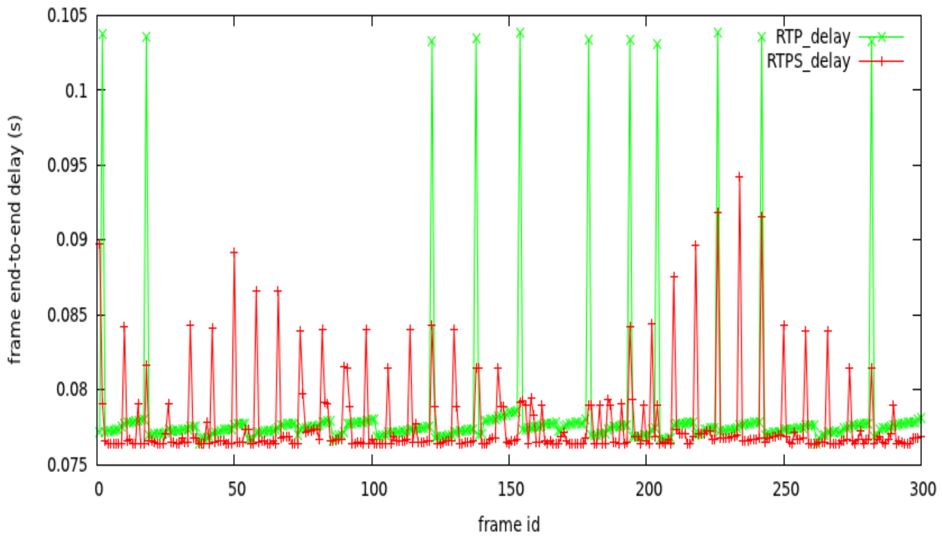


Figure 5.9: Frame End-to-End delay (1 to 1).

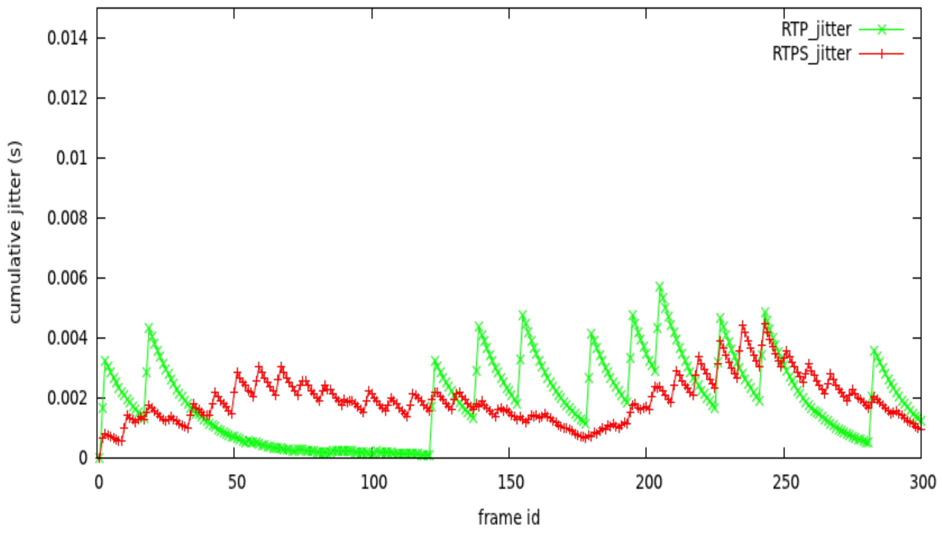


Figure 5.10: Cumulative jitter (1 to 1).

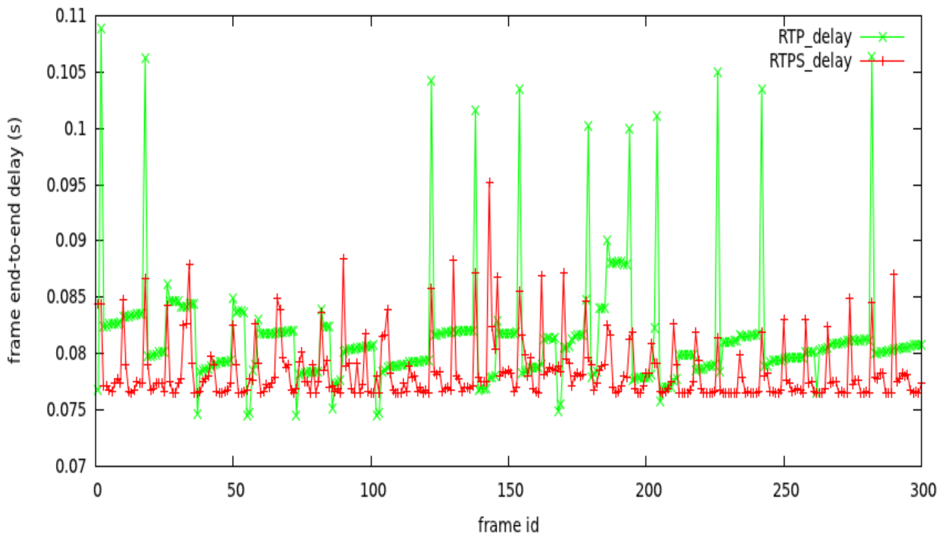


Figure 5.11: Frame End-to-End delay (1 to 6).

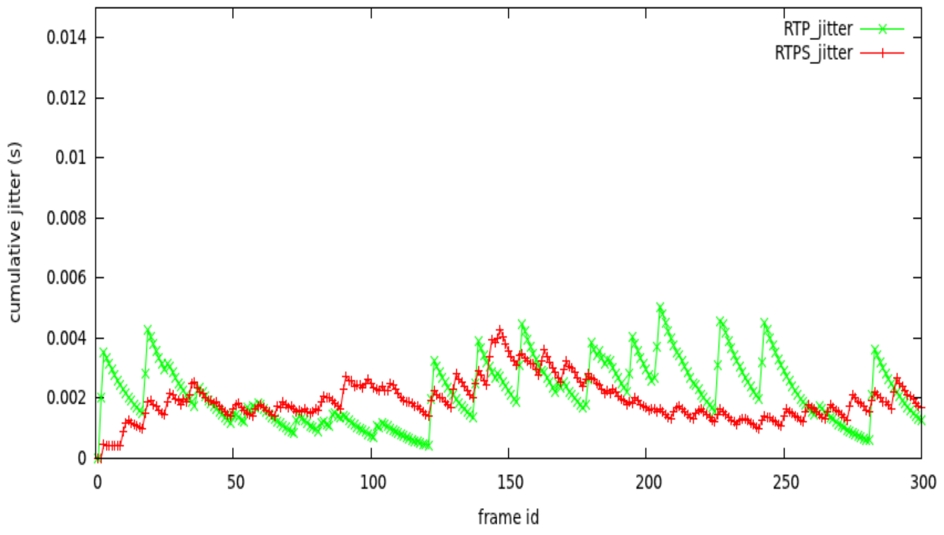


Figure 5.12: Cumulative jitter (1 to 6).

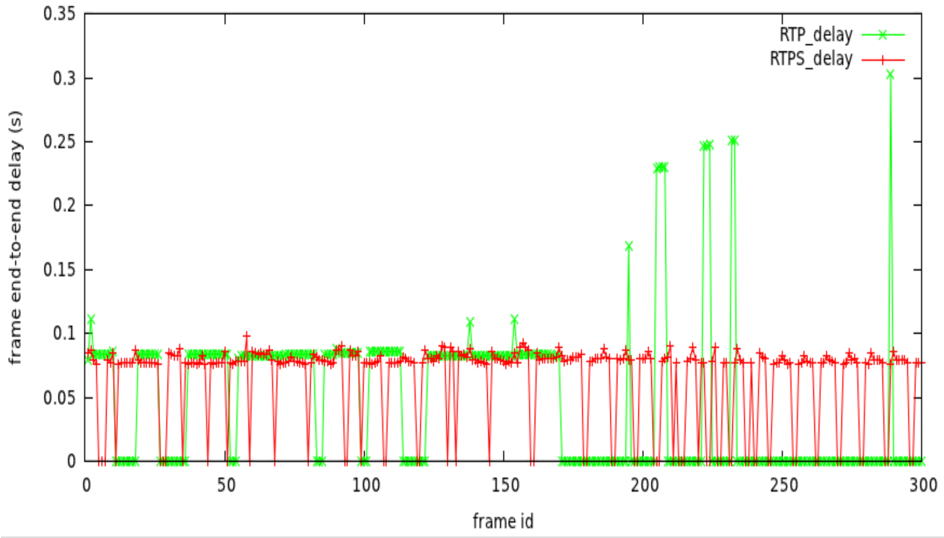


Figure 5.13: Frame End-to-End delay (1 to 12).

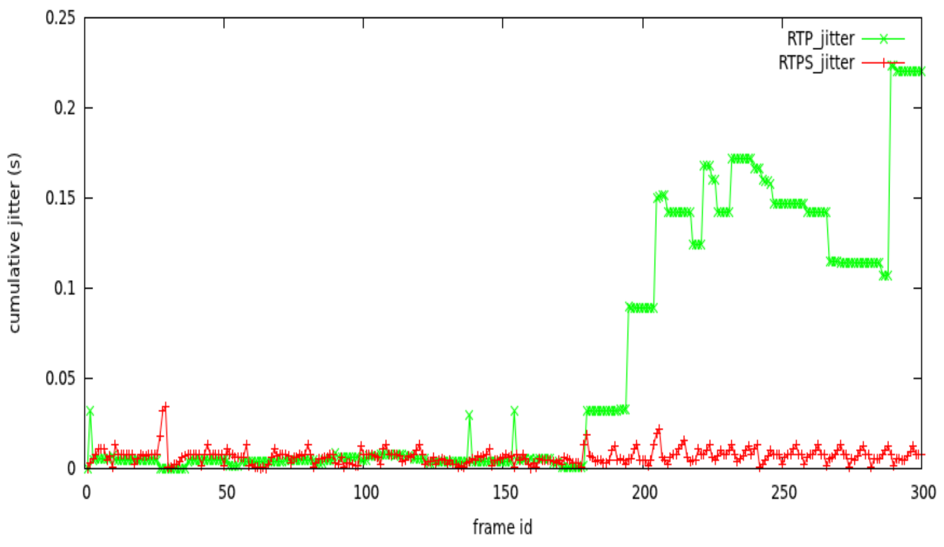


Figure 5.14: Cumulative jitter (1 to 12).

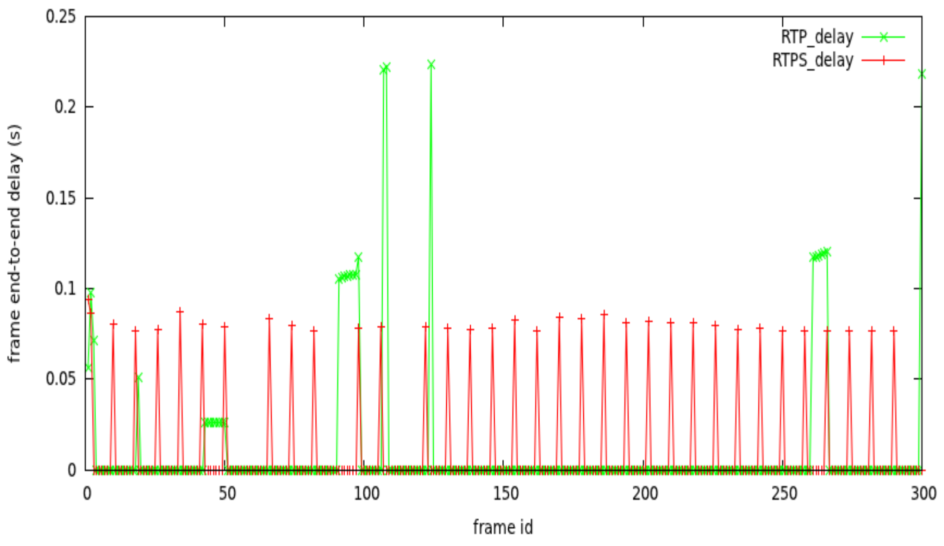


Figure 5.15: Frame End-to-End delay (1 to 18).

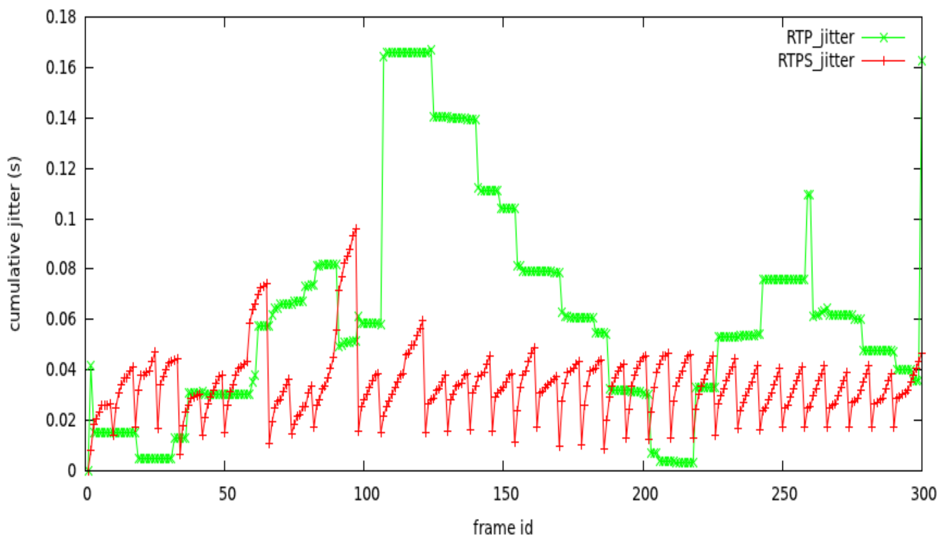


Figure 5.16: Cumulative jitter (1 to 18).

It is clear that RTP-based video streaming wasn't able to protect packets of the lower enhancement layers from burst packet loss when the number of receivers/subscribers increased. On the other hand, the proposed scalable RTPS-based video streaming approach was able to keep up a continuous but with a low quality video flow by intentionally dropping some enhancement packets to protect the lower and/or base temporal layer.

Figures from 5.17 to 5.24 show the sent and received video frames bit-rate (in kbps) in both RTP-based and RTPS-based video streaming scenarios. The results show that the proposed scalable RTPS-based Real-Time video streaming approach maintains a low bit-rate due to its adaptive and scalable video streaming QoS-based control mechanism. Moreover, the average throughput for all received packets shows that RTP-based video streaming severely loss its throughput when the number of video receivers increased to 12. This is because RTP wasn't able to tolerate the congestion and time-varying channel bandwidth, which consequently leads to excessive packet loss. On the other hand, the proposed scalable RTPS-based Real-Time video streaming approach keep up with a high throughput and only began to loss throughput gradually when the number of subscribers increased to 18 due to the intentionaly dropped packets as shown in Figure 5.25.

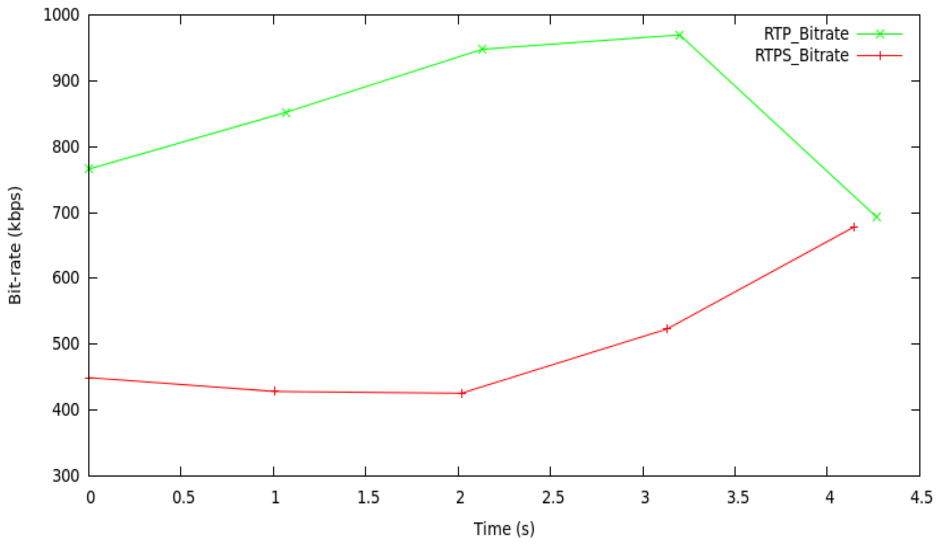


Figure 5.17: Sent Bitrate (1 to 1).

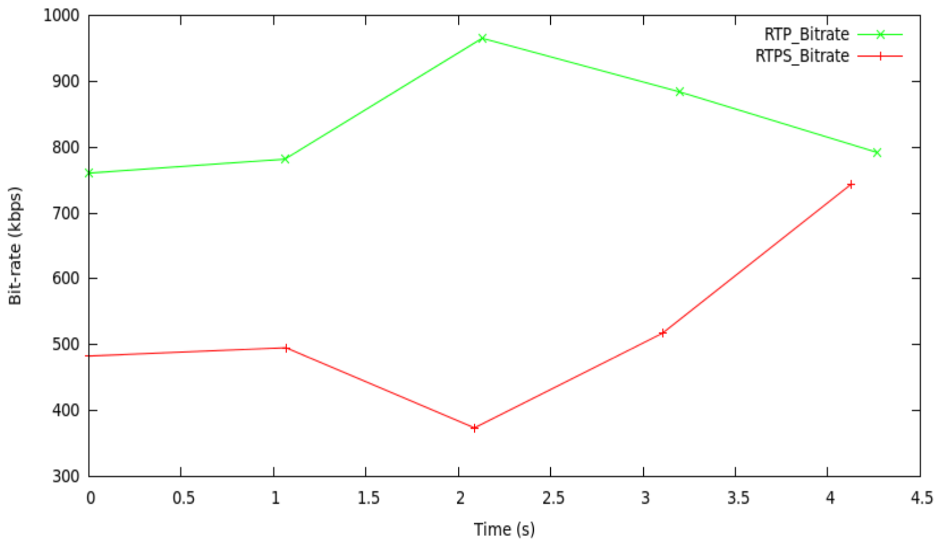


Figure 5.18: Received Bitrate (1 to 1).

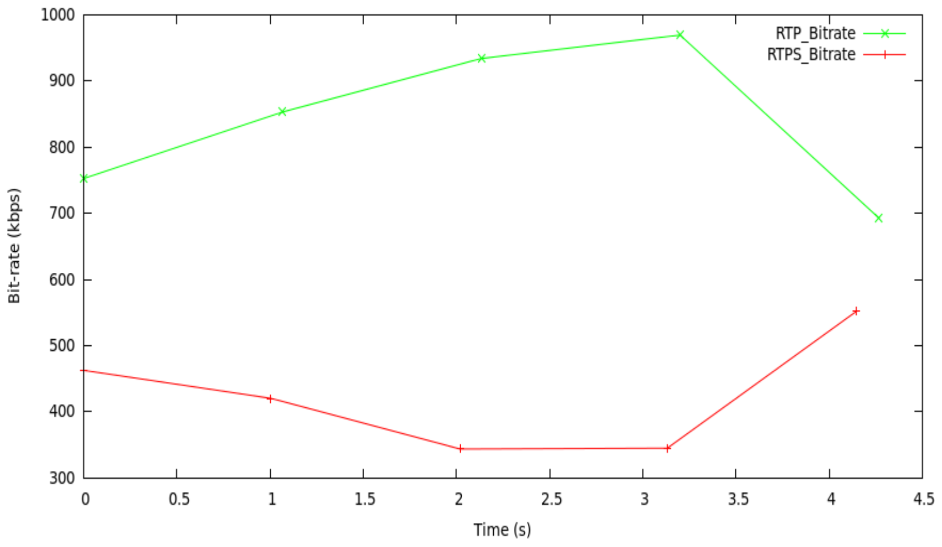


Figure 5.19: Sent Bitrate (1 to 6).

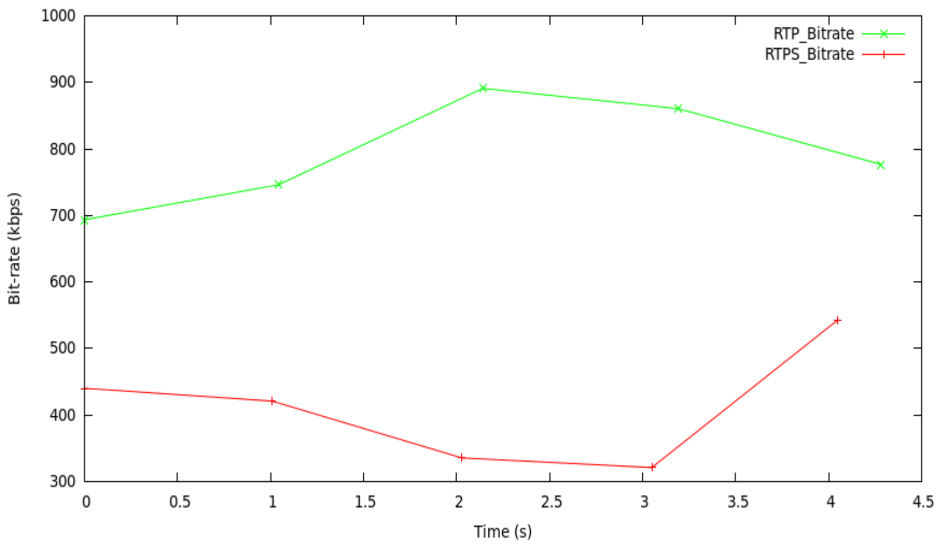


Figure 5.20: Received Bitrate (1 to 6).

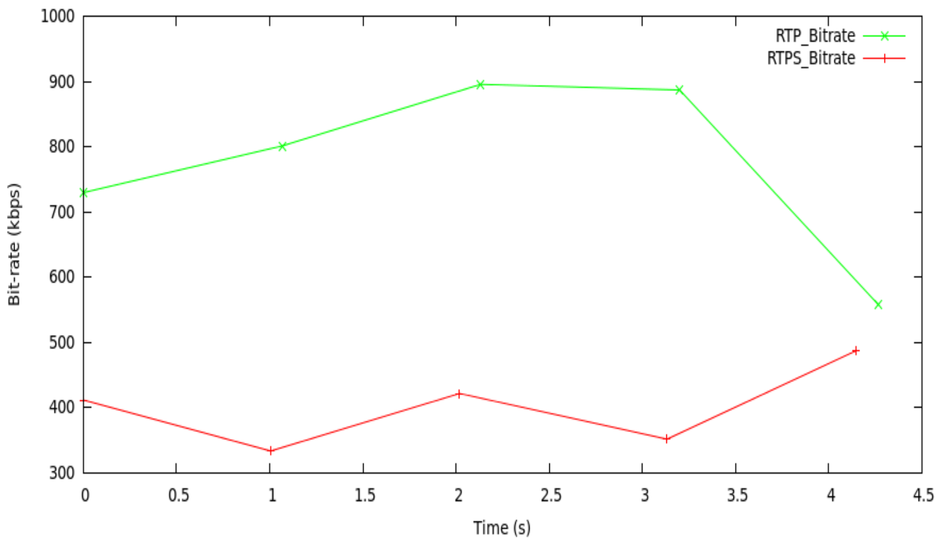


Figure 5.21: Sent Bitrate (1 to 12).

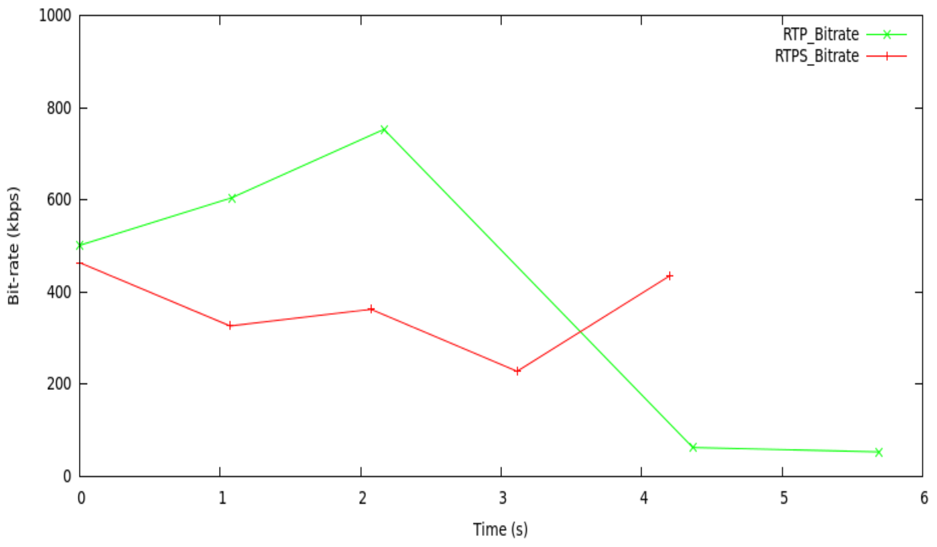


Figure 5.22: Received Bitrate (1 to 12).

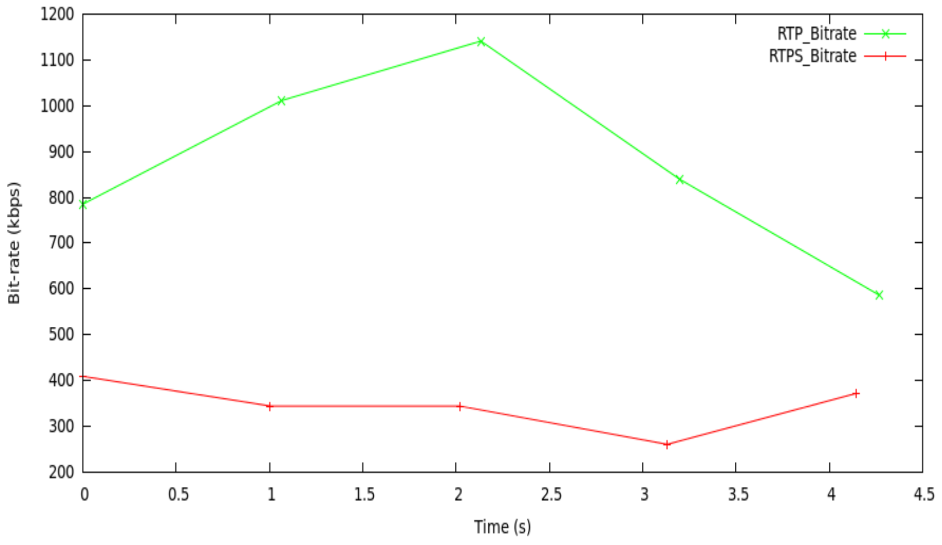


Figure 5.23: Sent Bitrate (1 to 18).

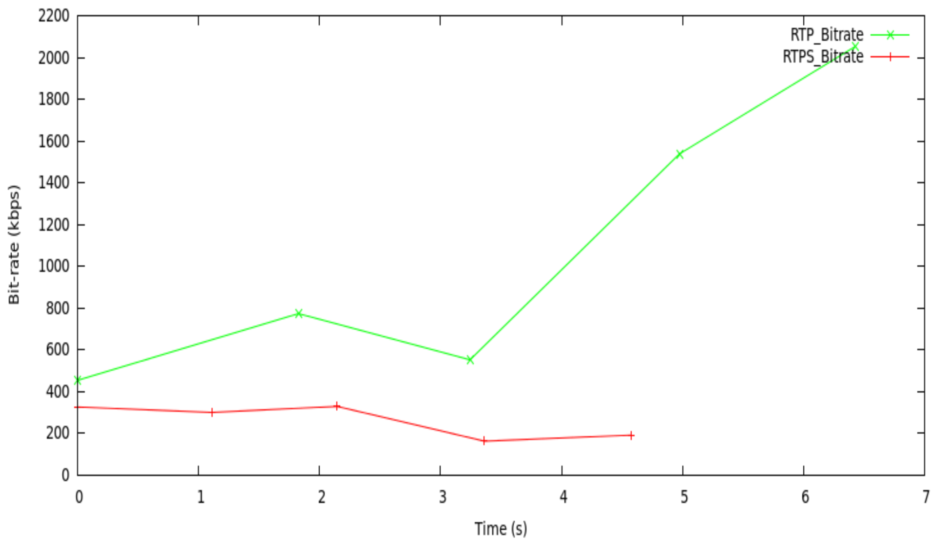


Figure 5.24: Received Bitrate (1 to 18).

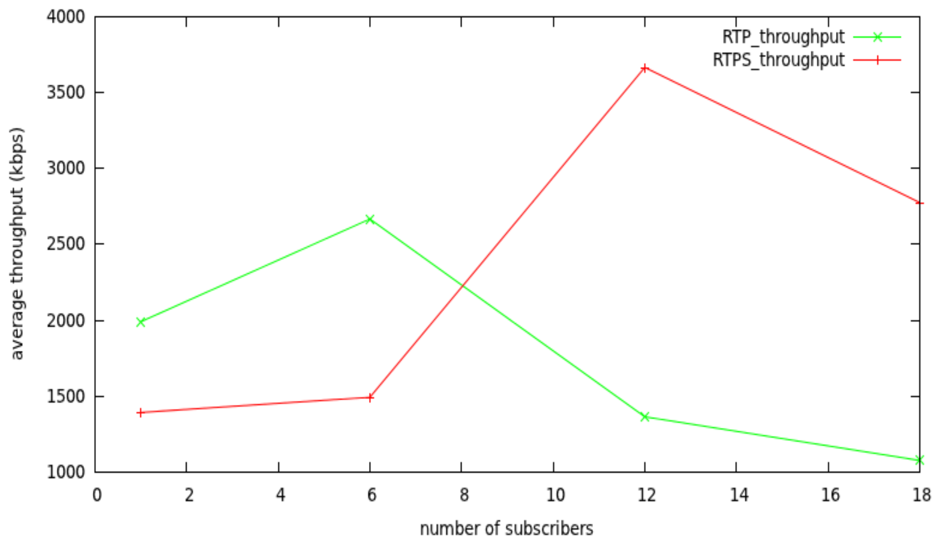


Figure 5.25: Received average throughput.

CHAPTER 6

CONCLUSION & FUTURE WORK

Real-Time video streaming over wireless networks faces challenges of time-varying packet loss rate and fluctuating bandwidth. Frames must be delivered and decoded by its playback time. Recently, layer coding (LC) enables Real-Time and scalable video streaming to clients of heterogeneous capabilities by dropping upper enhancement layers without the need of re-encoding. However, layer coding still facing unfair layer protection problem in which packets from the base or lower layers might be dropped while there is a chance to drop packets from the upper enhancement layers. Loosing packets from the base layer can significantly affect the delivered video quality and sometimes lead to an interruption especially in error-prone networks as wireless networks. In this thesis, I investigate the behaviour of video streaming over Real-Time publish-subscribe based middleware. I propose and develop an unequal layer protection mechanism for Real-Time video streaming based on the Data Distribution Service (DDS) middleware, and show the performance of my approach over IEEE 802.11g WLAN networks. my approach shows a graceful degradation of video quality while maintaining a robust video streaming free of visible error or interruptions.

For future work, the proposed approach needs to be evaluated under more efficent video scalabilities (spatial, SNR, or Combined) than temporal scalability and with larger frame resolution than (352×288). In addition, the proposed approach requires to be evaluated in multi-hob wireless networks such as ad hoc networks, sensor networks and mesh networks. An efficient partition switching mechanism is also required to decrease the number of dropped frames due to frequent switching among partitions at the subscriber side.

Appendices

APPENDIX A

DATA COLLECTION APPROACH

Study of Real-Time video streaming over wireless has been associated with obstacles and challenges. Limited and inefficient utilization of network resources, or inadaptability to network changes severely affect the streaming of video over wireless networks. Therefore, the provisioning of end-to-end QoS while transmitting video stream packets is required to keep video playback within its deadline constraints.

In this research, video streaming over Real-Time publish-subscribe based middleware has been investigated. A development of Real-Time video streaming implementation based on publish-subscribe middleware is a key point on this thesis work. An empirical study has been conducted in order to study the performance of publish-subscribe based video streaming over WLAN. The scalable RTPS-based video streaming implementation that has been developed in this thesis work is used for transmitting a live video show over IEEE 802.11g for a period of time from one publisher to one or more subscribers. The number of subscribers are increased over the time to study the scalability and performance of publish-subscribe based video streaming under a normal video conferencing sessions. Moreover, video streaming performance and quality have been measured under different quality of services (e.g., persistence, durability, reliability, etc) in order to estimate the best practice combination of QoSs for 802.11 WLAN scenarios. A comparison of the proposed approach to other solutions has also been conducted.

A.0.1 Implementation Components & Libraries

For implementing the scalable RTPS-based video streaming application, the following open source components and libraries have been used:

A.0.1.1 X264 Encoder

X264[82] is a free software library and application for encoding video streams into the H.264/MPEG-4 AVC format. It is released under the terms of GNU General Public License. X264 library is used in the implementaion for encoding the captured video show and convert this video into a single-layer video stream of an H.264/AVC format.

A.0.1.2 JSVM Encoder/Decoder

The JSVM[83] (Joint Scalable Video Model) software is the reference software for the Scalable Video Coding (SVC) project of the Joint Video Team (JVT) of the ISO/IEC Moving Pictures Experts Group (MPEG) and the ITU-T Video Coding Experts Group (VCEG). JSVM library is used in the experimental work for encoding the source video to multi-layer scalable video stream of H.264/SVC format.

A.0.1.3 ffmpeg Libavcodec

FFmpeg[86] is a complete, cross-platform solution to record, convert and stream audio and video. It includes libavcodec; the leading audio/video codec library. Libavcodec library of ffmpeg is used in the experimental work for decoding the transmitted video of H.264/AVC format to the decoded YUV format in order to be used for the PSNR video quality estimation.

A.0.1.4 OpenCV

OpenCV[84] (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. OpenCV library is used in the implementation for capturing video frames of YUV format directly from the camera device in order to be used by the video encoder module.

A.0.1.5 GPAC MP4Box

GPAC[85] is an Open Source multimedia framework. It provides three sets of tools based on a core library called libgpac: a multimedia player, called Osmo4 / MP4Client, a multimedia packager, called MP4Box, and some server tools included in MP4Box and MP42TS applications. MP4Box can be used for manipulating ISO files like MP4, 3GP: adding, removing, multiplexing audio, video and presentation data (including subtitles) from different sources and in different formats. MP4Box library backage is used in the expermintal work for creating ISO MP4 files containing the video samples (frames) and a hint track which describes how to packetize the frames for the transport with RTP.

A.0.1.6 EvalVid and EvalSVC

EvalVid[1] is a framework and tool-set for evaluation of the quality of video transmitted over a real or simulated communication network. It is targeted for researchers who want to evaluate their network designs or setups in terms of user perceived video quality. Besides measuring QoS parameters of the underlying network, like loss rates, delays, and jitter, standard video quality metrics like PSNR and SSIM and a subjective video quality evaluation metric of the received video are provided. Evalvid now supports standard MPEG4-H.264/AVC. However, it does not support the H.264 SVC. Fortunately, EvalSVC[7] comes to fill this gap and enabling the evaluation of a scalable video coding. It is capable of evaluating the enhanced features such as: spatial, temporal, SNR, and combined scalability of H.264 SVC bitstreams transmitting over real or simulated networks. Both EvalVid and EvalSVC are used in the experimental work for evaluating the performance and quality of the proposed scalable RTPS-based video streaming approach.

A.0.1.7 RTI-DDS Middleware

RTI Data Distribution Service is communications middleware for distributed Real-Time applications. It is the most reliable, flexible and highest performing implementation of the Object Management Groups (OMG) Data Distribution Service for Real-Time Systems (DDS)[5] standard available today. RTI Data Distribution Service (formerly NDDS) is fielding proven and is used in many time-critical and data-critical applications such as: National railways, Air traffic control, Traffic monitoring, Mission-critical combat systems, financial transaction processing and Industrial automation.

RTI Data Distribution Service is operating system and programming language agnostic, allowing heterogeneous systems to communicate easily with each other. System designers can connect multiple different physical connection points using a pluggable transport framework. Transports can include Ethernet network interfaces, shared memory, back-plane interfaces and various other connection mediums. Many Quality of Service (QoS) parameters are available in RTI-DDS. This rich set of the supported QoS allows designers to tune their application for the best combination of performance and resource usage.

RTI Data Distribution Service is a layer of software that sits on top of a networking stack. It simplifies the underlying low-level network code with a common, standards-based Application Programming Interface (API) that provides an easily understood publish-subscribe model of communication. This model defines two fundamental actors: Publishers, which create data, give the data a name (called a topic) and make that data available to interested subscribers, and Subscribers, which register interest in a topic to receive the data whenever it is available. Any node can be a publisher, a subscriber, or both simultaneously, of many different topics. RTI Data Distribution Service handles all the network I/O and management needed for reliable and transparent transfers: message addressing, data marshalling, and unmarshalling, flow control, retries, etc. No application intervention is needed.

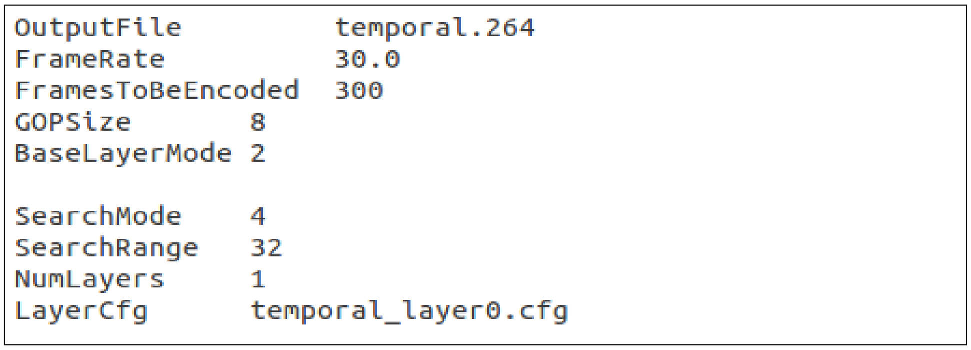
RTI-DDS middleware is selected to implement the proposed scalable RTPS-based video streaming architecture due to three main reasons: (1) the use of publish subscribe communication model in its architecture, (2) its outstanding reputation in Real-Time and critical mission systems, and (3) the rich amount of QoS it uses and its flexibility in using these QoS to control data. The RTI-DDS APIs have been used for implementing the proposed scalable RTPS-based video streaming tool. Moreover, C/C++ programming language is used due to its portability and high performance.

A.0.2 Data Collection

In order to collect the data for performance evaluation, the following step by step procedures have been carefully performed:

A.0.2.1 Video Encoding

For evaluation purpose the encoded video sample has been encoded in non-Real Time, this is because the encoding and decoding latency measurements are out of this thesis research scope. In this step JSVM[83] encoder has been used for generating a temporal scalable video sample. The source video file (foreman) of YUV formate and CIF resolution (352×288) with a frame rate of 30 Hz and a total of 300 frames has been encoded into a 30 frames per second and a GOP length of 8 frames. The main layer configuration files is depicted in Figure A.1.

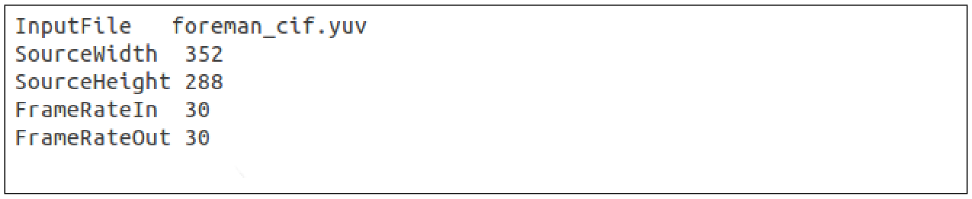


```
OutputFile          temporal.264
FrameRate           30.0
FramesToBeEncoded   300
GOPSize        8
BaseLayerMode  2

SearchMode     4
SearchRange    32
NumLayers      1
LayerCfg       temporal_layer0.cfg
```

Figure A.1: Main Encoding Configuration file for temporal scalability.

The most important parameters that need to be specified in the main configuration file are the *OutputFile*, the frame rate *FrameRate*, the number of frames to be encoded *FramesToBeEncoded*, the GOP size *GOPSize*, and the base layer mode *BaseLayerMode*. The parameter *BaseLayerMode* has been set to 2 hence an H.264 AVC compatible bitstream with additional sub-sequence SEI messages is written to be used for the extraction of a temporal sub-stream in an non-SVC encoders like X264[82] and can be set to 0 or 1 for single-layer coding and supporting AVC decoding only. In the layer configuration file, the filename of the input sequence *InputFile* has been specified to main source file *foreman* of (352×288) resolution with frame rate of 30 Hz as shown in Figure A.2.



```
InputFile    foreman_cif.yuv
SourceWidth  352
SourceHeight 288
FrameRateIn  30
FrameRateOut 30
```

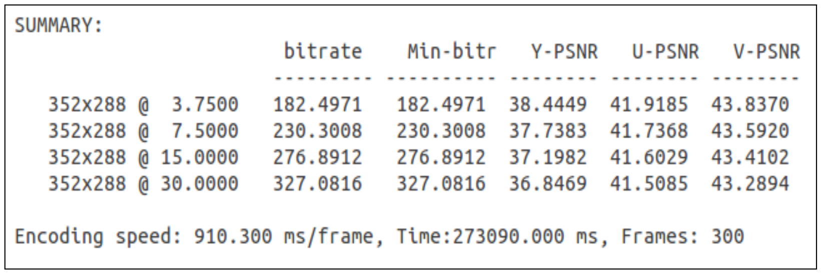
Figure A.2: Layer Encoding Configuration file for temporal scalability.

Given the configuration files in Figure A.1 and Figure A.2, the following JSVM[83] command has been used to encode the main sourece video sample:

```
> H264AVCEncoderLibTestStatic -pf temporal_main.cfg -lqp 0 30
```

The final summary of the encoded output video sample is shown in Figure A.3. Encoding summary shows that in this encoding scenario, only a single spatial resolution of 352×288 samples is supported. But the bit-stream provides 4 different temporal resolutions with frame rates of 3.75, 7.5, 15, and 30 Hz. A printout of some encoded frames is shown in Figure A.4. The encoded video samples (frames) have been encapsolated in an MP4 container format with a packetize hint track for supporting the transport with RTP by using MP4Box of GPAC[85] as shown in the following command:

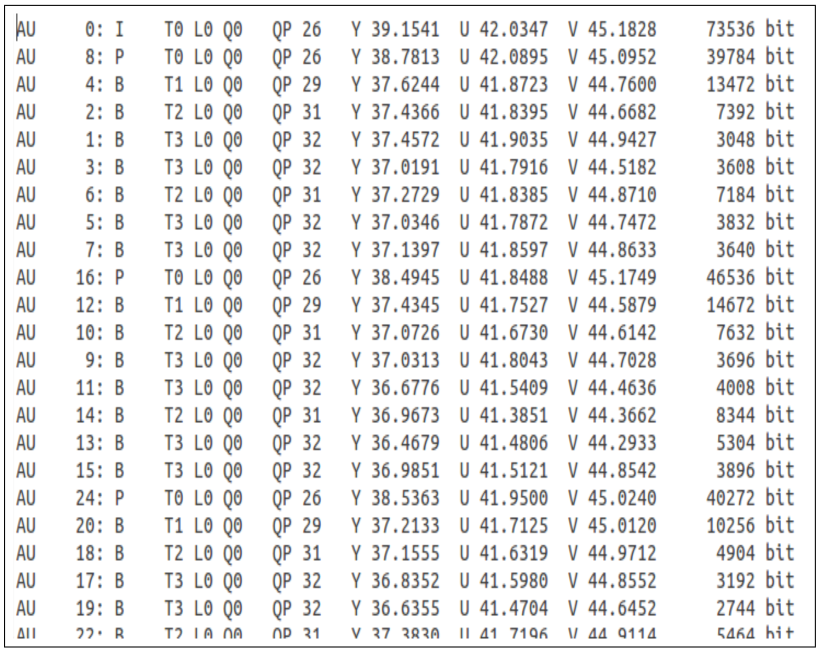
```
> MP4Box -hint -mtu 1024 -fps 30 -add temporal.264 temporal.mp4
```



```
SUMMARY:
                         bitrate    Min-bitr   Y-PSNR   U-PSNR   V-PSNR
                        --------- ---------- -------- -------- --------
   352x288 @  3.7500    182.4971   182.4971  38.4449  41.9185  43.8370
   352x288 @  7.5000    230.3008   230.3008  37.7383  41.7368  43.5920
   352x288 @ 15.0000    276.8912   276.8912  37.1982  41.6029  43.4102
   352x288 @ 30.0000    327.0816   327.0816  36.8469  41.5085  43.2894

Encoding speed: 910.300 ms/frame, Time:273090.000 ms, Frames: 300
```

Figure A.3: Encoding Summary for temporal scalability.



```
AU     0: I    T0 L0 Q0   QP 26   Y 39.1541  U 42.0347  V 45.1828     73536 bit
AU     8: P    T0 L0 Q0   QP 26   Y 38.7813  U 42.0895  V 45.0952     39784 bit
AU     4: B    T1 L0 Q0   QP 29   Y 37.6244  U 41.8723  V 44.7600     13472 bit
AU     2: B    T2 L0 Q0   QP 31   Y 37.4366  U 41.8395  V 44.6682      7392 bit
AU     1: B    T3 L0 Q0   QP 32   Y 37.4572  U 41.9035  V 44.9427      3048 bit
AU     3: B    T3 L0 Q0   QP 32   Y 37.0191  U 41.7916  V 44.5182      3608 bit
AU     6: B    T2 L0 Q0   QP 31   Y 37.2729  U 41.8385  V 44.8710      7184 bit
AU     5: B    T3 L0 Q0   QP 32   Y 37.0346  U 41.7872  V 44.7472      3832 bit
AU     7: B    T3 L0 Q0   QP 32   Y 37.1397  U 41.8597  V 44.8633      3640 bit
AU    16: P    T0 L0 Q0   QP 26   Y 38.4945  U 41.8488  V 45.1749     46536 bit
AU    12: B    T1 L0 Q0   QP 29   Y 37.4345  U 41.7527  V 44.5879     14672 bit
AU    10: B    T2 L0 Q0   QP 31   Y 37.0726  U 41.6730  V 44.6142      7632 bit
AU     9: B    T3 L0 Q0   QP 32   Y 37.0313  U 41.8043  V 44.7028      3696 bit
AU    11: B    T3 L0 Q0   QP 32   Y 36.6776  U 41.5409  V 44.4636      4008 bit
AU    14: B    T2 L0 Q0   QP 31   Y 36.9673  U 41.3851  V 44.3662      8344 bit
AU    13: B    T3 L0 Q0   QP 32   Y 36.4679  U 41.4806  V 44.2933      5304 bit
AU    15: B    T3 L0 Q0   QP 32   Y 36.9851  U 41.5121  V 44.8542      3896 bit
AU    24: P    T0 L0 Q0   QP 26   Y 38.5363  U 41.9500  V 45.0240     40272 bit
AU    20: B    T1 L0 Q0   QP 29   Y 37.2133  U 41.7125  V 45.0120     10256 bit
AU    18: B    T2 L0 Q0   QP 31   Y 37.1555  U 41.6319  V 44.9712      4904 bit
AU    17: B    T3 L0 Q0   QP 32   Y 36.8352  U 41.5980  V 44.8552      3192 bit
AU    19: B    T3 L0 Q0   QP 32   Y 36.6355  U 41.4704  V 44.6452      2744 bit
AU    22: B    T2 L0 Q0   QP 31   Y 37.3830  U 41.7196  V 44.9114      5464 bit
```

Figure A.4: List of the Encoded frames for temporal scalability.

A reference decoded YUV file of the encoded video sample has been created in order to assess the video quality (e.g., PSNR) of the transmitted video over the network by comparing it to the reference encoded YUV file. The following command is used for creating the reference decoded YUV file and the reference PSNR evaluation result respectively:

```
> H264AVCDecoderLibTestStatic temporal.264 temporal_ref.yuv

> psnr 352 288 420 foreman_cif.yuv temporal_ref.yuv > temporal_psnr_ref.txt
```

The output files of this step for every video streaming session are shown in Table A.1.

Table A.1: Video encoding output files.

foreman_cif.yuv	raw source video file
temporal.264	temporal scalablitiy encoded video sample
temporal.mp4	encoded, encapsulated and hinted video file
temporal_ref.yuv	reference YUV file before decoding
temporal_psnr_ref.txt	reference PSNR evaluation result

A.0.2.2 Video Transmission/Publishing

In this step, the encoded video sample is transmitted by using the proposed scalable RTPS/UDP based video streaming implementation and the RTP/UDP based mp4trace implementation of EvalSVC[7] in two different video streaming sessions. In the scalable RTPS-based video streaming session, the publisher side has published four different temporal sub-streams using the DDS[5] partition QoS. Only one of those partitions (sub-streams) are supposed to be delivered to the subscriber side. The proposed architecture is able to switch among those partitions adaptively in accordance with the network bandwidth limitation and network congestion. In the publisher side, the following terminal command has been used to publish the encoded, encapsulated and hinted video sample in each unicast or multicast session:

```
> ./DDS_VideoStream_publisher
```

Four different video publishing sessions have been performed. In every session, the number of subscribers have been increased in order to study the scalabality of the proposed scalable video streaming architecture. Meanwhile, a static background traffic of 2mbps has been sent in every session using the follwoing command:

```
> iperf -c <ip> -u -p <port> -b 2m -t 60s -i 1 -f m
```

The network Tcpdump monitoring tool has been also used to trace the publisher wireless outgoing IP traffic of every video streaming session using the follwoing command:

```
> sudo tcpdump -i wlan0 -w alls.cap
```

This tcpdump command records all the outgoing traffic regardless of how many receivers/subscribers in the receiver node. Therefore, the outgoing traffic of every receiver/subscriber by their traffic port number needs to be extracted as follows:

```
> tcpdump -r alls.cap -n -tt -v udp port <port> > temporal_sd
```

In the RTP-based video streaming session, the sender side has sent the encoded, encapsulated, and hinted video sample by the mp4trace tool of EvalSVC[7] using the following command:

```
> mp4trace -f -s <ip> <port> temporal.mp4 > temporal_st
```

For the unicast scenario of 1 sender to 1 receiver, the receiver specific IP address has been used while in the multicast sencarios of 1 sender to 6, 12, and 18 receiver, the multicast IP address (224.0.0.1) has been used. Moreover, The same background and traffic monitoring command mentioned befored are used in this RTP-based video streaming sessions.

The output files of this step for every video streaming session are shown in Table A.2.

Table A.2: Video transmission/publishing output files.

temporal_st	sender trace file (frame types, packet segmentation, ..)
alls.cap	sender tcpdump trace file
temporal_sd	sender tcpdump filtered trace file

A.0.2.3 Video Receiving/Subscribing

In the scalable RTPS-based video streaming session, the number of subscribers have been increased in order to study the scalablity of the proposed scalable architecture in a time-varying and limited wireless network resources (e.g., time-varying bandwidth and the traffic congestion). The test began with a unicast scenario of 1 publisher to 1 subscriber and increase the number of subscribers to be 1 publisher to 6, 12 and 18 subscribers respectively in multicast scenarios. The following command has been used to subscribe to the published video stream:

```
> ./DDS_VideoStream_subscriber
```

The network Tcpdump monitoring tool have been also used to trace the subscriber wireless incoming IP traffic of every video streaming session using the follwoing command:

```
> sudo tcpdump -i wlan0 -w allr.cap
```

This tcpdump command records all the incoming traffic regardless of how many receivers/subscribers in the same node. Therefore, the incoming traffic of every receiver/subscriber by their traffic port number needs to be extracted as follows:

```
> tcpdump -r allr.cap -n -tt -v udp port <port> > temporal_rd
```

In the RTP-based video streaming session, the transmitted video stream has been received by listining to the target UDP port using the following command:

```
> nc -l -u <port>
```

Moreover, the same traffic monitoring commands have been used in every RTP-based video streaming session.

The output files of this step for ever video streaming session is shown in Table A.3.

Table A.3: Video receiving/subscribing output files.

allr.cap	receiver tcpdump trace file
temporal_rd	receiver tcpdump filtered trace file

A.0.2.4 Video Reconstruction and Evaluation

In this step, the video reconstruction tool (etmp4) of EvalSVC[7] is used to reconstruct the transmitted video as it is seen by the receiver. The video and trace files which are generated by the previous steps are processed by etmp4 (Evaluation Traces of MP4-file transmission) as follows:

```
> etmp4 -f -x temporal_sd temporal_rd temporal\_st temporal.mp4 rtemporal
```

This generates a (possibly corrupted) MP4 video file, where all frames that got lost or were corrupted are removed from the original video track. Two files are saved, a MP4-file containing the damaged video track (rtemporal.mp4), and raw video file containing only the undamaged frames (rtemporal.264). These files are decoded by H264/SVC decoder as JSVM[83] to produce the YUV file as seen at the receiver using the following command:

```
> H264AVCDecoderLibTestStatic rtemporal.264 rtemporal.yuv
```

The resulting YUV file is used to calculate the PSNR of the received video using the following command:

```
> psnr 352 288 420 foreman_cif.yuv rtemporal.yuv > psnr_rtemporal.txt
```

Etmp4 also generates some more files as shown in Table A.4. Those output files include some important performance related results such as video end-to-end delay, sent and received bit rate, and frame loss ratio.

Table A.4: Video reconstruction and evaluation output files.

rtemporal.mp4	received MP4 video file
rtemporal.264	received raw video file
loss_rtemporal.txt	contains I, P, B and overall frame loss in %
delay_rtemporal.txt	containes end-to-end delay and jitter in seconds
rate_s_rtemporal.txt	contains the measured bit rate at the sender in bytes per second
rate_r_rtemporal.txt	contains the measured bit rate at the receiver in bytes per second
psnr_rtemporal.txt	contains the PSNR of the received video

APPENDIX B

CONFIGURED USER DATA QUALITY OF SERVICE

B.0.3 Publisher QoS

```
<?xml version="1.0"?>

<!-- Description

XML QoS Profile for DDS_VideoStream

The QoS configuration of the DDS entities in the generated example is

loaded from this file.

This file is used only when it is in the current working directory

or when the enviroment variable

NDDS_QOS_PROFILES is defined and points to this file.

For more information about XML QoS Profiles see Chapter 15 in the

RTI Connext user manual.-->
```

```
<dds xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="/home/pc6/RTI/ndds.5.0.0/scripts/..
/resource/rtiddsgen/../qos_profiles_5.0.0/schema/rti_dds_qos_profiles.xsd"
     version="5.0.0">
    <!-- QoS Library containing the QoS profile used in the generated example.
A QoS library is a named set of QoS profiles.
    -->
    <qos_library name="DDS_VideoStream_Library">
        <!-- QoS profile used to configure reliable communication between the DataW
  and DataReader created in the example code.
  A QoS profile groups a set of related QoS.
        -->
         <qos_profile name="Reliable">
            <datareader_qos>
                <reliability>
                    <kind>BEST_EFFORT_RELIABILITY_QOS</kind>
                </reliability>
                <history>
                    <kind>KEEP_LAST_HISTORY_QOS</kind>
                    <depth>30</depth>
                </history>
                <protocol>
                    <rtps_reliable_reader>
                        <min_heartbeat_response_delay>
                            <sec>0</sec>
                            <nanosec>0</nanosec>
                        </min_heartbeat_response_delay>
                        <max_heartbeat_response_delay>
                            <sec>0</sec>
                            <nanosec>0</nanosec>
                        </max_heartbeat_response_delay>
                    </rtps_reliable_reader>
                </protocol>
            </datareader_qos>
            <datawriter_qos>
                <reliability>
                    <kind>BEST_EFFORT_RELIABILITY_QOS</kind>
                    <max_blocking_time>
                        <sec>5</sec>
                        <nanosec>0</nanosec>
                    </max_blocking_time>
                </reliability>
                <history>
                    <kind>KEEP_LAST_HISTORY_QOS</kind>
            <depth>30</depth>
</history>
<resource_limits>
    <max_samples>30</max_samples>
</resource_limits>
<protocol>
    <rtps_reliable_writer>
        <low_watermark>5</low_watermark>
        <high_watermark>15</high_watermark>
        <heartbeat_period>
            <sec>0</sec>
            <nanosec>100000000</nanosec>
        </heartbeat_period>
        <fast_heartbeat_period>
            <sec>0</sec>
            <nanosec>10000000</nanosec>
        </fast_heartbeat_period>
        <late_joiner_heartbeat_period>
            <sec>0</sec>
            <nanosec>10000000</nanosec>
        </late_joiner_heartbeat_period>
        <max_heartbeat_retries>500</max_heartbeat_retries>
```

```
                    <min_nack_response_delay>
                        <sec>0</sec>
                        <nanosec>0</nanosec>
                    </min_nack_response_delay>
                    <max_nack_response_delay>
                        <sec>0</sec>
                        <nanosec>0</nanosec>
                    </max_nack_response_delay>
                    <min_send_window_size>32</min_send_window_size>
                    <max_send_window_size>32</max_send_window_size>
                </rtps_reliable_writer>
            </protocol>
        </datawriter_qos>
    </qos_profile>
    <qos_profile name="DDS_VideoStream_Profile" base_name="Reliable" is_default
        <!-- QoS used to configure the data writer created in the example code
        <datareader_qos>
            <resource_limits>
                <max_samples>100</max_samples>
                <initial_samples>100</initial_samples>
            </resource_limits>
            <protocol>
        <rtps_reliable_reader>

            <heartbeat_suppression_duration>

                <sec>0</sec>

                <nanosec>0</nanosec>

            </heartbeat_suppression_duration>

        </rtps_reliable_reader>

    </protocol>

</datareader_qos>

<datawriter_qos>

    <resource_limits>

        <max_samples>LENGTH_UNLIMITED</max_samples>

        <initial_samples>100</initial_samples>

    </resource_limits>

    <protocol>

        <rtps_reliable_writer>

            <low_watermark>10</low_watermark>

            <high_watermark>100</high_watermark>

            <heartbeats_per_max_samples>

                1000

            </heartbeats_per_max_samples>

            <!--

            Speed up the heartbeat rate. See reliable.xml for
```

```
more information about this parameter.
                    -->
                    <heartbeat_period>
                        <!-- 10 milliseconds: -->
                        <sec>0</sec>
                        <nanosec>10000000</nanosec>
                    </heartbeat_period>
                    <!--
                    Speed up the heartbeat rate. See reliable.xml for
                    more information about this parameter.
                    -->
                    <fast_heartbeat_period>
                        <!-- 1 millisecond: -->
                        <sec>0</sec>
                        <nanosec>1000000</nanosec>
                    </fast_heartbeat_period>
                </rtps_reliable_writer>
            </protocol>
        </datawriter_qos>
        <participant_qos>
            <!--
            The participant name, if it is set, will be displayed in the
```

```
RTI Analyzer tool, making it easier for you to tell one

application from another when you're debugging.

              -->

              <participant_name>

                  <name>RTI Example (Low Latency)</name>

              </participant_name>

              <discovery>

                     <initial_peers>

                     <element>239.255.0.1</element>

                     <element>builtin.shmem://</element>

                     <element>builtin.udpv4://127.0.0.1</element>

                     <element>builtin.udpv4://192.168.1.101</element>

                     <element>builtin.udpv4://192.168.1.102</element>

                     <element>builtin.udpv4://192.168.1.103</element>

                     <element>builtin.udpv4://192.168.1.104</element>

                     </initial_peers>

                     <multicast_receive_addresses>

                     <element>293.255.0.1</element>

                     </multicast_receive_addresses>

              </discovery>

          </participant_qos>

      </qos_profile>
```

```
    </qos_library>

</dds>
```

B.0.4 Subscriber QoS

```
<?xml version="1.0"?>

<!--

Description

XML QoS Profile for DDS_VideoStream

The QoS configuration of the DDS entities in the generated example is

loaded from this file.

This file is used only when it is in the current working directory

or when the enviroment variable

NDDS_QOS_PROFILES is defined and points to this file.

For more information about XML QoS Profiles see Chapter 15 in the

RTI Connext user manual.

-->

<dds xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

     xsi:noNamespaceSchemaLocation="/home/pc6/RTI/ndds.5.0.0/scripts/..

/resource/rtiddsgen/../qos_profiles_5.0.0/schema/rti_dds_qos_profiles.xsd"

     version="5.0.0">

    <!-- QoS Library containing the QoS profile used in the generated example.

A QoS library is a named set of QoS profiles.
```

```
    -->

    <qos_library name="DDS_VideoStream_Library">


        <!-- QoS profile used to configure reliable communication between the DataW
and DataReader created in the example code.
A QoS profile groups a set of related QoS.
        -->
         <qos_profile name="Reliable">
            <datareader_qos>
                <reliability>
                    <kind>BEST_EFFORT_RELIABILITY_QOS</kind>
                </reliability>
                <history>
                    <kind>KEEP_LAST_HISTORY_QOS</kind>
                    <depth>30</depth>
                </history>
                <protocol>
                    <rtps_reliable_reader>
                        <min_heartbeat_response_delay>
                            <sec>0</sec>
                            <nanosec>0</nanosec>
                        </min_heartbeat_response_delay>
                        <max_heartbeat_response_delay>
                            <sec>0</sec>
                            <nanosec>0</nanosec>
                        </max_heartbeat_response_delay>
                    </rtps_reliable_reader>
                </protocol>
            </datareader_qos>
            <datawriter_qos>
                <reliability>
                    <kind>BEST_EFFORT_RELIABILITY_QOS</kind>
                    <max_blocking_time>
                        <sec>5</sec>
                        <nanosec>0</nanosec>
                    </max_blocking_time>
                </reliability>
                <history>
                    <kind>KEEP_LAST_HISTORY_QOS</kind>
                    <depth>30</depth>
                </history>
                <resource_limits>
                    <max_samples>30</max_samples>
                </resource_limits>
```

```
<protocol>
    <rtps_reliable_writer>
        <low_watermark>5</low_watermark>
        <high_watermark>15</high_watermark>
        <heartbeat_period>
            <sec>0</sec>
            <nanosec>100000000</nanosec>
        </heartbeat_period>
        <fast_heartbeat_period>
            <sec>0</sec>
            <nanosec>10000000</nanosec>
        </fast_heartbeat_period>
        <late_joiner_heartbeat_period>
            <sec>0</sec>
            <nanosec>10000000</nanosec>
        </late_joiner_heartbeat_period>
        <max_heartbeat_retries>500</max_heartbeat_retries>
        <min_nack_response_delay>
            <sec>0</sec>
            <nanosec>0</nanosec>
        </min_nack_response_delay>
        <max_nack_response_delay>
```

```
                    <sec>0</sec>
                    <nanosec>0</nanosec>
                </max_nack_response_delay>
                <min_send_window_size>32</min_send_window_size>
                <max_send_window_size>32</max_send_window_size>
            </rtps_reliable_writer>
        </protocol>
    </datawriter_qos>
</qos_profile>
<qos_profile name="DDS_VideoStream_Profile" base_name="Reliable" is_default
    <!-- QoS used to configure the data writer created in the example code
    <datareader_qos>
        <resource_limits>
            <max_samples>100</max_samples>
            <initial_samples>100</initial_samples>
        </resource_limits>
        <protocol>
            <rtps_reliable_reader>
                <heartbeat_suppression_duration>
                    <sec>0</sec>
                    <nanosec>0</nanosec>
                </heartbeat_suppression_duration>
```

```
                    </rtps_reliable_reader>

                </protocol>

            </datareader_qos>

            <datawriter_qos>

                <resource_limits>

                    <max_samples>LENGTH_UNLIMITED</max_samples>

                    <initial_samples>100</initial_samples>

                </resource_limits>

                <protocol>

                    <rtps_reliable_writer>

                        <low_watermark>10</low_watermark>

                        <high_watermark>100</high_watermark>


                        <heartbeats_per_max_samples>

                            1000

                        </heartbeats_per_max_samples>

                        <!--

Speed up the heartbeat rate. See reliable.xml for

more information about this parameter.

                        -->

                        <heartbeat_period>

                            <!-- 10 milliseconds: -->
                            <sec>0</sec>
                            <nanosec>10000000</nanosec>
                        </heartbeat_period>
                        <!--
Speed up the heartbeat rate. See reliable.xml for
more information about this parameter.
                        -->
                        <fast_heartbeat_period>
                            <!-- 1 millisecond: -->
                            <sec>0</sec>
                            <nanosec>1000000</nanosec>
                        </fast_heartbeat_period>
                    </rtps_reliable_writer>
                </protocol>
            </datawriter_qos>
            <participant_qos>
                <!--
The participant name, if it is set, will be displayed in the RTI Analyzer tool, ma
                -->
                <participant_name>
                    <name>RTI Example (Low Latency)</name>
                </participant_name>
```

```
                <discovery>

                    <initial_peers>

                    <element>239.255.0.1</element>

                    <element>builtin.shmem://</element>

                    <element>builtin.udpv4://127.0.0.1</element>

                    <element>builtin.udpv4://192.168.1.101</element>

                    <element>builtin.udpv4://192.168.1.102</element>

                    <element>builtin.udpv4://192.168.1.103</element>

                    <element>builtin.udpv4://192.168.1.104</element>

                    </initial_peers>

                    <multicast_receive_addresses>

                    <element>293.255.0.1</element>

                    </multicast_receive_addresses>

                </discovery>


            </participant_qos>

        </qos_profile>

    </qos_library>

</dds>
```

REFERENCES

[1] C.-H. Ke, C.-K. Shieh, W.-S. Hwang, and A. Ziviani, "An Evaluation Framework for More Realistic Simulations of MPEG Video Transmission," *J. Inf. Sci. Eng.*, pp. 425–440, 2008.

[2] H. Sun, A. Vetro, J. Xin, H. Sun, A. Vetro, and J. Xin, "An overview of scalable video streaming," *Wireless Communications and Mobile Computing*, vol. 7, pp. 159–172, 2007.

[3] "Fraunhofer Heinrich Hertz Institute." [Online]. Available: http://www.hhi.fraunhofer.de/

[4] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 17, no. 9, pp. 1103–1120, 2007.

[5] OMG, "Data Distribution Service for Real-Time systems," *Object Management Group, 1.2 formal/07-01-01 edition*, 2007.

[6] R. Connext, "Core Library and Utilities User's Manual," *Real-Time Innovations, Inc.*, 2012.

[7] T. A. Le, H. Nguyen, and H. Zhang, "EvalSVC x2014; An evaluation platform for scalable video coding transmission," pp. 1–6, 2010.

[8] C. Lee, K. Song, Y. Joo, and Y. Kim, "Adaptive rate control for real-time video streaming over the mobile WiMAX," pp. 1454–1457, 2008.

[9] N. Cranley and M. Davis, "Study of the Behaviour of Video Streaming over IEEE 802.11b WLAN Networks," pp. 349–355, 2006.

[10] Y. Wang and Q.-F. Zhu, "Error control and concealment for video communication: a review," *Proceedings of the IEEE*, vol. 86, no. 5, pp. 974–997, 1998.

[11] Y. Wang, S. Wenger, J. Wen, and K. K. Aggelos, "Error Resilient Video Coding Techniques," no. July, 2000.

[12] T.Wiegand, G. Sullivan, J. Reichel, H. Schwarz, and M. W. (Editors), "Joint draft 9 of SVC amendment (revision 2)," *Document JVT- V201, Marrakech, Morocco, January 13-19, 2007*, 2007.

[13] H.-C. Huang, W.-H. Peng, T. Chiang, and H.-M. Hang, "Advances in the scalable amendment of H.264/AVC," *Communications Magazine, IEEE*, vol. 45, no. 1, pp. 68–76, 2007.

[14] J. G. Apostolopoulos, W. tian Tan, and S. J. Wee, "Video Streaming: Concepts, Algorithms, and Systems," 2002.

[15] I.-T. R. H.261., "Video codec for audiovisual services at px64 kbits/s," *Telecommunication Union, version 1, 1996; version 2, 1997.*, 1997.

[16] I. ITU-T Rec. H.263, "Video coding for low bit rate communication," *Inter. Telecommunication Union, 1993.*, 1993.

[17] "Advanced Video Coding for Generic Audiovisual Services," *Version 3, ITU Rec. H264/ISO IEC 14996-10 AVC*, 2005.

[18] I. 14496., "Coding of audio-visual objects." *International Organization for Standardization (ISO), 1999*, 1999.

[19] I. 802.11a 1999., "High-speed Physical Layer in the 5 GHz band," 1999.

[20] I. 802.11g 2003., "Further Higher Data Rate Extension in the 2.4 GHz Band," 2003.

[21] I. 802.11b 1999., "Higher Speed Physical Layer Extension in the 2.4 GHz band," 1999.

[22] I. 802.11n 2009Amendment 5., "Enhancements for Higher Throughput," *IEEE-SA. 29 October 2009.*, 2009.

[23] I. S. 802.11e Amendment., "Medium Access Control (MAC) Enhancements for Quality of Service (Qos)," 2005.

[24] T.Wiegand, "Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec. H.264ISO/IEC 14496-10 AVC),"

in Joint Video Team (JVT) of ISO/ICE MPEG and ITU-T VCEG, VT-G050, Pattaya, Thailand, 2003.

[25] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H.264/AVC video coding standard,” *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 560–576, 2003.

[26] Y. Fallah, H. Mansour, S. Khan, P. Nasiopoulos, and H. Alnuweiri, “A link adaptation scheme for efficient transmission of H. 264 scalable video over multirate WLANs,” *. . . Systems for Video . . .*, vol. 18, no. 7, pp. 875–887, 2008.

[27] S. . RFC 3550, “RTP: A Transport Protocol for Real-Time Applications.”

[28] S. . RFC 3551, “RTP Profile for Audio and Video Conferences with Minimal Control.”

[29] O. RFC 3984, “RTP Payload Format for H.264 Video.”

[30] R. Joshi, “Building effective Real-Time distributed publish-subscribe framework part 1,” *Real-Time Innovations.,2006*, 2006.

[31] S. Oh, J. hoon Kim, and G. Fox, “Real-Time Performance Analysis for Publish/Subscribe Systems,” 2009.

[32] M. AnisMastouri and S. Hasnaoui, “Performance of a Publish/Subscribe Middleware for the Real-Time Distributed Control systems Summary,” 2007.

[33] O. Karimi and M. Fathy, "Adaptive end-to-end QoS for multimedia over heterogeneous wireless networks," *Computers & electrical engineering*, vol. i, pp. 160–167, 2010.

[34] C.-M. Chen, C.-W. Lin, H.-C. Wei, and Y.-C. Chen, "Robust video streaming over wireless LANs using multiple description transcoding and prioritized retransmission," *Journal of Visual Communication and Image Representation*, vol. 18, no. 3, pp. 191–206, Jun. 2007.

[35] Z. Lei and N. D. Georganas, "Adaptive video transcoding and streaming over wireless channels," *Journal of Systems and Software*, vol. 75, no. 3, pp. 253–270, Mar. 2005.

[36] X. Zhang and M. Huang, "Error Resilient Transcoding for Wireless Video Transmission," *2009 International Conference on Wireless Networks and Information Systems*, pp. 286–289, Dec. 2009.

[37] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication - SIGCOMM '00*, pp. 43–56, 2000.

[38] X. Tong, W. Gao, and Q. Huang, "A Novel Rate Control Scheme for Video Streaming over Wireless Networks," *Third International Conference on Image and Graphics (ICIG'04)*, pp. 369–372, 2004.

[39] M. Chen and A. Zakhor, "Rate Control for Streaming Video over Wireless," vol. 00, no. C, 2004.

[40] G. a. Al-Suhail, N. Wakamiya, and R. S. Fyath, "Error-Resilience of TCP-Friendly Video Transmission over Wireless Channel," *2006 9th International Conference on Control, Automation, Robotics and Vision*, pp. 1–6, 2006.

[41] J.-Y. Pyun and H.-J. Choi, "TCP-Friendly Congestion Control for Streaming Video Service over Wireless Overlay Network," *ICC Workshops - 2008 IEEE International Conference on Communications Workshops*, pp. 47–51, May 2008.

[42] H. Luo, D. Wu, S. Ci, A. Argyriou, and H. Wang, "Quality-Driven TCP Friendly Rate Control for Real-Time Video Streaming," *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference*, pp. 1–5, 2008.

[43] H. Luo, D. Wu, S. Ci, H. Sharif, and H. Tang, "TFRC-Based Rate Control for Real-Time Video Streaming over Wireless Multi-Hop Mesh Networks," *2009 IEEE International Conference on Communications*, pp. 1–5, Jun. 2009.

[44] H. Luo, S. Ci, D. Wu, and H. Tang, "End-to-end optimized TCP-friendly rate control for real-time video streaming over wireless multi-hop networks," *Journal of Visual Communication and Image Representation*, vol. 21, no. 2, pp. 98–106, Feb. 2010.

[45] A. Markopoulou, E. Setton, M. Kalman, and J. Apostolopoulos, "WiSE video: using in-band wireless loss notification to improve rate-controlled video streaming," vol. 1, pp. 249–252 Vol.1, 2004.

[46] M. Chen and A. Zakhor, "Rate control for streaming video over wireless," *Wireless Communications, IEEE*, vol. 12, no. 4, pp. 32–41, 2005.

[47] Y.-S. Hong, H.-S. Lim, and J.-S. Hong, "A Cost Effective Rate Control for Streaming Video Dedicated to Wireless Handheld Devices," pp. 537–542, 2008.

[48] Y. Huang, S. Mao, and S. Midkiff, "A Control-Theoretic Approach to Rate Control for Streaming Videos," *Multimedia, IEEE Transactions on*, vol. 11, no. 6, pp. 1072–1081, 2009.

[49] G. Maione and D. Striccoli, "Transmission control of Variable-Bit-Rate video streaming in UMTS networks," *Control Engineering Practice*, vol. 20, no. 12, pp. 1366–1373, Dec. 2012.

[50] L. Atzori, G. Ginesu, A. Floris, and D. Giusto, "Rate control based on reduced-reference image quality estimation for streaming video over wireless channels," pp. 2021–2025, 2012.

[51] Y. Xiaogang, L. Jiqiang, and L. Ning, "Congestion Control Based on Priority Drop for H.264/SVC," *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, pp. 585–589, 2007.

[52] T. Schierl, H. Schwarz, D. Marpe, and T. Wiegand, "Wireless Broadcasting Using the Scalable Extension of H. 264/AVC," *2005 IEEE International Conference on Multimedia and Expo*, pp. 884–887, 2005.

[53] T. Schierl, C. Hellge, S. Mirta, K. Grüneberg, and T. Wiegand, "Using H . 264 / AVC-based Scalable Video Coding (SVC) for Real Time Streaming in Wireless IP Networks," pp. 3455–3458, 2007.

[54] G. Ji and B. Liang, "Stochastic Rate Control for Scalable VBR Video Streaming over Wireless Networks," *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*, pp. 1–6, Nov. 2009.

[55] H.-L. Chen, P.-C. Lee, and S.-H. Hu, "Improving Scalable Video Transmission over IEEE 802.11e through a Cross-Layer Architecture," *2008 The Fourth International Conference on Wireless and Mobile Communications*, pp. 241–246, 2008.

[56] H. Mansour, Y. Fallah, P. Nasiopoulos, and V. Krishnamurthy, "Dynamic resource allocation for MGS H. 264/AVC video transmission over link-adaptive networks," *Multimedia, IEEE . . .* , vol. 11, no. 8, pp. 1478–1491, 2009.

[57] H. Zhang, Y. Zheng, M. Khojastepour, and S. Rangarajan, "Cross-layer optimization for streaming scalable video over fading wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 3, pp. 344–353, Apr. 2010.

[58] M. Li, Z. Chen, and Y.-P. Tan, "Cross-layer optimization for SVC video delivery over the IEEE 802.11e wireless networks," *Journal of Visual Communication and Image Representation*, vol. 22, no. 3, pp. 284–296, Apr. 2011.

[59] D. Song and C. W. Chen, "Maximum-throughput delivery of SVC-based video over MIMO systems with time-varying channel capacity," *Journal of Visual Communication and Image Representation*, vol. 19, no. 8, pp. 520–528, Dec. 2008.

[60] D. Radakovic and R. Ansari, "Priority-aware transfer of SVC encoded video over MIMO communications system," *. . . , 2009. PCS 2009*, 2009.

[61] C.-h. Wang, R.-i. Chang, J.-m. Ho, and S.-c. Hsu, "Rate-sensitive ARQ for real-time video streaming," *GLOBECOM '03. IEEE Global Telecommunications Conference (IEEE Cat. No.03CH37489)*, pp. 3361–3365, 2003.

[62] H.-B. Yu, S. Yu, and C. Wang, "A highly efficient, low delay architecture for transporting H.264 video over wireless channel," *Signal Processing: Image Communication*, vol. 19, no. 4, pp. 369–385, Apr. 2004.

[63] Y. Wang, L.-P. Chau, and K.-H. Yap, "Bit-Rate Allocation for Broadcasting of Scalable Video Over Wireless Networks," *IEEE Transactions on Broadcasting*, vol. 56, no. 3, pp. 288–295, Sep. 2010.

[64] A. Nasipuri and S. Das, "Multichannel CSMA with signal power-based channel selection for multihop wireless networks," vol. 1, pp. 211–218 vol.1, 2000.

[65] T.-T. Luong, B.-S. Lee, and C. Yeo, “Channel Allocation for Multiple Channels Multiple Interfaces Communication in Wireless Ad Hoc Networks,” vol. 4982, pp. 87–98, 2008.

[66] H. S. Chiu, K. Yeung, and K.-S. Lui, “WSN15-2: J-CAR: an Efficient Channel Assignment and Routing Protocol for Multi-channel Multi-interface Mobile Ad Hoc Networks,” pp. 1–5, 2006.

[67] H. Cheng, G. Chen, N. Xiong, and X. Zhuang, “Static channel assignment algorithm in multi-channel wireless mesh networks,” pp. 49–55, 2009.

[68] P. Kyasanur, “Routing and Link-layer Protocols for Multi-Channel Multi-Interface Ad hoc Wireless Networks,” *Sigmobile Mobile Computing and Communications Review*, vol. 10, pp. 31–43, 2006.

[69] H. Skalli, S. Ghosh, S. Das, L. Lenzini, and M. Conti, *Channel Assignment Strategies for Multiradio Wireless Mesh Networks: Issues and Solutions*, 2007, vol. 45, no. 11.

[70] *A hybrid channel assignment strategy to QoS support of video-streaming over multi-channel ad hoc networks*, 2012, vol. 85, no. 2.

[71] J. Liu, F. Li, F. Dou, X. He, Z. Luo, and H. Xiong, “An Adaptive Cross-Layer Mechanism of Multi-channel Multi-interface Wireless Networks for Real-Time Video Streaming,” pp. 165–170, 2010.

[72] S. Lee and K. Chung, *Combining the rate adaptation and quality adaptation schemes for wireless video streaming*, 2008, vol. 19, no. 8.

[73] W. Saesue, C. T. Chou, and J. Zhang, "CROSS-layer QoS-optimized EDCA adaptation for wireless video streaming," pp. 2925–2928, 2010.

[74] M. Loiacono, J. Johnson, J. Rosca, and W. Trappe, "Cross-Layer Link Adaptation for Wireless Video," pp. 1–6, 2010.

[75] D. Kumar, R. Sudarson, and R. Sivasankari, "A QoS aware cross-layer optimisation for wireless video streaming," pp. 1–6, 2011.

[76] J. Clavijo, M. Segarra, C. Baeza, C. Moreno, R. Sanz, A. Jimnez, R. Vzquez, F. Daz, and A. Dez, "Real-Time Video for Distributed Control Systems," *Engineering Practice*, vol. Vol. 9. No. . pp.459-466., 2001.

[77] V. Rostami, S. Ebrahimijam, and O. Sojodishijani, "Real-time distributed control system for navigating omnidirectional soccer robot," pp. 1–4, 2007.

[78] D. Kaff, C. Rodrigues, Y. Krishnamurthy, I. Pyarali, and D. Schmidt, "Application of the QuO quality-of-service framework to a distributed video application," pp. 299–308, 2001.

[79] Z. C., "Distributed Programming with ICE," *Zero Company*, vol. version 3.3.1., 2009.

[80] M. Garcia-Valls, P. Basanta-Val, and I. Estevez-Ayres, "Adaptive real-time video transmission over DDS," pp. 130–135, 2010.

[81] A. Detti, P. Loreti, N. Blefari-Melazzi, and F. Fedi, "Streaming H.264 scalable video over data distribution service in a wireless environment," pp. 1–3, 2010.

[82] "x264." [Online]. Available: http://www.videolan.org/developers/x264.html

[83] "Joint Scalable Video Model JSVM-6," *Geneva, Switzerland: Joint Video Team, Doc. JVT-S202*, 2006.

[84] "opencv." [Online]. Available: http://opencv.org/

[85] J. Le Feuvre, C. Concolato, and J.-C. Moissinac, "GPAC: open source multimedia framework," pp. 1009–1012, 2007.

[86] "ffmpeg." [Online]. Available: http://www.videolan.org/developers/x264.html

[87] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RFC 3550: RTP: A Transport Protocol for Real-Time Applications," 2003.

Vitae

- Name: Mohammed Abduljalil Mohammed Al-saeedi
- Nationality: Yemen
- Date of Birth: Feb. 5 1983
- Email: *modstarr@gmail.com*
- Permenant Address: King Fahd University of Petroleum and Minerals Dhahran, 31261, Saudi Arabia