Department of Electrical and Computer Engineering

University of Thessaly, Volos, Greece

Master Thesis

Cooperative SVC coded Video Transmission over Wireless Network using Network Coding

Vasileios I. Pashos

Supervisor:

Assistant Professor Dr. Antonios Argyriou, University of Thessaly

Advisors:

Assistant Professor Dr. Athanasios Korakis, University of Thessaly

Lecturer Dr. Nikolaos Thomos, University of Essex

Volos, January 2016

Περίληψη

Η κωδικοποίηση βίντεο (SVC) είναι μια νέα αρκετά αποδοτική μέθοδος κωδικοποίησης που αντιμετωπίζει με αποτελεσματικό τρόπο τις δυσκολίες που προκύπτουν απο τα χαρακτηριστικά των σύγχρονων συστημάτων μετάδοσης βίντεο. Η συγκεκριμένη κωδικοποίηση μας παρέχει η δυνατότητα να παραλείπουμε συγκεκριμένα κομμάτια του προς μετάδοση βίντεο bitstream προκειμένου αυτό να μπορεί να μεταδοθεί αποτελεσματικά λαμβάνοντας υπόψιν κάθε φορά τις διαφορετικές απαιτήσεις των τελικών χρηστών, τις συνεχώς μεταβαλλόμενες συνθήκες που υπάρχουν στα δίκτυα μετάδοσης δεδομένων, όπως επίσης και τους διαφορετικούς τύπους τερματικών συσκευών. Η SVC κωδικοποίηση είναι μία επέκταση του H.264/MGEG4-AVC προτύπου κωδικοποίησης βίντεο. Στόχος του νέου αυτού προτύπου (SVC) είναι η αποδοτικότερη κωδικοποίηση βίντεο υψηλης ποιότητας σε ένα bitstream που περιέχει ένα ή περισσότερα υποσύνολα του αρχικού bitstream αρχείου τα οποία μπορούν με τη σειρά τους να αποκωδικοποιηθούν με πολυπλοκότητα και ποιότητα αντίστοιχη με εκείνη που επιτυγχάνεται από το υπάρχον πρότυπο κωδικοποίησης H.264/AVC. Η κωδικοποίηση δικτύου (Network Coding) επιπλέον, είναι μια καινούρια τεχνική, που χρησιμοποιείται για τη μετάδοση πακέτων μέσα απο ένα δίκτυο, βάσει της οποίας μπορούμε να επιτύχουμε ταχύτερη, πιο αποδοτική και ασφαλή μεταδοση δεδομένων. Σε αυτή την περίπτωση κάθε κόμβος του δικτύου, αντί απλά να προωθεί τα πακέτα δεδομένων που λαμβάνει, συνδυάζει ένα σύνολο πακέτων μεταξύ τους προκειμένου να δημιουργήσει νέα κωδικοποιημένα πακέτα προς μετάδοση. Αυτή η τεχνική προσφέρει αποδεδειγμένα έναν πιο αποδοτικό τρόπο προκειμένου να επιτύχουμε τη μέγιστη δυνατή μετάδοση δεδομένων σε κάποιο δίκτυο. Σκοπός της συγκεκριμένης εργασίας είναι η έρευνα της αποδοτικότητας της χρήσης της τεχνικής της κωδικοποίησης δικτύου για μετάδοση βίντεο SVC κωδικοποίησης σε μία ομάδα γειτονικών χρηστών που είναι συνδεδεμένοι στο ίδιο ασύρματο δίκτυο σε τοπολογία πλέγματος.

Acknowledgements

This thesis has been an inspiring, very challenging, but always interesting and exciting experience. I am deeply indebted to my advisor, Professor Antonios Argyriou for his support encouragement and invaluable advice during my master's studies, his understanding and personal guidance has provided an excellent basis for the present thesis.
Besides my advisor I would also like to thank all academic staff of the department of Electrical and Computer engineering of University of Thessaly who help me gain all this knowledge through the period my postgraduate studies.
Last but not least I would like to thank my family, my parents who have given to me continuous and unconditional support and for supporting me spiritually throughout writing this thesis and my life in general.

Abstract

Scalable Video coding (SVC) is a highly attractive solution to the problem posed by the characteristics of modern video transmission systems. The term scalability refers to the removal of parts of the video bit-stream in order to adapt it to the various needs or preferences of end users as well as to varying terminal capabilities or network conditions. SVC is an extension to the H.264/MGEG4-AVC video coding standard. The objective of the SVC standardization is to enable the encoding of a high-quality video bit-stream that contains one or more subset bit-streams that can themselves be decoded with a complexity and reconstruction quality similar to that achieved, using the existing H.264/AVC design with the same quantity of data as in the subset bit-stream. Network Coding is a technique which can be used to improve a network's throughput, efficiency, scalability and security. Using this technique a node instead of simply relaying the packets of information it receives, it takes several packets and combine them together for transmission. This has proven efficient in order to attain the maximum possible information flow in a network. The scope of the present work is to investigate the performance of network coding for a SVC coded video transmitted in a group of users with proximity to each other in a wireless mesh topology.

Table of Contents

1 INTRODUCTION....................1

2 VIDEO ENCODING....................3

2.1 General....................3

2.2 Video Encoding Formats....................3

3 SCALABLE VIDEO CODING (SVC)....................4

3.1 General....................4

3.2 Overview of H.264/SVC....................5

3.2.1 Temporal Scalability....................5

3.2.2 Spatial Scalability....................8

3.2.3 Spatial Scalability....................9

3.3 SVC Architecture....................10

3.3.1 NAL Units....................10

3.3.2 NAL Units in Byte-Stream format....................11

3.3.3 NAL Units in Packet Transport System....................12

3.3.4 VCL and non – VCL NAL Units....................12

3.3.5 Parameter Sets....................12

3.3.6 Access Units....................13

3.3.7 Coded Video Sequences....................14

4 NETWORK CODING....................16

4.1 Introduction....................16

4.1.1 Benefits....................17

4.1.1.1 Throughput....................17

4.1.1.2 Wireless Resources....................19

4.1.1.3 Security....................20

4.1.2 Challenges....................20

4.1.2.1 Complexity....................21

4.1.2.2 Security....................21

4.1.2.3 Integration with Existing Infrastructure....................21

5 FRAMEWORK OF NETWORK CODING....................22

5.1 Linear Coding....................22

5.2 Random Linear Coding....................23

5.2.1 Decoding, Vector Space and Rank....................26

5.3 Efficient Wireless Broadcasting with Network Coding....................27

6 COOPERATIVE MOBILE WIRELESS SYSTEMS....................28

6.1 Overview....................28

6.2 Network Coding in Cooperative Wireless Systems....................29

7 ENVIRONMENT TO SIMULATE....................30

7.1 Environment Overview....................30

7.2 Microcast Architecture....................31

7.3 MicroNC-P2 Algorithm....................32

7.3.1 Algorithm Description....................32

7.4 Reception Rate....................36

8 NETWORK SIMULATOR....................37

8.1 Implementation Details....................37
8.2 JSVM....................38
8.2.1 General Information....................38
8.2.2 Configuration Files....................38
8.2.3 Packet Trace Files....................39
8.3 Simulation Scenario....................40
9 EXPERIMENTS AND RESULTS....................44
9.1 Single Layer coding....................44
9.2 Comparison of QCIF sequences....................46
9.2.1 Experiment 1 (2 Temporal Layers L0,L1)....................46
9.2.2 Experiment 2 (3 Temporal Layers L0,L1,L2)....................48
9.2.3 Experiment 3 (4 Temporal Layers L0,L1,L2,L3)....................50
9.2.4 Experiment 4 (5 Temporal Layers L0,L1,L2,L3,L4)....................51
9.3 Comparison of CIF sequences....................52
9.3.1 Experiment 1 (2 Temporal Layers L0,L1)....................52
9.3.2 Experiment 2 (3 Temporal Layers L0,L1,L2)....................54
9.3.3 Experiment 3 (4 Temporal Layers L0,L1,L2,L3)....................55
9.3.4 Experiment 4 (5 Temporal Layers L0,L1,L2,L3,L4)....................56
10 CONCLUSIONS....................57
11 FUTURE WORK....................59
12 LITERATURE....................60

List of Abbreviations and Symbols

Bit-stream: Contiguous sequence of bits, representing a stream of data, transmitted continuously over a communication path.

SVC: Scalable video coding, an extension of the H.264 (MPEG-4 AVC) video compression standard for video encoding.

Network Coding: Networking technique in which the transmitted data is encoded and decoded to increase network throughput, reduce delays and make the network more robust.

(Random) Linear Network coding: Technique which can be used to improve network's throughput, efficiency and scalability etc. According to this nodes instead of simply relaying the packets of information they receive, they create (random) linear combinations of them and after this they transmit them.

Finite (Galois) Field: Field that contains a finite number of elements. The finite field is a set on which the operations of multiplication, addition, subtraction and division are defined and satisfy certain basic rules.

Cooperative Transmission: Applied when several users that are close to each other exploit the links between them to exchange data that are not in common.

Group Of Pictures (GoP) : A specific number of frames with specific order in which intra and inter-frames are arranged within a coded video stream.

1 Introduction

Today's mobile devices (smartphones, tablets, laptops) are equipped with significant processing, storage and sensing capabilities, as well as wireless connectivity through cellular (3G,4G), WIFI and Bluetooth. They provide ubiquitous Internet access, primarily through their cellular connection and secondary through WIFI, and enable a big number of new applications. Among these applications, video is increasingly popular. However meeting the growing demand for high quality video is currently a challenge in cellular networks. Indeed cellular traffic is growing exponentially (almost tripling every year). In 2014 video content accounted for an 55% of all the world's internet traffic. According to a Cisco report [8] by 2019 online video will be responsible approximately for the fourth-fifths (72%) of the global internet traffic. This dramatic increase in demand poses a challenge for 3G networks, which is likely to remain in 4G networks as well. Furthermore, the data rate of the cellular connection may fluctuate over time (e.g. throughout the day) the service loss rate can be as high as 50% and coverage can be spotty depending on the location and user mobility.

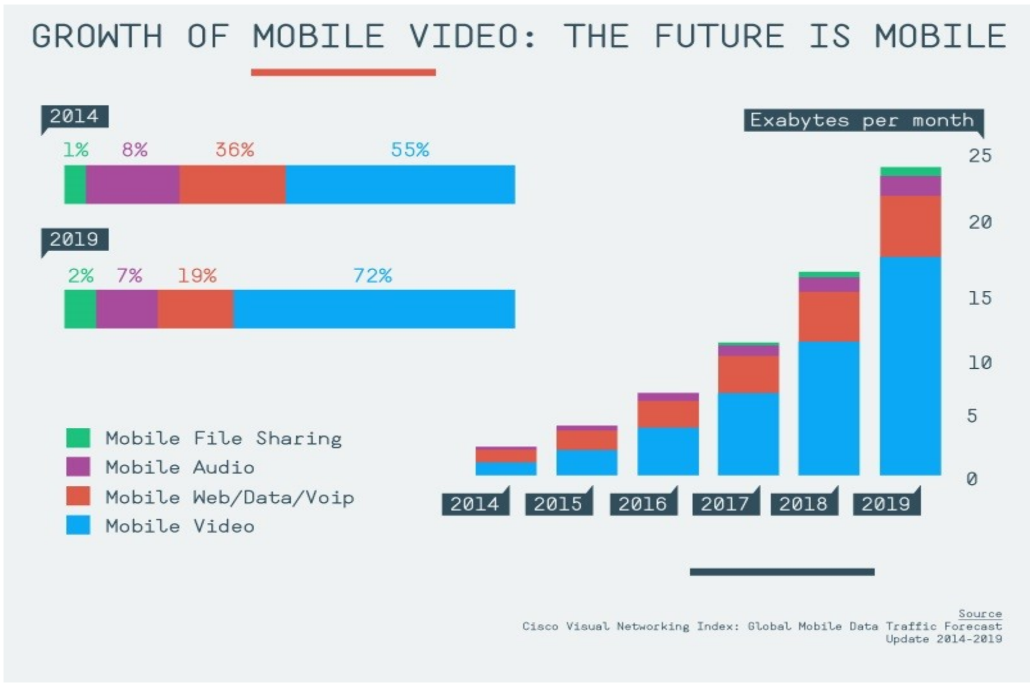


Figure 1: Prediction about mobile Video usage by Cisco.

In this thesis we are interested in the scenario where a group of users with mobile devices within proximity of each other, are interested in watching the same video at the same time. Watching the video on one phone screen is not comfortable for more than two people, so users may also prefer to watch the video on their own screens. The default operation today is that each user with a cellular connection downloads part of the video independently from the server. However each phone's individual cellular connection may not be sufficient for providing high video quality. In such a case some or all of the users may have poor or intermittent cellular connectivity, depending on the coverage of their providers, or may face congestion in the local network (e.g. when they use WIFI to download).

Fortunately, because the users engage in a group activity, this scenario naturally lends itself to cooperation. Furthermore, when every phone has multiple parallel connections (e.g. 3G, WIFI and Bluetooth), there are even more available resources that, if properly used can further improve the user experience. We propose a novel cooperative cheme for video downloading to a group of smart devices (smartphones, tablets etc) within proximity of each other. Each phone utilizes simultaneously two network interfaces:

1. Cellular Interface to connect to the video server and download parts of the bit-stream file.
2. WiFi interface to connect to the rest of the group and exchange downloaded parts.

2 Video Encoding

2.1 General

Video encoding also known as video transcoding is basically a process of converting a given video input into a digital format that is compatible with most types of web players and mobile devices.

2.2 Video Encoding Formats

All the video people watch via computers, mobile devices, tablets, and other mobile devices have undergone a video encoding process to transform them from the original source format into a format viewable on the said devices. This is because different browsers, video players and devices support or can play videos in different formats. The most important point to remember is that each video format comes with its own specifications such as video codec with H.264 and next H.625 and HEVC, WMV and Webfm, audio codec with MP3 and WMA, video or audio bit-rates and containers such as FLV, MP4 and AVI among others. However not all computers, tablets, and smart devices support the same video format which makes it necessary for a video to be encoded into the required format. So, basically if a video has to meet its precise specifications, or if the video's current format is not the same as the required end format, then it must be encoded into the right format using video encoding technology. In this thesis we focus on H.264/SVC encoding.

3 Scalable Video Coding (SVC)

3.1 General

The evolution of digital video technology and the continuous improvements in communication infrastructure is propelling a great number of interactive multimedia applications, such as real time video conference, web video streaming and mobile TV, among other. The new possibilities on interactive video usage have created an exigent market of consumers which demands the best video quality wherever they are and whatever their network support. On this purpose each transmitted video must match the receiver's characteristics such as the required bit-rate, resolution and frame rate, thus aiming to provide the best quality subject to receiver's and network's limitations. Besides, the same link is often used to transmit to either restricted devices such as small cell phones or to high-performance equipments, e.g. HDTV workstations. In addition, the stream should adapt to wireless lossy networks. Based on this reasoning, these heterogeneous and non-deterministic networks represent a great problem for traditional video encoders which do not allow for on the fly video streaming adaptation. To overcome all these difficulties the concept of scalability for video coding has been lately proposed as an efficient solution for supporting distinct video processing capabilities. The principle of a scalable video encoder is to break the conventional single-stream video in a multi-stream flow, composed by distinct and complementary components, often referred to as layers. Therefore receivers can select and decode different number of layers, each corresponding to distinct video characteristics, in accordance with the processing constrains of both the network and the device itself. The layered structure of any scalable video content can be defined as the combination of a base layer and several additional enhancement layers. The base layer corresponds to the lowest supported video performance, whereas the enhancements layers allow for the refinement of the aforementioned base layer. The adaptation is based on a combination within the set of selected strategies for the spatial, temporal and quality scalability. In the last years, several specific scalable video profiles have been included in video codecs such as MPEG-2 (2000), H.263 (2000) and MPEG-4 Visual(2004). However all these solutions present a reduced coding efficiency when compared with non-scalable video profiles. As a consequence, scalable profiles have been scarcely utilized in real applications, whereas widespread solutions have been strictly limited to non-scalable, single layer coding schemes. In October 2007, the scalable extension of the H.264 codec, also known as H.264/SVC was jointly standardized by ITU-T VCEG and ISO MPEG as an amendment of the H.264/AVC (Advanced Video Coding) standard. Among several innovative features, H.264/SVC combines temporal, spatial and quality scalability into a single multi-layer stream.

3.2 Overview of H.264/SVC

The sophisticated architecture of the H.264/SVC standard is particularly designed to increase the codec capabilities while offering a flexible encoder solution that supports three different scalabilities: temporal, spatial and SNR quality.

In H.264/SVC, each spatial dependency layer requires its own prediction module in order to perform both motion-compensated prediction and intra prediction within the layer. Besides, there is a SNR refinement module that provides the necessary mechanisms for quality scalability within each layer. The dependency between subsequent spatial layers is managed by the inter-layer prediction module, which can support reusing of motion vectors, intra texture or residual signals from inferior layers so as to improve compression efficiency. Finally the scalable H.264/SVC bit-stream is merged by the so called mutliplex, where different temporal, spatial and SNR levels are simultaneously integrated into a single scalable bit-stream. In the sections below will be described each scalability type individually.

3.2.1 Temporal Scalability

The term “temporal scalability” refers to the ability to represent video content with different frame rates by as many bit-stream subsets as needed. Encoded video streams can be composed by three distinct type of frames: I (intra), P (predictive) or B (Bi-predictive). I frames only explore the spatial coding within the picture, i.e. compression techniques are applied to information contained only inside the current picture, not using references to any other picture. On the contrary, both P and B frames do have interrelation with different pictures, as they explore directly the dependencies between them. While in P frames inter-picture predictive coding is performed based on (at least) one preceding reference picture, B-frames consist of a combination of inter picture bi-predictive coding (i.e. samples of both previous and posterior reference pictures are considered for the prediction). In addition, the H.264 standard family requires the first frame to be an instantaneous Decoding Refresh (IDR) access unit, which corresponds to the union of one I frame with several critical non-data related information (e.g. the set of coding parameters). Generally speaking the GoP structure specifies the arrangement of those frames within an encoded video sequence.

Certainly, the singular dependency and predictive characteristics of each frame type imply divergent coded video stream features. In H.264/SVC the basis of a temporal scalability is found on the GoP structure, since it divides each frame into distinct scalability layers (by jointly combining I,P and B frame types). As for the H.264/SVC codec the GoP definition can be rephrased as the arrangement of the coded bit-stream's frames between two successive pictures of the temporal base layer. However the frames of the temporal base layer do not necessarily need to be an I frame. Actually only the first picture of a video stream is strictly forced to be coded as an I frame and to be included in the initial IDR access unit.

In order to increase the flexibility of the codec the H.264/SVC standard defines a distinct structure for temporal prediction, where reference frames for each video sequence are reorganized in a hierarchical tree scheme. This tree scheme improves the distribution of information between consecutive frames.

A so-called group of pictures (GoP) consists of a key picture and hierarchically predicted B pictures that are located between the key picture of the current GoP and the key picture of the previous GoP. The hierarchical prediction structure is illustrated for a group of 8 pictures, which is employed in the example above.

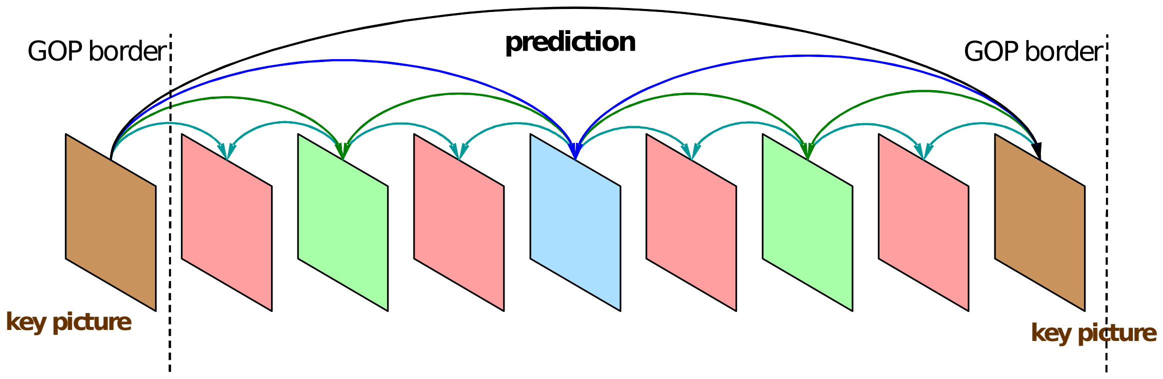


Figure 2: Hierarchical prediction structure for groups of 8 pictures.

The key pictures are either intra-coded (e.g. in order to enable random access) or inter-coded by using previous key pictures as reference for motion-compensated prediction. The sequence of key pictures is independent from any other pictures of the video sequence, and in general it represents the minimal temporal resolution that can be decoded. Furthermore, the key pictures can be considered as re-synchronisation points between encoder and decoder when arbitrarily discardable MGS enhancement layers are added.
The first picture of a video sequence is always intra-coded as IDR picture and represents a special GoP, which consists of exactly one picture.
The remaining pictures of a GoP are hierarchically predicted. For the example in Figure 2, the picture in the middle (blue) is predicted by using the surrounding key pictures as references. It depends only on the key pictures, and represents the next higher temporal resolution together with the key pictures. The pictures of the next temporal level (green) are predicted by using only the pictures of the lower temporal resolution as references, etc. It is obvious that this hierarchical prediction structure inherently provides temporal scalability.

3.2.2 Spatial Scalability

The spatial scalability is based on representing, through a layered structure, videos with distinct resolutions, i.e. each enhancement layer is responsible for improving the resolution of lower layers. H.264/SVC encoder introduces a more flexible and complex prediction module called Inter-Layer Prediction

(ILP). The main goal of the ILP module is to increase the amount of reused data in the prediction from inferior layers, so that the reduction of redundancies increases the overall efficiency. ILP module supports 3 prediction techniques shown below:

1. **Inter-Layer Motion Prediction**: the motion vectors from lower layers can be used by superior enhancement layers. In some cases, the motion vectors and their attached information must be rescaled so as to adjust the values to the correct equivalents in higher layers.
2. **Inter-Layer Intra Texture Prediction:** H264/SVC supports texture prediction for internal blocks within the same reference layer (intra). The intra block predicted in the reference layer can be used for other blocks in superior layers. This module up-samples the resolution of inferior layer's texture to superior layer resolutions, subsequently calculating the difference between them.
3. **Inter-layer Residual Prediction:** As a consequence of several coding process observations, it has been identified that when two consecutive layers have similar motion information, the inter-layer residues register high correlation. Based on this, in H.264/SVC the inter-layer residual prediction method can be used after the motion compensation process to explore redundancies in the spatial residual domain.

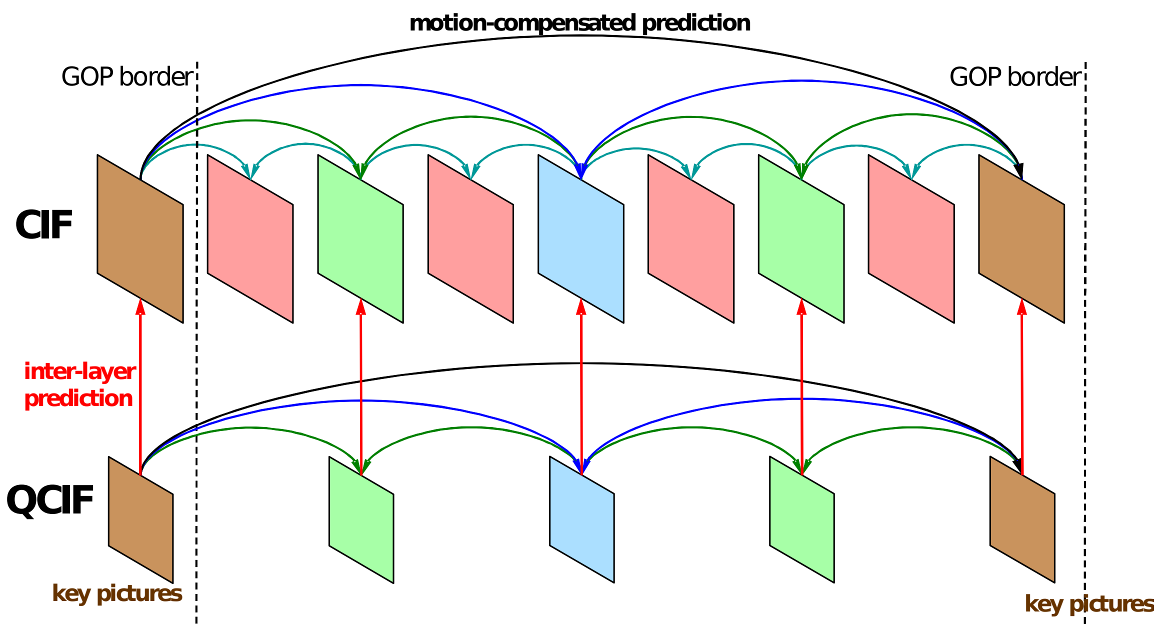


Figure 3: Coding structure with 2 spatial layers

3.2.3 SNR Scalability

The SNR scalability or quality scalability empowers transporting complementary data in different layers in order to produce videos with distinct quality levels. This scalability type basically hinges on adopting distinct quantization parameters for each layer. The H264/SVC standard supports three distinct SNR scalability modes:

- **Coarse Grain Scalability (CGS):** in this strategy, each layer has an independent prediction procedure in a similar fashion to the SNR scalability of MPEG-2. In fact, the CGS strategy can be regarded as a special case of spatial scalability when consecutive layers have the same resolution.
- **Medium Grain Scalability (MGS):** The MGS approach increases efficiency by using a more flexible prediction module, where both types of layer (base and enhancement) can be referred. However this strategy can induce a drifting effect (i.e. it can introduce a synchronism offset between the encoder and the decoder) if only the base layer is received. To solve this issue, the MGS specification proposes the use

of periodic key pictures, which immediately resynchronizes the prediction module.

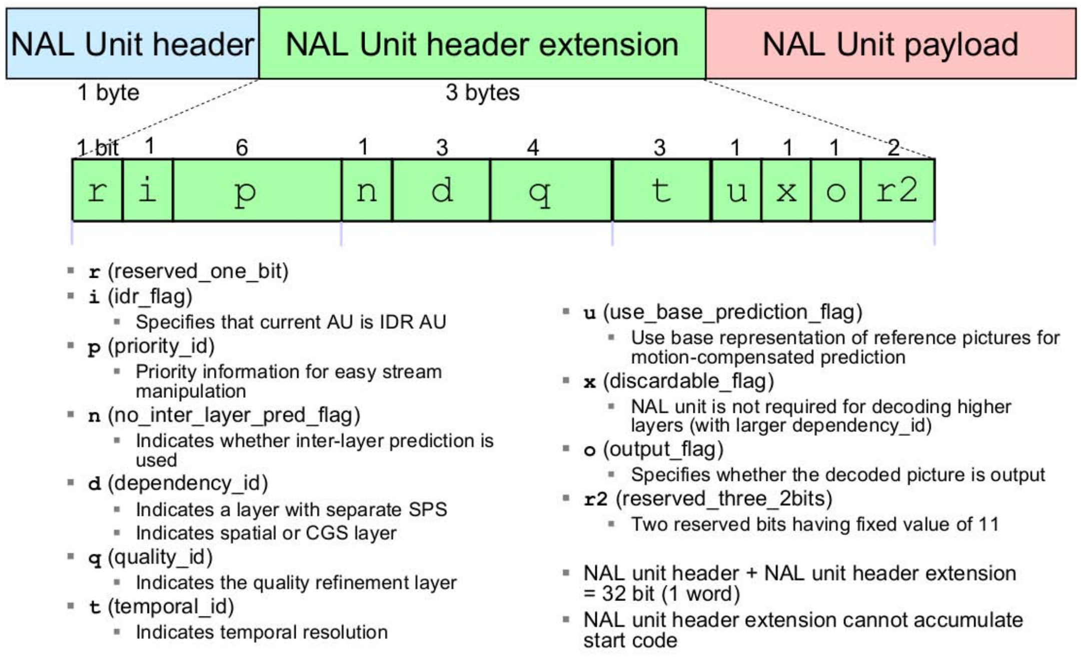
- **Fine Grain Scalability (FGS):** This version of the SNR scalability aims at providing a continuous adaptation of the output bit-rate in relation to the real network bandwidth. FGS employs an advanced bit-plane technique where different layers are responsible for transporting distinct subsets of bits corresponding to each data information. The scheme allows for data truncation at any arbitrary point in order to support the progressive refinement of transform coefficients. In this type of scalability, only the base layer casts motion prediction techniques.

3.3 SVC Architecture

The basic building blocks in SVC design are video coding layer (VCL) and network abstraction layer (NAL) which is similar to H.264/AVC. VCL is responsible for coding the source contents and NAL is responsible for formatting the VCL representation and provides header information appropriate for transmission through transport layer and storage media.

3.3.1 NAL Units

The encoded video data is organized in NAL units, each of which is effectively a packet that contains an integer number of bytes. The first byte of each NAL unit is a header byte that contains an indication of the type of data in the NAL unit. All the remaining bytes contain payload data of the type indicated by the header. The NAL unit structure definition specifies a generic format for use in both packet-oriented and bit-stream-oriented transport systems, and series of NAL units generated by an encoder is referred to as a NAL unit stream.



- **r** (reserved_one_bit)
- **i** (idr_flag)
  - Specifies that current AU is IDR AU
- **p** (priority_id)
  - Priority information for easy stream manipulation
- **n** (no_inter_layer_pred_flag)
  - Indicates whether inter-layer prediction is used
- **d** (dependency_id)
  - Indicates a layer with separate SPS
  - Indicates spatial or CGS layer
- **q** (quality_id)
  - Indicates the quality refinement layer
- **t** (temporal_id)
  - Indicates temporal resolution
- **u** (use_base_prediction_flag)
  - Use base representation of reference pictures for motion-compensated prediction
- **x** (discardable_flag)
  - NAL unit is not required for decoding higher layers (with larger dependency_id)
- **o** (output_flag)
  - Specifies whether the decoded picture is output
- **r2** (reserved_three_2bits)
  - Two reserved bits having fixed value of 11

- NAL unit header + NAL unit header extension = 32 bit (1 word)
- NAL unit header extension cannot accumulate start code

Figure 4: NAL unit Packet

3.3.2 NAL Units in Byte-Stream format

Some systems require delivery of the entire or partial NAL unit system as an ordered stream of bytes or bits within which the locations of NAL unit boundaries need to be identifiable from patterns within the coded data itself. For use in such systems, the H.264/AVC and HEVC specifications define a byte stream format. In the byte stream format, each NAL unit is prefixed by a specific pattern of three bytes called a start code prefix. The boundaries of the NAL unit can then be identified by searching the coded data for the unique start code prefix pattern. The use of emulation prevention bytes guarantees that start code prefixes are unique identifiers of the start of a new NAL unit. A small amount of additional data (one byte per video picture) is also added to allow decoders that operate in systems that provide streams of bits without alignment to byte boundaries to recover the necessary alignment from the data in the stream. Additional data can also be inserted in the byte stream format that allows expansion of the amount of data to be sent and can aid in achieving more rapid byte alignment recovery, if desired.

3.3.3 NAL Units in Packet Transport System

In other systems (e.g., IP/RTP systems), the coded data is carried in packets that are framed by the system transport protocol, and identification of the boundaries of NAL units within the packets can be established without use of start code prefix patterns. In such systems, the inclusion of start code prefixes in the data would be a waste of data carrying capacity, so instead the NAL units can be carried in data packets without start code prefixes.

3.3.4 VCL and non – VCL NAL Units

NAL units are classified into VCL and non-VCL NAL units. The VCL NAL units contain the data that represents the values of the samples in the video pictures, and the non-VCL NAL units contain any associated additional information such as parameter sets (important header data that can apply to a large number of VCL NAL units) and supplemental enhancement information (timing information and other supplemental data that may enhance usability of the decoded video signal but are not necessary for decoding the values of the samples in the video pictures).

3.3.5 Parameter Sets

A parameter set is supposed to contain information that is expected to rarely change and offers the decoding of a large number of VCL NAL units. There are two types of parameter sets:

1. sequence parameter sets, which apply to a series of consecutive coded video pictures called a coded video sequence.
2. picture parameter sets, which apply to the decoding of one or more individual pictures within a coded video sequence.

The sequence and picture parameter-set mechanism decouples the transmission of infrequently changing information from the transmission of coded representations of the values of the samples in the video pictures. Each VCL NAL unit contains an identifier that refers to the content of the relevant picture parameter set and each picture parameter set contains an identifier that refers to the content of the relevant sequence parameter set. In this manner, a small amount of data (the identifier) can be used to refer to a larger amount of information (the parameter set) without repeating that information within each VCL NAL unit. Sequence and picture parameter sets can be sent well ahead of the VCL NAL units that they apply to, and can be repeated to provide robustness against data loss. In some applications, parameter sets may be sent within the channel that carries the VCL NAL units (termed "in-band" transmission). In other applications, it can be advantageous to convey the parameter sets "out-of-band" using a more reliable transport mechanism than the video channel itself.

3.3.6 Access Units

A set of NAL units in a specified form is referred to as an access unit. The decoding of each access unit results in one decoded picture. Each access unit contains a set of VCL NAL units that together compose a primary coded picture. It may also be prefixed with an access unit delimiter to aid in locating the start of the access unit. Some supplemental enhancement information containing data such as picture timing information may also precede the primary coded picture. The primary coded picture consists of a set of VCL NAL units consisting of slices or slice data partitions that represent the samples of the video picture. Following the primary coded picture may be some additional VCL NAL units that contain redundant representations of areas of the same video picture. These are referred to as redundant coded pictures, and are available for use by a decoder in recovering from loss or corruption of the data in the primary coded pictures. Decoders are not required to decode redundant coded pictures if they are present. Finally, if the coded picture is the last picture of a coded video sequence (a sequence of pictures that is independently decodable and uses only one sequence parameter set), an end of sequence NAL unit may be present to indicate the end of the sequence; and if the coded picture is the last coded picture in the entire NAL unit stream, an end of stream NAL unit may be present to indicate that the stream is ending.

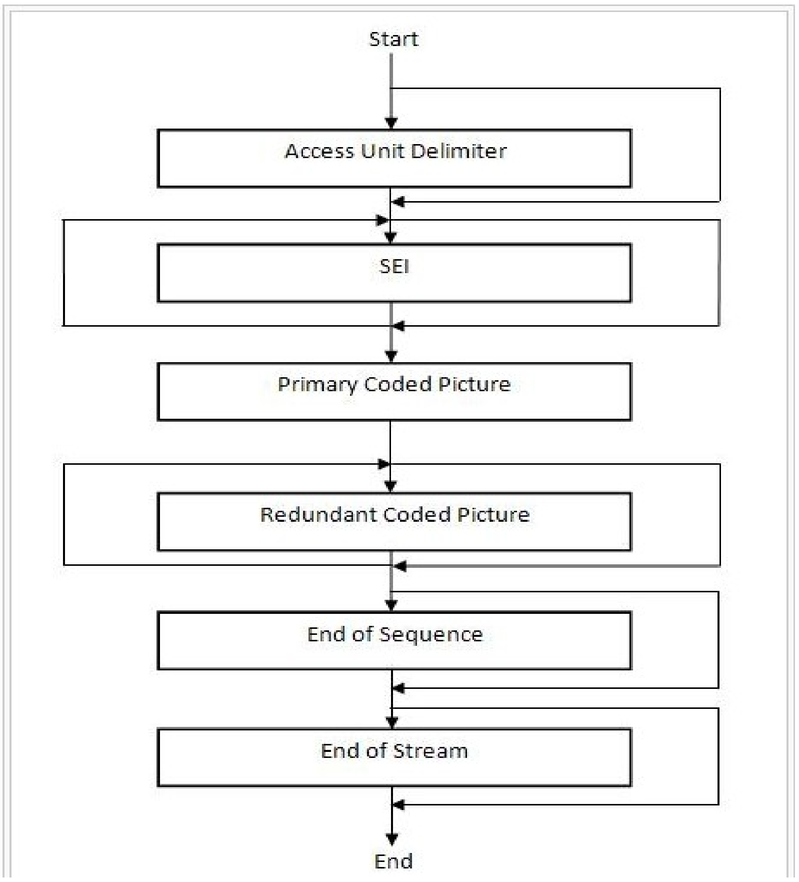


Figure 5: Structure of NAL Access Unit

3.3.7 Coded Video Sequences

A coded video sequence consists of a series of access units that are sequential in the NAL unit stream and use only one sequence parameter set. Each coded video sequence can be decoded independently of any other coded video sequence, given the necessary parameter set information, which may be conveyed "in-band" or "out-of-band". At the beginning of a coded video sequence is an instantaneous decoding refresh (IDR) access unit. An IDR access unit contains an intra picture which is a coded picture that can be decoded without decoding any previous pictures in the NAL unit stream, and the presence of an IDR access unit indicates that no subsequent picture in the stream will require reference to pictures prior to the intra picture it contains in order to be decoded. A NAL unit stream may contain one or more coded video sequence.

4 Network Coding

Network coding is an elegant and novel technique introduced the last few years to improve network throughput and performance. It is expected to be a critical technology for networks of the future.

4.1 Introduction

Networked systems arise in various communication contexts such as phone networks, the public Internet, peer to peer networks, ad-hoc wireless networks, and sensor networks. Such systems are becoming central to our way of life. During the past half a century, there has been significant body of research effort devoted to the operation and management of networks. A pivotal, inherent premise behind the operation of all communication networks today lies in the way information is treated. Whether it is packets in the Internet, or signals in a phone network, if the originate from different sources, they are transported much in the same manner as cars on a transportation network of highways, of fluids through a network of pipes. Namely, independent information streams are kept separate. Today, routing, data storage, error control and generally all network functions operate on this principle.
Only recently, with the advent of network coding, the simple but important observation was made that in communication networks, we can allow nodes to not only forward but also process the incoming independent information flows. At the network layer, for example, intermediate nodes can perform binary addition of independent bit-streams, whereas, at the physical layer of optical networks, intermediate nodes can superimpose incoming optical signals. In other words,data streams that are independently produced and consumed do not necessarily need to be kept separate when they are transported throughout the network. There are ways to combine and later extract independent information. Combining independent data streams allows to better tailor the information flow to the network environment and accommodate the demands of specific traffic patterns. This shift in paradigm is expected to revolutionize the way we manage, operate, and understand organization in networks, as well as to have a deep impact on a wide range of

areas such as reliable delivery, resource sharing, efficient flow control, network monitoring and security.

This new idea is based in some part on the prediction that computational processing is becoming cheaper (according to Moore's law) and therefore the bottleneck has shifted to network bandwidth for support of ever-growing demand in applications. Network coding utilizes cheap computational power to dramatically increase network throughput. The interest in this area continues to increase network throughput. The interest in this area continues to increase as we become aware of new applications of these ideas in both the theory and practice of networks, and discover new connections with many diverse areas.

4.1.1 Benefits

Network coding promises to offer benefits along very diverse dimensions of communication networks, such as throughput, wireless resources, security, complexity and resilience to link failures.

4.1.1.1 Throughput

The first demonstrated benefits of network coding were in terms of throughput when multi-casting.

Example 1.1 (Butterfly network)

Figure 6 below depicts a communication network represented as a directed graph where vertices correspond to terminals and edges correspond to channels. This example is commonly known in the network coding literature as the butterfly network. Assume that we have slotted time, and that through each channel we can send one bit per time slot. We have two sources S_1 and S_2 and two receivers R_1 and R_2. Each source produces one bit per time slot which we denote by x_1 and x_2 respectively.

If receiver R_1 uses all the network resources by itself, it could receive both sources. Indeed, we could route the bit x_1 from source S_2 along the path {AC, CE, EF} and the bit x_2 from source S_2 along the path {BF} as depicted in figure 6 (b).

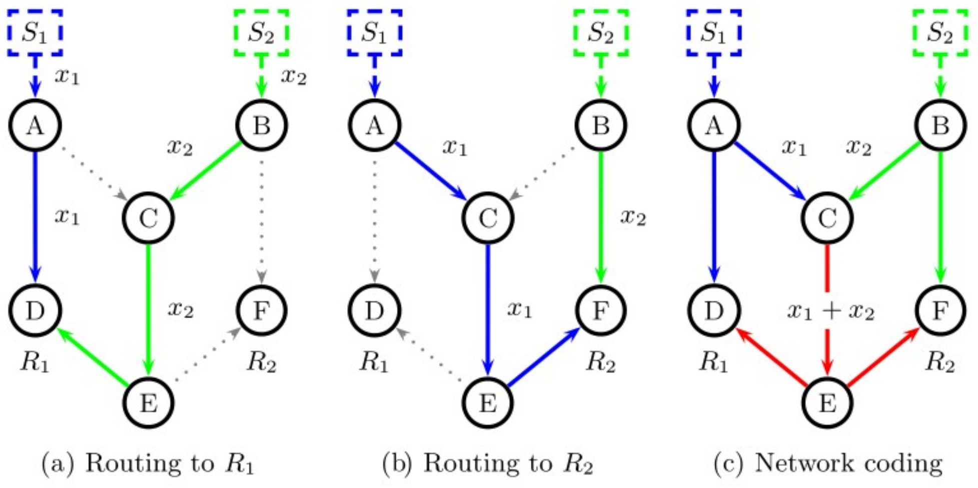


Figure 6: The Butterfly network. Sources S_1 and S_2 multicast their information to receivers R_1 and R_2.

Now assume that both receivers want to simultaneously receive the information from both sources. That is, we are interested in multi-casting. We then have a contention for the use of edge CE, arising from the fact that through this edge we can only send one bit per time slot. However, we would like to simultaneously send bit x_1 to reach receiver R_2 and bit x_2 to reach receiver R_1.

Traditionally, information flow was treated like fluid through pipes, and independent information flows were kept separate. Applying this approach we would have to make a decision at edge CE : Either use it to send bit $x_{1,}$ then receiver R_1 will only receive x_1 while receiver R_2 will receive both x_1 and x_2.

The simple but important observation is that we can allow intermediate nodes int the network to process their incoming information streams, and not just forward them. In particular, node C can take bits x_1 and x_2 and **xor** them to create a third bit $x_3 = x_1 + x_2$ which it can then send through edge CE (the **xor** operation corresponds to addition over binary field). R_1 receives { x_1, $x_1 + x_2$} and can solve this system of equations to retrieve x_1 and x_2. Similarly R_1 receives { x_2, $x_1 + x_2$} and can solve this system of equations to retrieve x_1 and x_2.

The example presented above shows that if we allow intermediate node in the network to combine information streams and extract the information at the receivers, we can increase the throughput when multi-casting.

4.1.1.2 Wireless Resources

In a wireless environment, network coding can be used to offer benefits in terms of battery life, wireless bandwidth, and delay.

Example 1.2

Consider a wireless ad-hoc network, where devices A and C would like to exchange the binary files x_1 and x_2 using device B as a relay. We assume that time is slotted, and that a device can either transmit or receive a file during a timeslot (half-duplex communication). Figure 7 depicts on the left the standard approach : Nodes A and C send their files to the relay B, who in turn forwards each file to the corresponding destination.

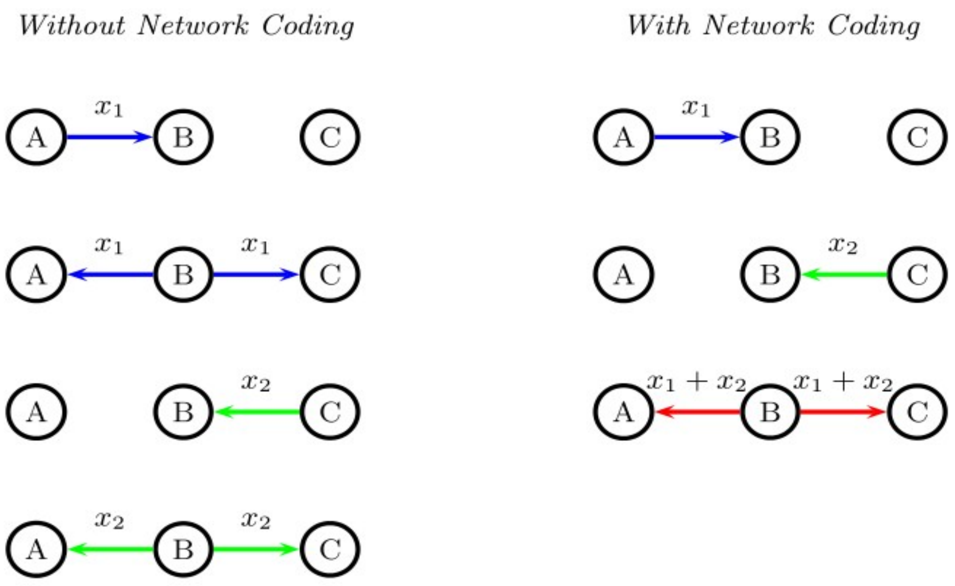


Figure 7: Nodes A and C exchange information via relay B. The network coding approach uses one broadcast transmission less.

The network coding approach takes advantage of the natural capability of wireless channels for broadcasting to give benefits in terms of resource utilization, as illustrated in Figure 7. In particular, node C receives both files x_1

and x_2 and bitwise **xor**s them to create the file $x_1 + x_2$, which it then broadcasts to both receivers using a common transmission. Node A has x_1 and can thus decode x_2. Node C has x_2 and can thus decode x_1.

This approach offers benefits in terms of energy efficiency (node B transmits once instead of twice), delay (the transmission is concluded after three instead of four timeslots), wireless bandwidth (the wireless channel is occupied for a smaller amount of time), and interference (if there are other wireless nodes attempting to communicate in the neighbourhood).

The benefits on the example presented above arise from that broadcast transmissions are made maximally useful to all their receivers. Network coding for wireless is examined in the second part of this review. As we will discuss there $x_1 + x_2$ is nothing but some type of binning or hashing for the pair ($x_1 + x_2$) that the relay needs to transmit. Binning is not a new idea in wireless communications. The new element is that we can efficiently implement such ideas in practice, using algebraic operations.

4.1.1.3 Security

Sending linear combinations of packets instead of not coded data offers a natural way to take advantage of multipath diversity for security against wiretapping attacks. Thus systems that only require protection against such simple attacks, can get it “for free” without additional security mechanisms.

4.1.2 Challenges

The deployment of network coding is challenged by a number of issues that will also be discussed in more detail throughout the review. Here we briefly outline some major concerns.

4.1.2.1 Complexity

Employing network coding requires nodes in the network to have additional functionalities. An important question in network coding research today is assessing the complexity and performance.

4.1.2.2 Security

Networks where security is an important requirement, such as networks for banking transactions, need to guarantee protection against sophisticated attacks. The current mechanisms in place are designed around the assumption that the only eligible entities to tamper with the data are the source and the destination. Network coding on the other hand requires intermediate routers to perform operations on the data packets. Thus deployment of network coding in such networks would require to put in place mechanisms that allow network coding operations without affecting the authenticity of the data.

4.1.2.3 Integration with Existing Infrastructure

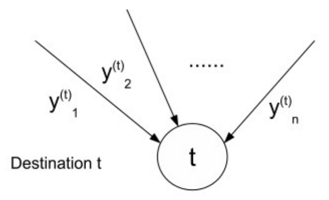
As communication networks evolve toward an ubiquitous infrastructure, a challenging task is to incorporate the emerging technologies such as network coding, into the existing network architecture. Ideally, we would lkie to be able to profit from the leveraged functionalities network coding can offer, without incurring dramatic changes in the existing equipment and software. A related open question is, how could network coding be integrated in current networking protocols. Making this possible is also an area of current research.

5 Framework of Network Coding

In this chapter, we explain the practical framwork of network coding in order to use linear coding and random linear coding.

5.1 Linear Coding

Network coding give to a source node the opportunity to multicast information in a rate approaching the smallest minimum cut between the source and any receiver. Since then, coding methods to achieve the multicast max-flow rates had been researched. In the paper [1] a signle source multicasting using linear coding on a finite field (Galois Field) is presented. Linear coding is a notable coding method, because it only uses a linear transformation, such a plus and multiplexing, which can be implemented with low computational cost. In order to apply the low complex linear coding on network packets, all packets are assumed to have identical size, and the same sized packets are viewed as vectors over a finite Galois Field. (GF) F_q. With this view of packets, a node linearly transforms incoming packets into outgoing packets like vectors. In linear network coding, all encoding vectors are linear functions over a finite field GF(q) and so we can write the global transfer function from the source s to each destination node t as depicted in the figure below.



$$\begin{vmatrix} y^{(t)}_1 \\ \cdot \\ \cdot \\ \cdot \\ y^{(t)}_{n'} \end{vmatrix} = \begin{vmatrix} g^{(t)}_{11} \cdots\cdots g^{(t)}_{1n} \\ \cdot \qquad \cdot \\ \cdot \qquad \cdot \\ \cdot \qquad \cdot \\ g^{(t)}_{n'n} \cdots\cdot g^{(t)}_{n'n} \end{vmatrix} \begin{vmatrix} p_1 \\ \\ \\ \\ p_n \end{vmatrix}$$

$$Y^{(t)} = G^{(t)} \times P$$

Figure 8: Received encoding data at destination node t.

n linear network coding the each received packet at destination node t relates to the source packets. We call the coefficients [g_{i1}^t, g_{i2}^t, ..., g_{in}^t] a global encoding vector for the encoded packet y_i^t for $1 \le i \le n$. Provided that table $G^{(t)}$ in **Figure 8** has rank n the destination t can recover P=[p_1, p_1, … p_n] by multiplying $Y^{(t)}$ by a left inverse of $G^{(t)}$: $G^{(-t)}$ x $Y^{(t)} =$IxP.

The main benefit of linear coding is low complexity. The low complexity is emphasized at the size of GF becomes smaller. A smaller field makes the operations of linear coding much simpler.

Although linear coding decreases the computation cost of all coding and decoding operations, we still have to decide a local encoding vector that must guarantee decoding every node. Some polynomial algorithms are presented to select the local encoding vectors, however polynomial algorithms use global network topology and need to inform each node which coefficient to use. Namely the polynomial algorithms operate in a centralized way which is not sufficient enough. In contrast to the deterministic codes using the centralized way, there exists a simple and fully decentralized method. This method is the random linear coding.

5.2 Random Linear Coding

In order to use linear coding, we need to select coefficients, and polynomial algorithms can be used to select these coefficients. The algorithms assign the valid network codes that guarantee decoding at every node, but they operate in a centralized and deterministic way. One notable coding method to operate in a fully decentralized way is random linear coding, which randomly selects its local encoding vector on a fixed Galois field F_q without any coordination [2]. For example let be $a_1, a_2, \ldots, a_n$ are randomly chosen over GF(q) to comprise a local encoding vector of intermediate node υ in **Figure 8** for encoded data $a_1 y_{e1} + a_2 y_{e2} + \ldots + a_n y_{en}$.

With random linear coding, each node can randomly select a local vector for each outgoing packet, namely an information vector. This random selection causes a local encoding vector to continuously change even at the same node. Hence, it is hard to determine which local encoding vectors are sequentially used in order to generate the coded packet. If we do not know the local encoding vectors that are sequentially used to generate the coded packet, we cannot find a global encoding vector for the coded packet, and therefore decoding is impossible. Hence, we need a method to track the sequentiallity used local encoding vectors for eventual decoding. The method is packetization

The packetization in [2][3] suggested inserting a global encoding vector in a special packet header and transmitting the global encoding vector with the encoded data. So, let us assume that the source s sends n packets ($(p_1, p_2, \dots, p_n)$. When linear network coding is applied, each packet flowing in a network is a linear combination of the source packets as :

ith received coded packet at node υ : $y_i^{(\upsilon)} = \sum g_{i,j}^{(\upsilon)} p_j$

where the $(p_j)_j = 1, \dots n$ are n packets generated from the source. The sequence of coefficients for source packets in a coded packet $y_i^{(\upsilon)}$ forms a global encoding vector $[g_{i,1}^{\upsilon}, g_{i,2}^{\upsilon}, \dots., g_{i,n}^{\upsilon}]$.

As shown in this expression of a coded packet, a global vector contains the information how the received encoded packet $y_i^{(\upsilon)}$ is related with the source packets. Hence, a packet including n extra symbols for a global encoding vector ($[g_{i,1}, g_{i,2}, \dots., g_{i,n}]$) in a special header with encoded data allows destinations to decode without knowing the network topology or the encoding rules in a completely decentralized way.

However the packetization is not enough to practically use random linear coding. For simplicity we assume that all incoming packets from all incoming edges arrive synchronously without delay. But in a real network, packets arrive asynchronously and depart with arbitrarily varying rates, delays and losses. To handle this real network condition it is suggested to use buffering as depicted in **Figure 9** below. We can see that a receiving node saves incoming encoded packets and their global vectors delivered through its incoming edges. When a node should transmit a new coded packet, the node uses all or a portion of encoded packets saved in its local buffer and hence its incoming packets do not need to arrive synchronously at the node through its incoming edges.

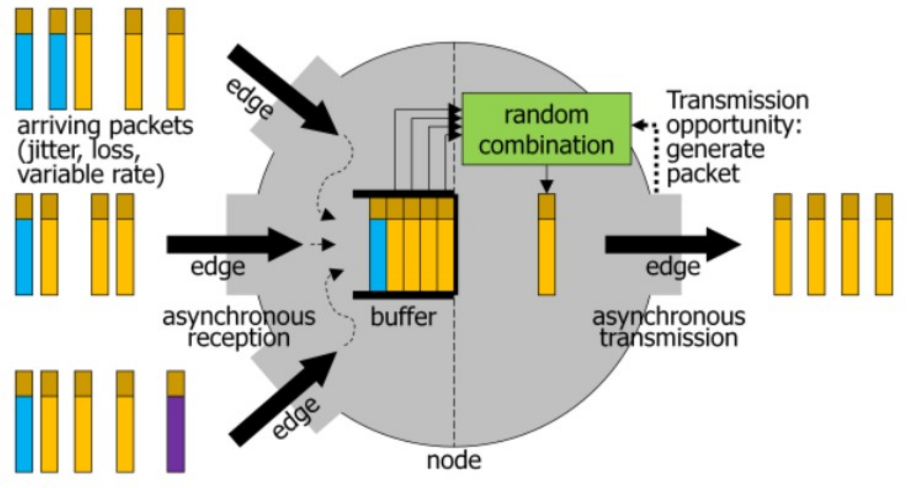


Figure 9: Buffering received packets (global vectors and information vectors)

The process to generate a new coded packet is almost same as the process explained in next section. The difference is that is uses encoded packets (information vectors) and their global encoding vectors saved in its local buffer. More specifically, node υ generates a newly encoded packet by linearly combining randomly chosen coefficients $[l_1, l_2, \ldots, l_n](1 \times n' \text{ matrix})$ and the stored encoded packets $[[y_1^{(\upsilon)}][y_2^{(\upsilon)}]\ldots[y_{n'}^{(\upsilon)}]]$. These two vectors (new encoded data and new global encoding vector for the new encoded data) are delivered in the same packet by using packetization. Thanks to the packetization and buffering, we can use random linear coding for practical networks, even for wireless networks which are unreliable and unstable.In order to ultimately recover all of the source's packets, the rank of the node must be equal to the total number of source packets. This means that in the worst case, a node is supposed to store all received innovative packets in its local buffer until it can recover all source packets at once. In practice, the size of the local buffer in a node cannot infinitely increases, and hence we need to limit the number of encoding packets which is saved in a local buffer. The limitation of encoding packets is based on a *generation,* n is called a *generation size* [4].

5.2.1 Decoding, Vector Space and Rank.

With packetization and buffering, a node stores received encoded packets in its local buffer and the stored encoded packets compose a matrix of a node υ. Let G_{υ} denote global encoding vectors of the matrix of a node υ. At any point of time, a node υ is associated with the *vector space* spawned by the global encoding vectors G_{υ}. The dimension of that vector space is denoted as D_{υ} and also as the rank of the matrix. In the rest of this thesis, that rank and dimension is called the *rank of a node*. The rank of a node is a direct metric for the amount of useful received packets. When a received packet increases the rank of the receiving node, the received packets brings useful information and is called *innovative*. On the contrary, if a received packet does not increas the rank of the receiving node, the packet is not useful and is non-innovative. The non-innovative packet contains redundant information and cannot contribute to recovering the source packets. Thus, depending on whether a received packet is innovative or not, the receiving node can decide to drop or store the received packet. If the matrix of the node $G^{(\upsilon)}$ has full rank(n)

$$\begin{vmatrix} y^{(v)}_{1} \\ y^{(v)}_{2} \\ \cdots\cdots \\ y^{(v)}_{n} \end{vmatrix} = \begin{vmatrix} g^{(v)}_{1,1} & g^{(v)}_{1,2} & \cdots & g^{(v)}_{1,n} \\ g^{(v)}_{2,1} & g^{(v)}_{2,2} & \cdots & g^{(v)}_{2,n} \\ & \cdots\cdots\cdots\cdots & & \\ g^{(v)}_{n,1} & g^{(v)}_{n,2} & \cdots & g^{(v)}_{n,n} \end{vmatrix} \begin{vmatrix} p_{1} \\ p_{2} \\ \cdots \\ p_{n} \end{vmatrix} \qquad G^{-(v)} \begin{vmatrix} y^{(v)}_{1} \\ y^{(v)}_{2} \\ \cdots\cdots \\ y^{(v)}_{n} \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & .. & 0 \\ 0 & 1 & 0 & .. & 0 \\ & & \cdots\cdots\cdots\cdots & & \\ 0 & 0 & 0 & .. & 1 \end{vmatrix} \begin{vmatrix} p_{1} \\ p_{2} \\ \\ p_{n} \end{vmatrix}$$

$$Y^{(v)} = G^{(v)} \times P \qquad G^{-(v)} \times Y^{(v)} = I \times P$$

Figure 10: A decoding process at node v.

or a sub-matrix with full rank ($r < n$) exists, the node can decode n or r source packets at the same time. The decoding amounts to inverting the matrix or sub-matrix, for instance using Gaussian elimination.

5.3 Efficient Wireless Broadcasting with Network Coding

In this chapter, we focus on using network coding as an efficient method to broadcast data to the whole network, where the cost is the total transmission number. Finding an optimal solution with classical store-and-forward routing for energy efficient broadcasting in wireless networks is known to be NP-complete. However, with network coding we can find the optimal solution in polynomial time. The polynomial solution with network coding has two parts: a cost minimization part that decides a rate of each node to minimize the sum of the rates and a coding part that finds valid network codes to guarantee decoding at all nodes.

For the coding part, results from information theory suggest a notable coding method, random linear coding, which can execute encoding and decoding in a fully distributed way without any coordination, and still guarantees with high probability that every node can recover the original packets from the source. Along with the simplicity, random linear coding also guarantees with high probability to achieve the bound of a maximum multicast rate if the multicast session is executed sufficiently long. To take advantage of these novel results, our broadcast protocol uses random linear coding.

When random linear coding is used, a remaining problem is to minimize the cost for efficient broadcasting, and to minimize the cost is essentially deduced to find the average transmission rates of the nodes – the average rates are established on the entire duration of the stream (even for packet networks). As a result, finding an optimal solution to minimize the cost requires finding the optimal average rate of every node. To do this a quite simple approach is adopted. Considering that a coded packet can bring new information to multiple receivers at the same time, the maximum-efficiency benefit of wireless network coding can be achieved to make every transmission bring new information to all receivers. For practical solution, our approach intends to achieve the maximal-efficiency-benefit of wireless network coding with a simple rate selection method.

6 Cooperative Mobile Wireless Systems

6.1 Overview

When several users are interested in the same content and they are in proximity of each other, some of them may be able to use device to device connections e.g. through WiFi or Bluetooth, to get the content in a cooperative and opportunistic way. Opportunistic device to device communication is often used for the purpose of offloading the cellular network. For example consider the scenario in which device to device and cellular connections are used to disseminate the content, considering the social ties and geographical proximity. Instead of offloading cellular networks our goal is to use cellular and local connectivity so as to essentially allow each user to enjoy the aggregated downlink rate.

Furthermore, cooperation between mobile devices for content dissemination or in delay tolerant networking, possibly taking into account social ties, has extensively been studied. However dissemination of content stored on a mobile device is only a special case of our framework, which uses only the local links, but not the downlinks. More importantly, we exploit single-hop broadcast transmissions, as opposed to multi-hop, peer to peer communication that exploits mobility (at the expense of delay, which is crucial in our setting), but ignores broadcast.

Regarding the cellular connections of multiple smart devices (smartphones, tablets, e.t.c.) we try to improve the download rate and jointly utilize local connections. After the announcement of Android Ice Cream Sandwich (Android 4.0) devices provide peer to peer connectivity using WiFi direct. WiFi Direct is an industrial standard that prompts WiFi – equipped devices to establish ad-hoc peer-to-peer connections, while maintaining the infrastructure mode connection to the internet. This is achieved by having one of the devices act as a virtual access point to provide infrastructure to the rest. So the core idea of the broadcasting algorithm used, is to exploit the broadcast nature of the wireless channel, while the core idle of WiFi Direct is to offer ad-hoc connectivity to mobile devices.

6.2 Network Coding in Cooperative Wireless Systems

Cellular links (3G or 4G) as well as WiFi suffer from packet loss due to noise and interference. One possible solution to this problem is to have several devices in a close proximity help each other with retransmissions of lost packets. Network coding is particularly beneficial as it can make each retransmission maximally useful to all nodes. In our scheme we try to utilize two interfaces 3G and WiFi for data delivery. Network coding has been applied to peer to peer (P2P) networks for content distribution and live streaming. The scheme we are investigating is proposed in [5]. It is called MircroCast and is explicitly designed to exploit WiFi overhearing and network coding, and to avoid fundamental weaknesses of previous algorithms (more details about Microcast will be discussed below). This system is designed in a way to exploit cooperation and broadcast of wireless transmissions. MicroCast architecture is modular, thus allowing for swapping various components (e.g different wireless technologies, streaming protocols, scheduling and dissemination algorithms). Our simulation environment is based on MicroCast however there was a need for some assumptions to be done in order to simulate that system. The whole environment that we trying to simulate is described in the chapter below.

7 Environment to Simulate

7.1 Environment Overview

We consider the scenario presented in Figure 11 in which a group of smart device users, within proximity to each other are all interested in downloading and watching the same video at the same time. To achieve this goal, we use cooperation among the users. Furthermore, each phone simultaneously uses two interfaces: The cellular interface 3G to connect to the server and the local interface (WiFi) to connect to all other users in the group. Each phone downloads segments of the video from the server and shares these with the rest of the group locally.

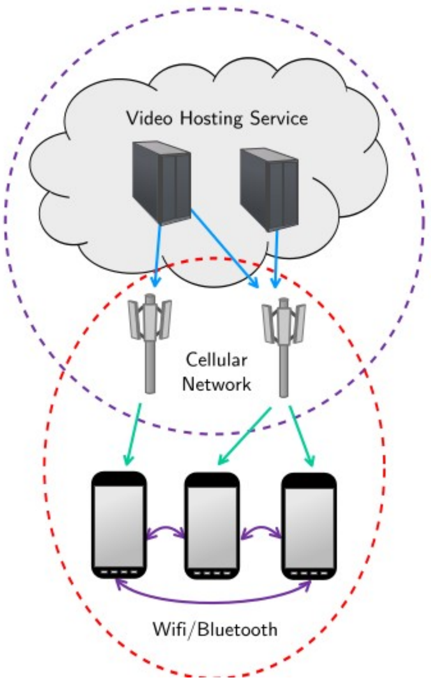


Figure 11: MicroCast scenario

We optimized our system under certain assumptions that hold for our setup of interest: First, there is a small number of users (up to 6-7). Second these users know and trust each other, as it is the case in the motivating examples we provided in the introduction. Third, all users are within proximity of each other and all local links have similar rates on average. This is important as we need to use only single-hop broadcast transmissions. Fourth in every phone we use the cellular connection for the downlink, and WiFi to establish local, device to device links. This has the implication that we can use the two connections (cellular and WiFi) on each phone simultaneously and independently. This would not be possible otherwise, as we discuss later. Note that cellular and WiFi connections are used in parallel, but one connects directly to the server (3G), while the other connects to the other users (WiFi). We do not use both these connections to connect a user directly to the server through two different paths.

7.2 Microcast Architecture

Microcast is a scheme developed for this purpose and consists of some concrete modules presented here. The architecture of Microcast is depicted in figure below:

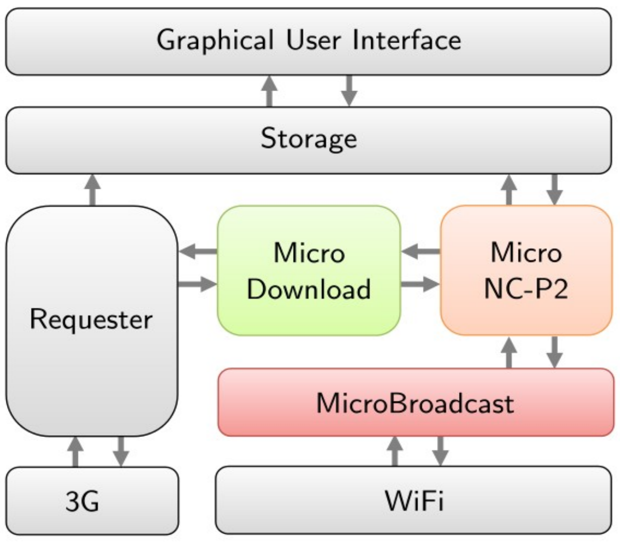


Figure 12: Architecture of MicroCast

As we can see Microcast consists of several dependent modules. The modules that we are interested in are Requester, MicroNC-P2, MicroDownload and Microbroadcast. Through this thesis we tried to simulate the environment described above using algorithm MicroNC-P2 on our custom simulator which is a main part of Microcast architecture. This algorithm is responsible for distributing segments of data through the local wireless network. MicroNC-P2 specifically exploits the broadcast capability provided be its lower layer

MicroBroadcast to distribute segments quickly and efficiently(MicroNC-P2 algorithm will be described in details in next section). Microbroadcast on the other side implements a comprehensive networking stack which currently operates on wireless technologies including WiFi. The most important functionality that MicroBroadcast provides is the ability to pseudo-broadcast over WiFi. Nonetheless, MicroBroadcast also supports unicast, reliable and unreliable message exchange between the network participants over WiFi.
Requester is the part of the system that is responsible for the reception of the segments of the video source.

7.3 MicroNC-P2 Algorithm

Each segment downloaded by a smart device, is divided into packets and distributed to the remaining devices using the local network. To do so a custom dissemination scheme had been designed that exploits the benefits of overhearing and network coding. At a high level this scheme takes advantage of pseudo-broadcast i.e. unicast and overhearing, to reduce the number of transmissions. Furthermore, instead of disseminating regular packets, our scheme disseminates random linear combinations of packets that belong on the same segment. i.e. dimensions of subspaces or network coded packets. This is to maximize the usefulness of overheard packets. We describe in detail how we use the network coding in next section.

7.3.1 Algorithm Description

In MicroNC-P2, a phone, A, periodically advertises the segments that it currently has to its neighbours. Then a neighbour, lets say phone B, requests segments that it does not have based on the advertisement. Upon receiving the request, phone A sends the requested segments to phone B. More specifically:

- When phone B requests a segment from phone A, it takes into account previously overheard dimensions of the subspace representing segment s.In particular, it explicitly indicates in the request how many additional dimensions (coded packets) it needs to receive to decode s. This reduces the number of dimensions to be sent.

- When phone A is about to serve a segment requested be phone B, it first checks if there are pending requests for the same segments from other neighbours. If there are, it finds the maximum number of dimensions requested among these requests. Denote this maximum dimension by *d.* If there is none, *d* is the number of dimensions requested by B. Afterwards, A serves *d* dimensions of segment *s* to B. The other phones, which need up to *d* dimensions of *s, s*hould be able to get the dimensions through overhearing.

After serving B, A notifies all phones that requested some dimensions of segment s. Upon receiving the notification, these phones check if they received all the necessary dimensions to decode s. If not, they send requests for additional dimensions. This is necessary because overhearing is not guaranteed for all dimensions sent by A and for all phones. Overhearing and unicast effectively allow for pseudo-broadcast. As described, the amount of traffic saved by pseudo-broadcasting segment s depends not only on the quality of the overhearing but also on the number of requests of segments s from other phones that A processes at the time of broadcasting.

To be concrete consider a local network consisting of four smart devices A, B, C and D. After finishing downloading segment s using 3G, A advertises it to B, C and D. After that B, C, and D send requests for this segment to A. For simplicity, assume perfect overhearing. First, consider the case where all requests arrive at A at a similar time. In this case, when A serves, e.g., B's request for segment s, A removes C and D's requests for s. Effectively, A serves all B, C and D using a single transmission of s. Now, consider the case where the request from D arrives later than the time A initially serves s to B and C. The late arrival of D's request could be due to various reasons such as large receiving and sending queues of D. A now has to serve D all dimensions of s even though D may overheard some dimensions initially sent by A to B (or C). Apparently, A needs to send more than needed.

To address this issue, we propose an initial push of segment s. Specifically, when A finishes downloading a segment s, it sends all dimensions (coded packets regarding this segment) of s to a randomly selected neighbour before advertising the segment. By doing so, A ensures that the initial dissemination of segment s is taken into account in subsequent requests of segment s (if any) of A's neighbours. This effectively creates a perfect synchronization of the reception of the initial requests of segment s. We provide a space-time diagram of MicroNC-P2 in figure below:

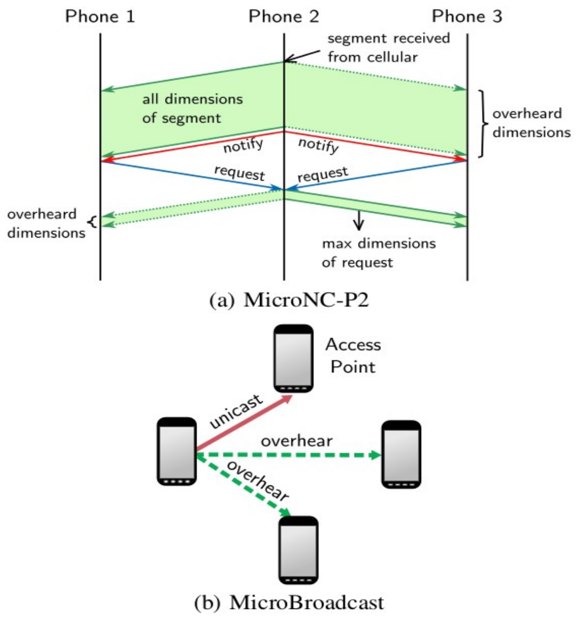


Figure 13: Space-time diagram of MicroNC-P2 and Microbroadcast.

The pseudocode of MicroNC-P2 algorithm is depicted below:

MicroNC-P2 Algorithm

```
1.  When a new segment s is received
2.      if s is received be the requester then
3.          //initial push
4.          Send all dimensions of s to a neighbour
5.      end if
6.      Add s to the list of segments to be advertised
7.  end when
8.  when a packet p is received from A
9.      if p is an advertisement
10.         or notification containing s then
11.         //subsequent pulls exploit overhearing
12.     else if p is a request for d dimensions of s then
13.         Add this request to the request queue
14.     else if p is a dimension of s then
15.         Progressively decode s using p
16.     end if
17. end when
18. when there is a request for d dim of s from A
19.    //pseudo-broadcast
20.    if there are other similar requests then
21.        Let d be the largest requested dimension
22.        Remove these requests from the request queue
23.    end if
24.    Send d dimensions of s to A
25. end when
```

7.4 Reception Rate

The system described in this chapter aims to allow each smart device receive at a rate equal to the sum of the 3G/4G download rates of all users in the cooperative group. This is the best rate we can hope to achieve since this is the maximum rate at which our network gets new information from the server. This rate may be higher than the playback rate of the video and this is useful to reduce the probability of a buffer underflow during playback in case of video streaming. Downloading at a rate higher than the playback rate requires caching of the stream locally. However this is not a problem for modern mobile devices which come with large storage space.

8 Network Simulator

8.1 Implementation Details

In this thesis we are trying to simulate the environment described in the previous section. To do this we had to make some assumptions about the system's behaviour. Here, we are going to describe the implementation details and some design choices we made to address some major challenges that we've faced. Our custom network simulator is developed in C++. The input to this system is the path with the video bit-stream file produced after encoding of a raw (.yuv file) with SVC codec using JSVM tool [7] and a packet trace file corresponding to the given stream. Some network parameters such as loss probabilities on downlink (3G) channel, loss probabilities on the local wireless network, number of users e.t.c. are denoted inside the code of the simulator and can change any time. The output of the network simulator is a bit-stream file collected from each one of the users. This bit-stream file contains part of the information of the original bit-stream file due to the loss of various coded packets which leads to the inability of the node to decode the regarding segments. All these are described schematically in the figure below:

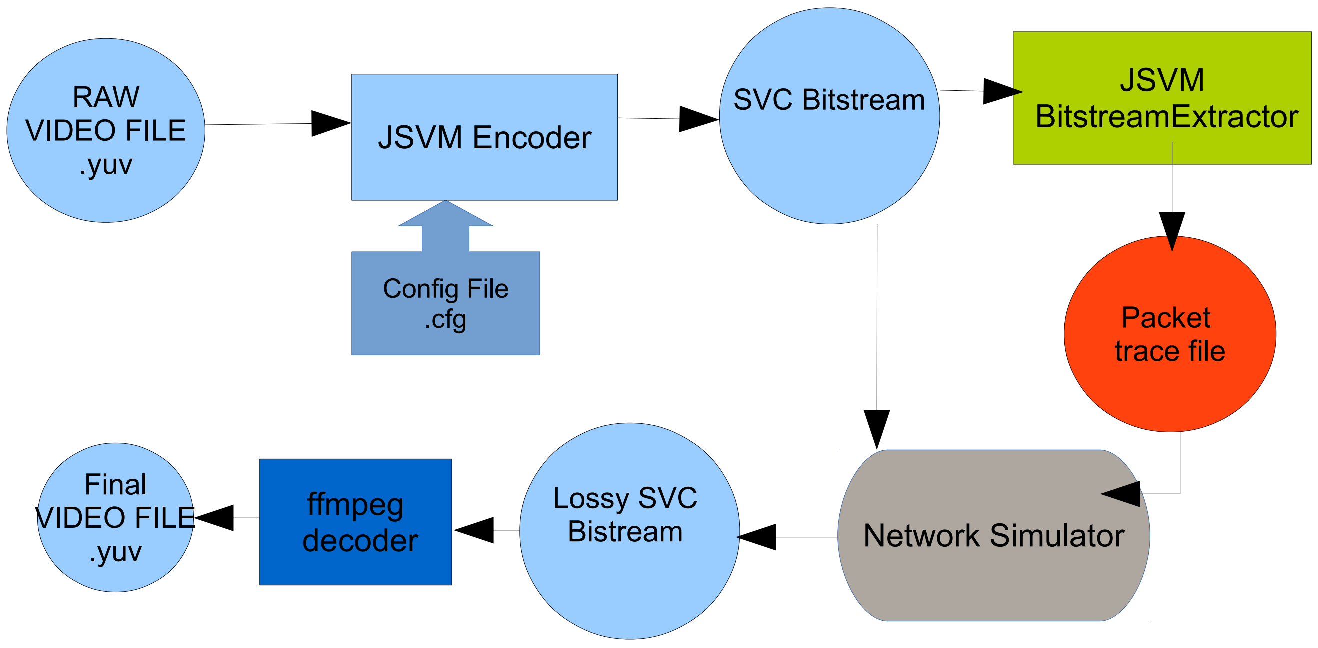


Figure 14: Simulator software chain

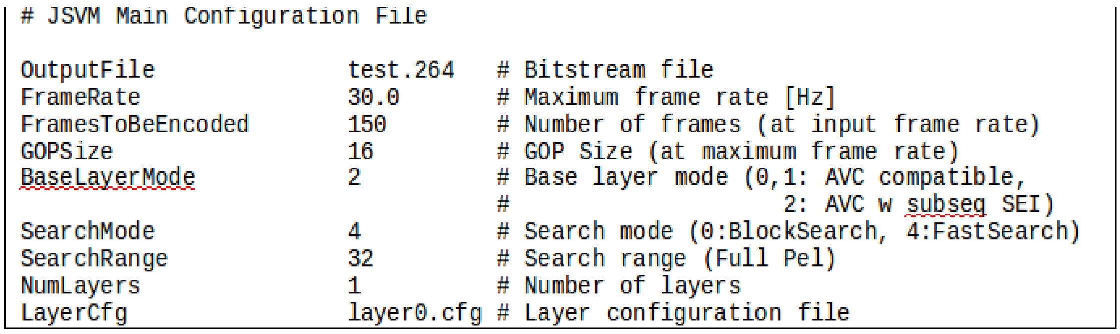
8.2 JSVM

8.2.1 General Information

The JSVM (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). Since the SVC project is still under development, the JSVM Software as is also under development and changes frequently.
The JSVM software is written in C++ and is provided as source code. Using this software a user can encode a video file into an SVC bit-stream with various number of layers regarding temporal, spatial, SNR scalability or combination of these.

8.2.2 Configuration Files

All the parameter for the encoding of a video file are included inside a config file (.cfg). An example of such a file is depicted on the figure below:



```
# JSVM Main Configuration File

OutputFile              test.264   # Bitstream file
FrameRate               30.0       # Maximum frame rate [Hz]
FramesToBeEncoded       150        # Number of frames (at input frame rate)
GOPSize                 16         # GOP Size (at maximum frame rate)
BaseLayerMode           2          # Base layer mode (0,1: AVC compatible,
                                   #                    2: AVC w subseq SEI)
SearchMode              4          # Search mode (0:BlockSearch, 4:FastSearch)
SearchRange             32         # Search range (Full Pel)
NumLayers               1          # Number of layers
LayerCfg                layer0.cfg # Layer configuration file
```

Figure 15: main configuration file for single layer coding

Inside the main configuration file we set some parameters about the encoded file like FrameRate, FramesToBeEnconded, GoPSize, NumLayers etc. In this example in the case of single layer coding one layer configuration file is needed “layer0.cfg”. In this file we denote the name path for the input raw video file and many other parameters like frame width, height etc. In our example all these are depicted right below:

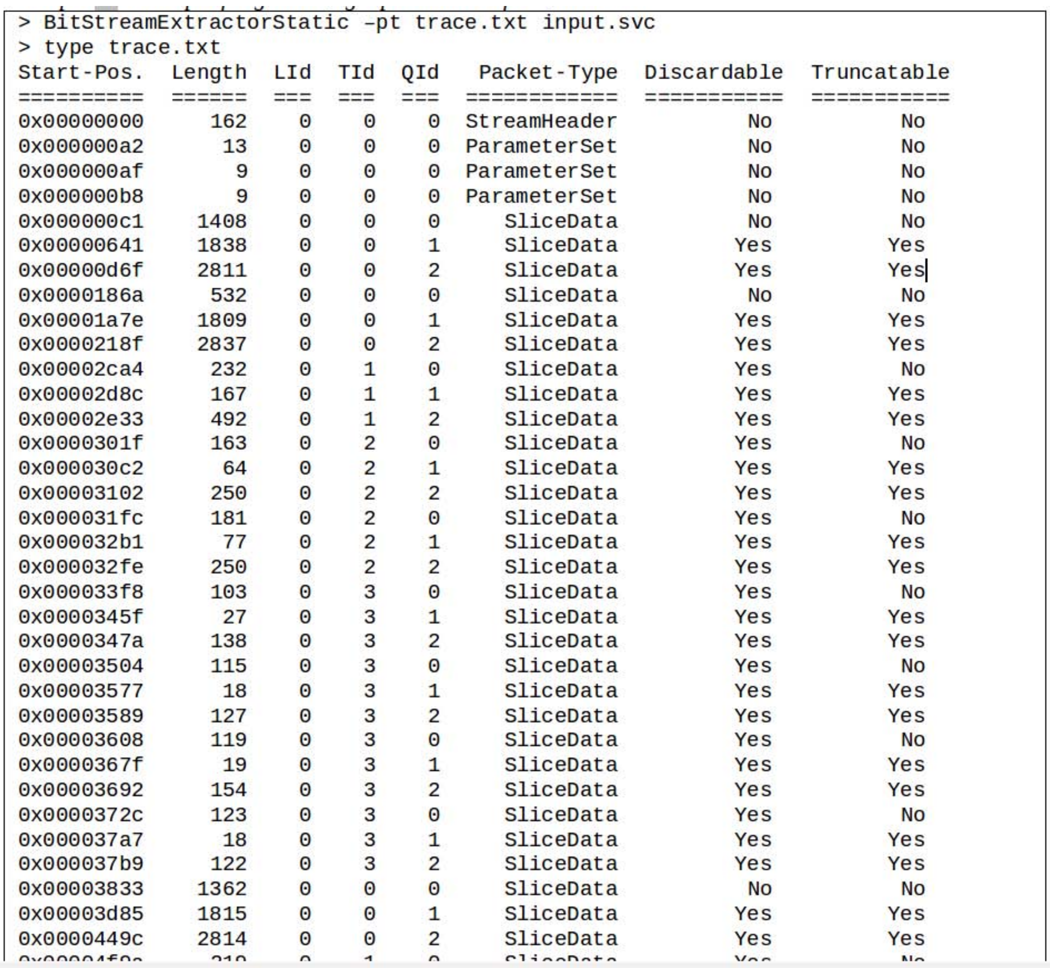
```
# JSVM Layer Configuration File

InputFile           BUS_QCIF15.yuv # Input   file
SourceWidth         176            # Input   frame width
SourceHeight        144            # Input   frame height
FrameRateIn         15             # Input   frame rate [Hz]
FrameRateOut        15             # Output  frame rate [Hz]
```

Figure 16: Layer configuration file “layer0.cfg” for single layer coding.

The encoding of the raw video file given as input has as result a bit-stream file which is SVC encoded according to the parameters defined on the corresponding configuration files. This SVC bit-stream file is used as input in our network simulator.

8.2.3 Packet Trace Files

A second input needed in order to start the simulation is a packet trace file. This packet trace file always refers to a given svc encoded stream. This trace file is a text file, which specifies various parameters for each single packet inside the given bit-stream. These parameters include the start position (in unints of bytes) of the packet inside the bit-stream file, the length of the packet (in units of packets) the values of dependency_id (Lid), temporal_level(TId) and quality_level (Qid) for the packet, the type of packet, and two flags which indicate whether the packet is discardable or truncatable (truncatable packets are not supported in SVC). An example of such a trace file is depicted in the figure below:



```
> BitStreamExtractorStatic –pt trace.txt input.svc
> type trace.txt
Start-Pos.  Length  LId  TId  QId   Packet-Type  Discardable  Truncatable
==========  ======  ===  ===  ===  ============  ===========  ===========
0x00000000     162    0    0    0  StreamHeader           No           No
0x000000a2      13    0    0    0  ParameterSet           No           No
0x000000af       9    0    0    0  ParameterSet           No           No
0x000000b8       9    0    0    0  ParameterSet           No           No
0x000000c1    1408    0    0    0     SliceData           No           No
0x00000641    1838    0    0    1     SliceData          Yes          Yes
0x00000d6f    2811    0    0    2     SliceData          Yes          Yes
0x0000186a     532    0    0    0     SliceData           No           No
0x00001a7e    1809    0    0    1     SliceData          Yes          Yes
0x0000218f    2837    0    0    2     SliceData          Yes          Yes
0x00002ca4     232    0    1    0     SliceData          Yes           No
0x00002d8c     167    0    1    1     SliceData          Yes          Yes
0x00002e33     492    0    1    2     SliceData          Yes          Yes
0x0000301f     163    0    2    0     SliceData          Yes           No
0x000030c2      64    0    2    1     SliceData          Yes          Yes
0x00003102     250    0    2    2     SliceData          Yes          Yes
0x000031fc     181    0    2    0     SliceData          Yes           No
0x000032b1      77    0    2    1     SliceData          Yes          Yes
0x000032fe     250    0    2    2     SliceData          Yes          Yes
0x000033f8     103    0    3    0     SliceData          Yes           No
0x0000345f      27    0    3    1     SliceData          Yes          Yes
0x0000347a     138    0    3    2     SliceData          Yes          Yes
0x00003504     115    0    3    0     SliceData          Yes           No
0x00003577      18    0    3    1     SliceData          Yes          Yes
0x00003589     127    0    3    2     SliceData          Yes          Yes
0x00003608     119    0    3    0     SliceData          Yes           No
0x0000367f      19    0    3    1     SliceData          Yes          Yes
0x00003692     154    0    3    2     SliceData          Yes          Yes
0x0000372c     123    0    3    0     SliceData          Yes           No
0x000037a7      18    0    3    1     SliceData          Yes          Yes
0x000037b9     122    0    3    2     SliceData          Yes          Yes
0x00003833    1362    0    0    0     SliceData           No           No
0x00003d85    1815    0    0    1     SliceData          Yes          Yes
0x0000449c    2814    0    0    2     SliceData          Yes          Yes
```

Figure 17: Example of packet trace file

8.3 Simulation Scenario

In this section we are going to describe the simulation scenario. As we said in previous chapters we consider the scenario where a group of smart-devices users, within proximity of each other, are all interested on downloading the same video file at the same file. We assume that this video file is located on a server in SVC coded bit-stream format. The information contained inside the bit-stream file is going to be packetized according to the packet trace file. The packet trace file given as input from the user is a mapping of the data that each packet is going carry. After the packetization of the information on *n* packets we create segments of data. This is done by denoting the segment size (in our case we chose 10) which means that each segment consists of 10 packets. Segments have also another meaning to our simulation, it is the group of packets that are going to be randomly linearly combined to create coded packet as described in the section of random linear network coding. After the segmentation is finished we allocate on users the set of segments that each one is going to download from the 3G downlink. To do this we use a simple round robin algorithm so we can say that segments are uniformly distributed among users.

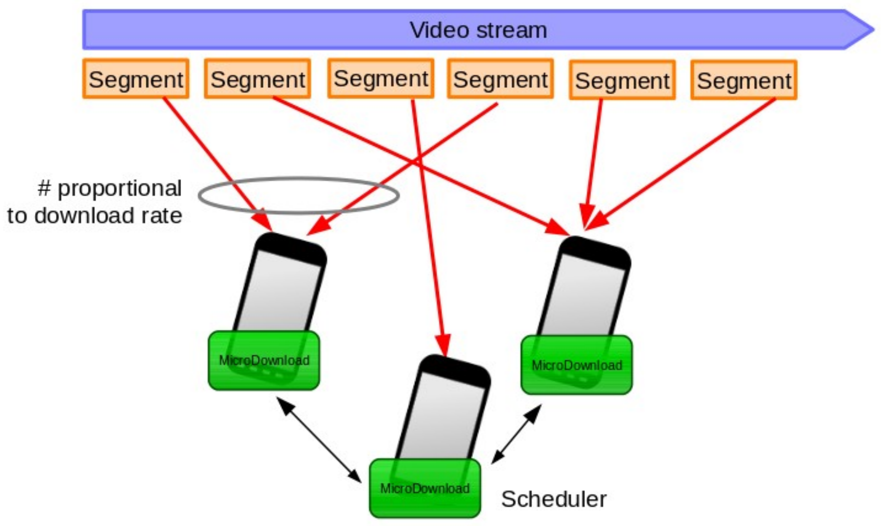


Figure 18: Video stream segments allocation among users.

We assume that all users can achieve approximately the same download rate regarding the downlink. So in general when the simulation starts each nodes starts downloading data using 3G connection these data correspond to the packets that belong on segments that were previously allocated on each user. Whenever one or more users have downloaded a whole segment they create a group of coded packets regarding to that segments and whenever the wiifi channel is free to use, the broadcast of packets can start. Each time only one user can use the wifi channel to broadcast the coded packets and it releases it whenever all the coded packets regarding the latest downloaded segment are broadcasted. During broadcast some packets may not be delivered due to a wlan loss probability defined as parameter to our simulator. We also assume that users exchange data (messages) which are relative to information about the coded packets that each one has (advertisements) and also about coded packets that each one needs to decode a specific segment (requests), all these messages are transmitted through the local network and can be delivered or not according to the same wlan loss probability. The transmission of data among the users is performed according to the MicroNC-P2 (described on previous section 7.3) and for all the transmissions of data that take place inside the local wireless network we use network coding technique. The network coding scheme used here is generation-based network coding over finite field $GF(2^8)$. According to this scheme each segment is broken down into m packets which together form one generation (or segment) where m is the segment or generation size. Each packet contains n bytes, and we treat each byte as a symbol in $GF(2^8)$. We also augment each packet with the m coding coefficients, each of which is selected uniformly at random from $GF(2^8)$. Thus each packet can be seen as a vector of length n + m symbols from $GF(2^8)$. Phone *A* sends to phone *B* linear combinations of packets of the same segment, where the coding coefficients used to create linear combinations are selected uniformly at random from $GF(2^8)$

Phone *B* can decode a segment upon receiving m linearly independent combinations of packets of the segment. Let *M* denote a matrix formed by m linearly independent packets : $M=[E|C]$, where *E* is of size $m\,x\,n$ and *C* is the coefficient matrix of size $m\,x\,m$. Original packets can be recovered by finding the inverse of *C*.In particular $C^{-1}x[E|C]=[B|I]$ where *B* is the matrix of size $m\,x\,n$ whose row *i* is the data of packet *i* and *I* is the $m\,x\,m$ identity matrix.

Each time a node receives a coded packet from another user that had previously broadcasted if compares it's coefficient vector contained in the coded packet header and checks if the coefficient vector just received is linearly independent comparing to the pool of coefficient vectors that he has received regarding the same segment. If the new coded packet has coefficient vector that is linearly independent comparing to the ones received from coded packets regarding the same segment user adds this packet to his incoming queue with the coded packets. In this case the coded packet is called *innovative packet.* On the other case user rejects this packets and waits to receive another coded packet etc.

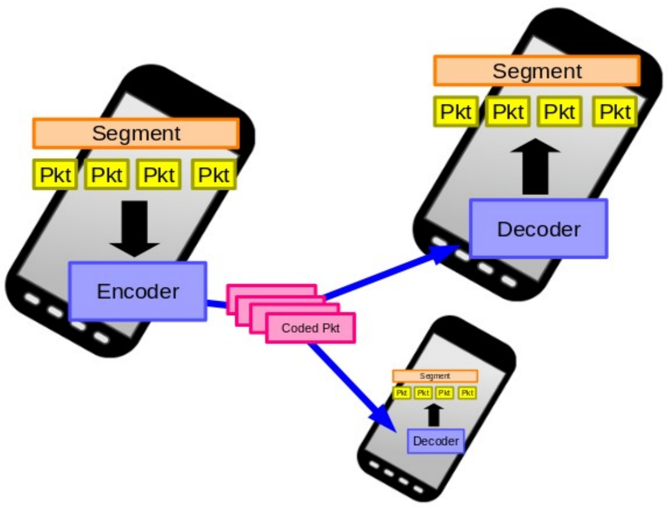


Figure 19: Efficient broadcast using network coding.

The extra information needed for packet decoding is hidden inside the header of each of the packet with some extra sub-header. The simulation time stops whenever all the transmissions inside the local network have been concluded and there is no node with something to transmit. After this, each node composes the video stream file by extracting and adding all information of the packets received according to its sequence id.

9 Experiments and Results

9.1 Single Layer coding

At this point we will analyse the method we used in order to test and evaluate the performance of the transmission of an SVC bit-stream using our simulator. In order to do that we choose to perform single layer coding on the input video file using the JSVM software. For the tests that we performed JSVM software is run in scalable mode for supporting temporal scalability (see previous section 3.2.1). Single layer coding means that the encoded bit-stream does not provide several spatial resolutions or several bit-rates for a specific spatio-temporal resolution. Thus the generated bit-stream does not contain spatial or CGS/MGS enhancement layer.

The video we are going to encode is a raw video file in QCIF resolution (176x144 samples) with a frame rate of 30Hz and the configuration files used to perform this encoding are depicted below:

```
# JSVM Main Configuration File

OutputFile                streamSVC.264   # bit-stream file
FrameRate                 30.0       # Maximum frame rate [Hz]
FramesToBeEncoded         2000        # Number of frames (at input
frame rate)
GOPSize                   16         # GOP Size (at maximum frame rate)
BaseLayerMode             2          # Base layer mode (0,1: AVC
compatible,
                                     #                    2: AVC w
subseq SEI)
SearchMode                4          # Search mode (0:BlockSearch,
4:FastSearch)
SearchRange               32         # Search range (Full Pel)
NumLayers                 1          # Number of layers
LayerCfg                  layer0.cfg # Layer configuration file
```

Figure 20: Main configuration file main.cfg for single layer coding

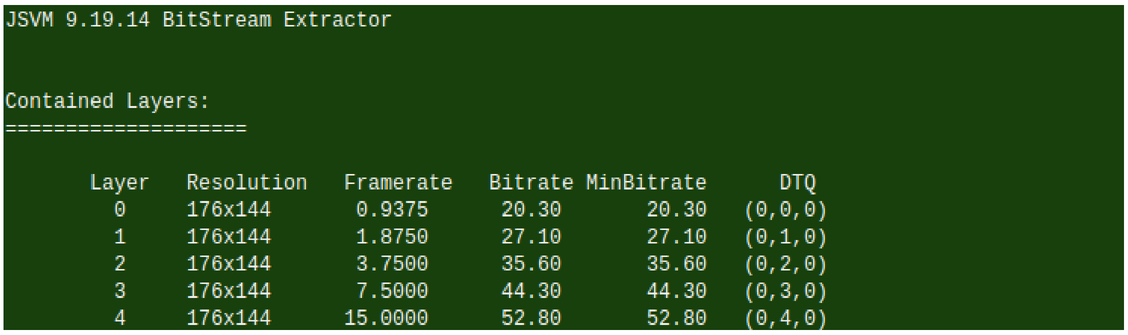
The most important parameters that need to be specified in the main configuration file are the name for the bit-stream Outputfile, the frame rate, the number of frames to be encoded, the GoP size, and the base layer mode. Furthermore the parameter NumLayers has to be set equal to 1 for single layer coding, and exactly one layer configuration file has to be specified via the parameter LayerCfg.

In the layer configuration file, the filename of the input sequence InputFile has to be specified, the frame width and height, and the frame rates of the input images. These parameters are set in the configuration file depicted below:

```
# JSVM Layer Configuration File

InputFile            foreman_qcif.yuv # Input  file
SourceWidth          176            # Input  frame width
SourceHeight         144            # Input  frame height
FrameRateIn          15             # Input  frame rate [Hz]
FrameRateOut         15             # Output frame rate [Hz]
```

Figure 21: Layer configuration file for single layer coding Foreman QCIF (176x144).

The aformentioned parameters were used as input for the experimental trials aiming to compare an AVC stream (no layer) stream PSNR values with its corresponding SVC (stream) encoded at the same bit-rate and containing 2(L0, L1) 3(L0, L1, L2), 4(L0, L1, L2, L3), 5(L0, L1, L2, L3, L4) temporal layers respectively. We selected our initial raw video file which contains 2000 frames on QCIF resolution (highway_qcif). Using the appropriate available binaries from JSVM we can create the corresponding SVC streams for each one of the 5 layers. According to parameters set on the configurations file depicted above the available layers are show below,in this output there are all summarized regarding the supported spatial, resolutions, frame rates, and bit-rates of the sub-streams. For our example only a single spatial resolution 176X144 samples is supported. However the bit-stream provides 4 different temporal resolutions with frame rates that are depicted on figure below:



```
JSVM 9.19.14 BitStream Extractor


Contained Layers:
====================

       Layer   Resolution   Framerate   Bitrate MinBitrate      DTQ
         0     176x144       0.9375      20.30       20.30   (0,0,0)
         1     176x144       1.8750      27.10       27.10   (0,1,0)
         2     176x144       3.7500      35.60       35.60   (0,2,0)
         3     176x144       7.5000      44.30       44.30   (0,3,0)
         4     176x144      15.0000      52.80       52.80   (0,4,0)
```

Figure 22: Temporal layers included on SVC stream.

It is assumed that the number of mobile device's users is standard set to 5, the loss probability regarding the 3G network is the same for all users and set to 0.35 .

On each experiment we compare the performance of two sub-streams on each case :

1. The SVC sub-stream that contains all or part of the 5 layers (Each substream is derived from the main svc stream produced after SVC encoding)
2. The corresponding no layer AVC stream on each case.

These two sub-streams are given as input to our simulator and after each simulation is performed a new “corrupted” bit-stream file is produced due to packet losses that take place inside the network. These “corrupted” bit-stream files are given as input to the ffmpeg decoder so we can get the final.yuv video files on each case. In the sections below you'll find the results from the experiments we've run on the simulator and distinguish between two simulation scenarios regarding loss probability for the transmissions that take place inside the local wireless lan scenario1 loss prob=0.1, scenario2 loss prob=0.2.
In all experiments we've performed we assume that the number of users is fixed on number 5.

9.2 Comparison of QCIF sequences

In this section we will present the result of different simulation scenarios we've run in order to compare the performance of AVC and SVC video sequences regarding the algorithm presented on previous section. The video sequence tested here is FOREMAN video on 15fps and 176x144 (QCIF) resolution.

9.2.1 Experiment 1 (2 Temporal Layers L0,L1)

Sub-streams given as input:

1. SVC sub-stream that contains the layers L0,L1 with frame rate 1.875Hz. The sub-stream is derived from the main svc stream

produced after SVC encoding. The initial raw .yuv video file given as input to the JSVM encoder had 2000 frames. Considering this the svcSubStream1 will contain $\frac{300*1.875}{15}=37$ frames.

2. The corresponding AVC stream with 37 frames.

a) Wlan loss probability 0.1

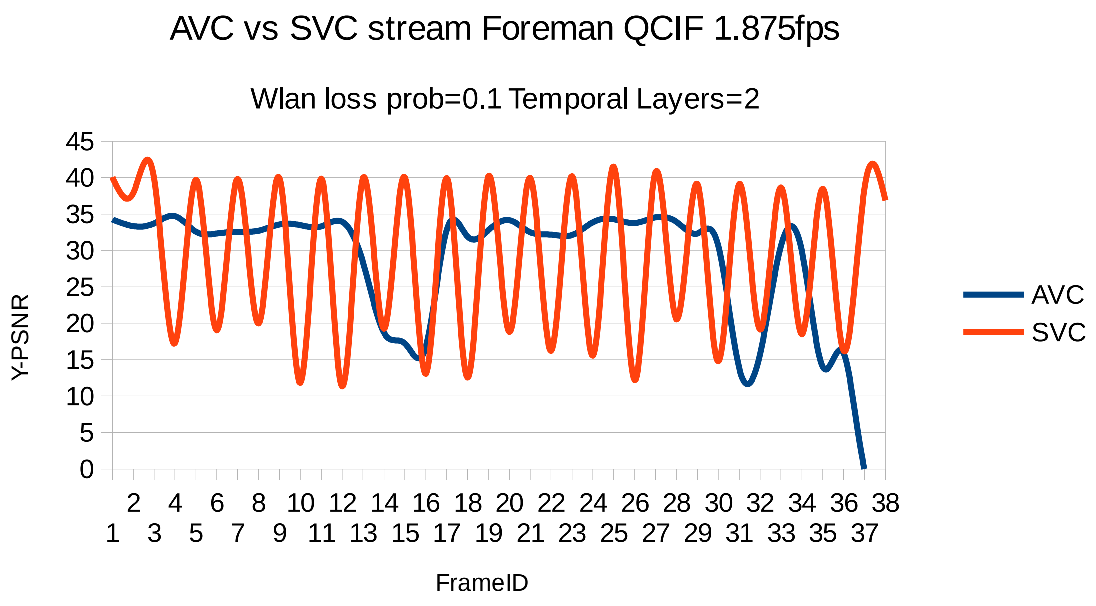


Figure 23: AVC vs SVC stream, QCIF 1.875fps ,wlan loss prob=0.1

b) Wlan loss probability 0.2

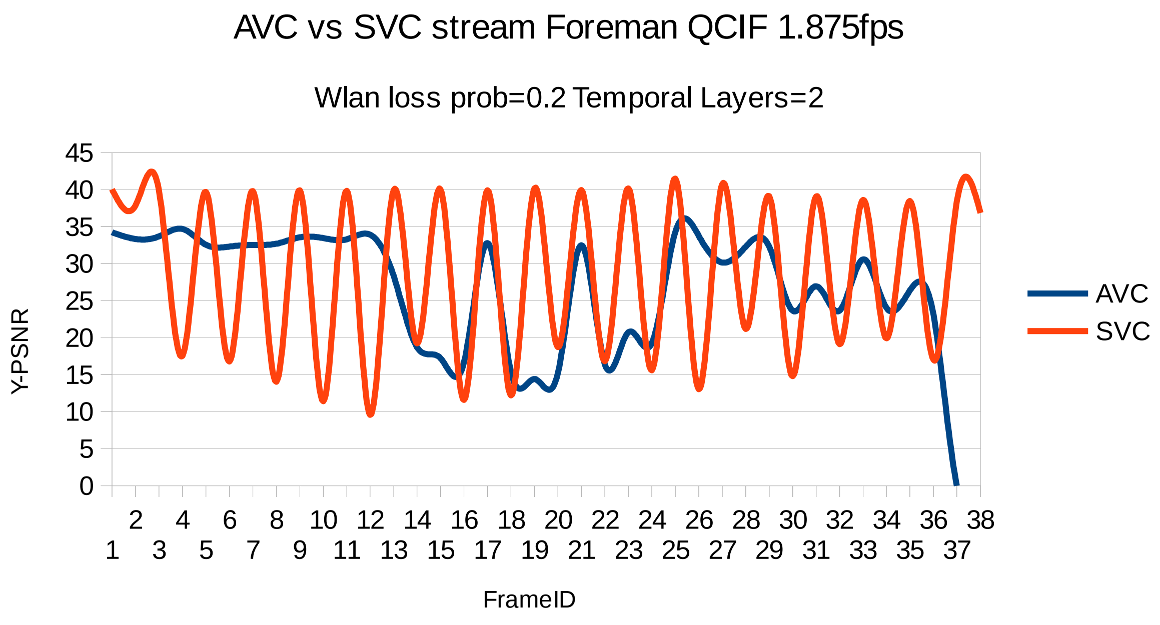


Figure 24: AVC vs SVC stream, QCIF 1.875fps ,wlan loss prob=0.2

9.2.2 Experiment 2 (3 Temporal Layers L0,L1,L2)

Sub-streams given as input:

- SVC sub-stream that contains the layers L0,L1,L2 with frame rate 3.75Hz. The svcSubStream2 will contain fr $\frac{300 * 3.75}{15} = 75$ frames.
- The corresponding AVC stream with 75 frames.

a) Wlan loss probability 0.1

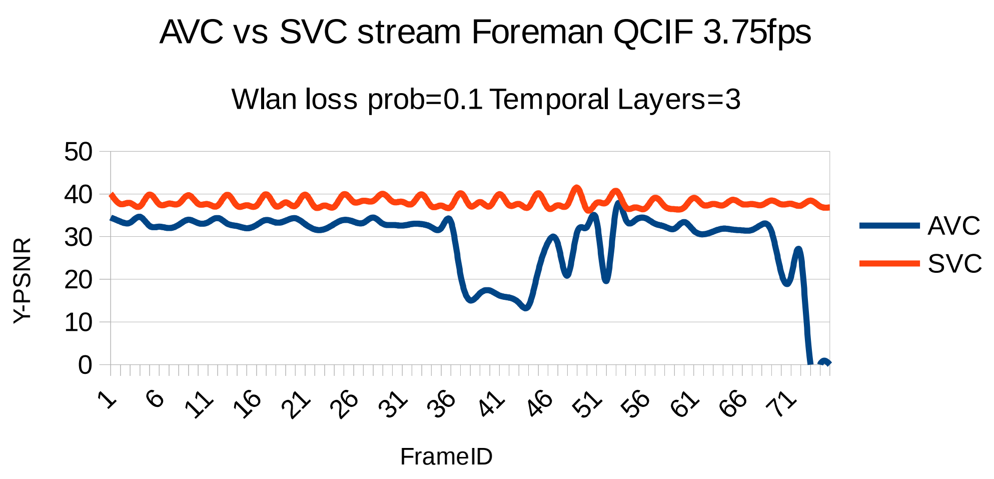


Figure 25: AVC vs SVC stream, QCIF 3.75fps ,wlan loss prob=0.1

b) Wlan loss probability 0.2

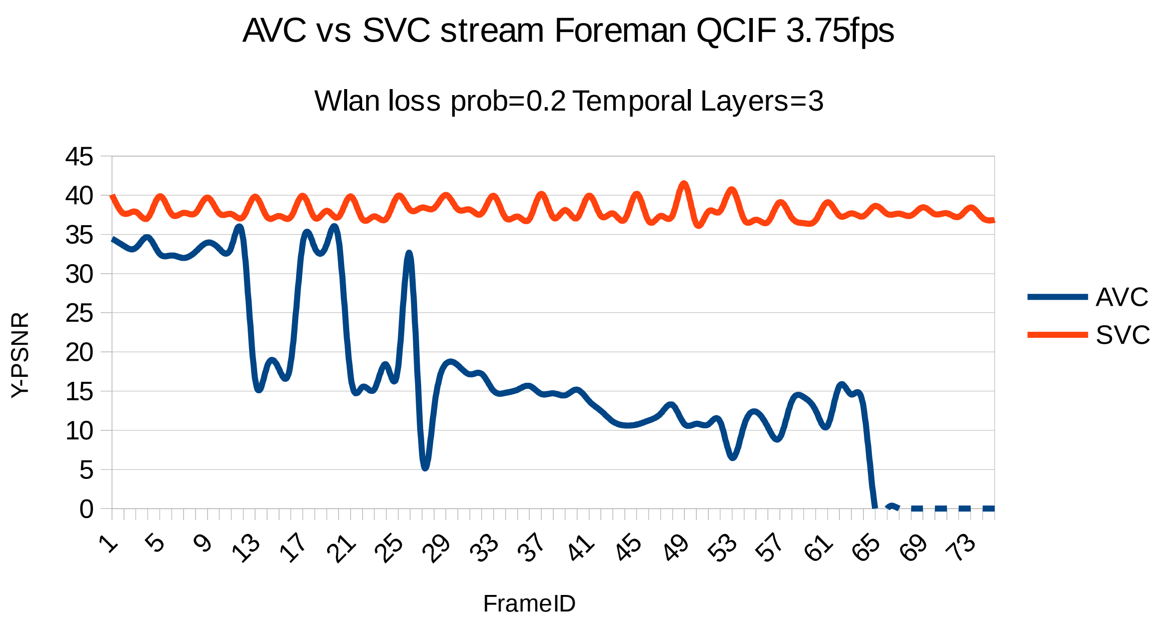


Figure 26: AVC vs SVC stream, QCIF 3.75fps ,wlan loss prob=0.2

9.2.3 Experiment 3 (4 Temporal Layers L0,L1,L2,L3)

Sub-streams given as input:

- SVC sub-stream that contains the layers L0,L1,L2,L3 with frame rate 15Hz. The svcSubStream3 will contain fr $\frac{300*7.5}{15}=150$ frames.
- The corresponding AVC stream with 150 frames.

a) Wlan loss probability 0.1

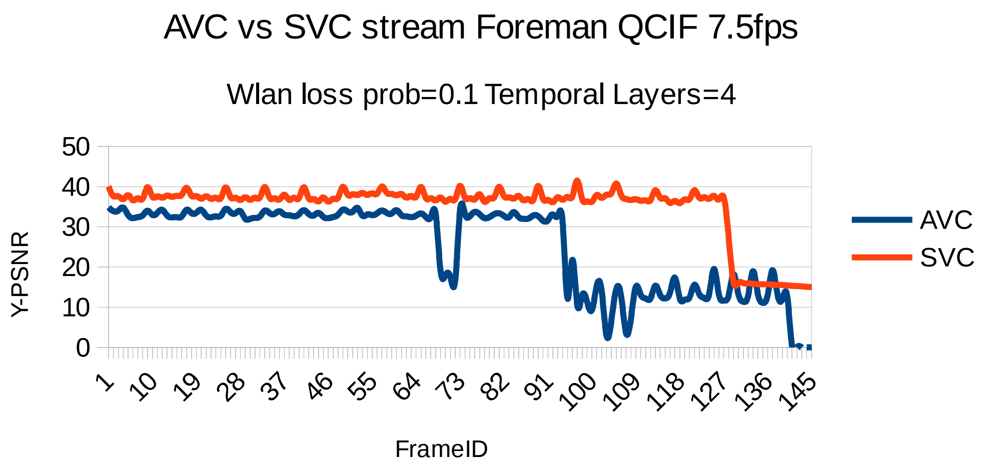


Figure 27: AVC vs SVC stream, QCIF 7.5fps ,wlan loss prob=0.1

b) Wlan loss probability 0.2

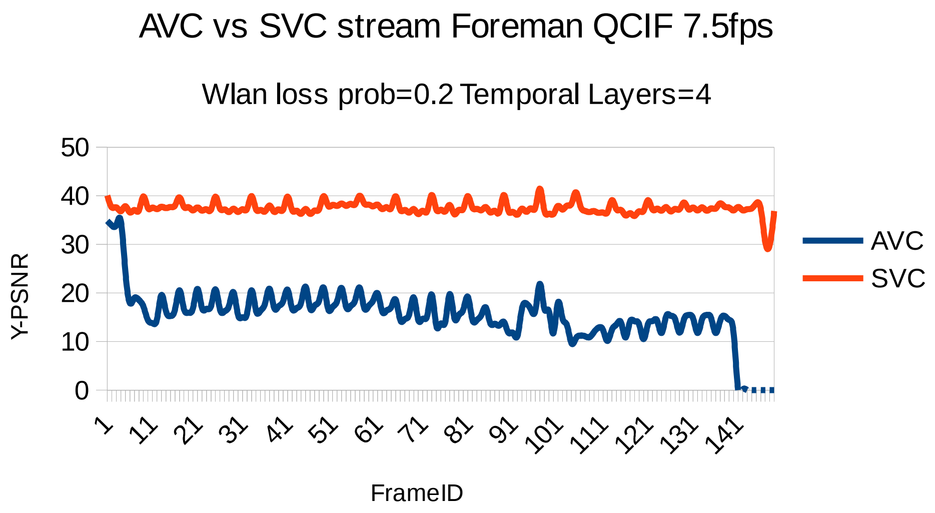


Figure 28: AVC vs SVC stream, QCIF 7.5fps ,wlan loss prob=0.2

9.2.4 Experiment 4 (5 Temporal Layers L0,L1,L2,L3,L4)

Sub-streams given as input:

1. SVC sub-stream that contains the layers L0,L1,L2,L3,L4 with frame rate 30Hz. The svcSubStream2 will contain 300 frames.

2. The corresponding AVC stream with 300 frames.

a) Wlan loss probability 0.1

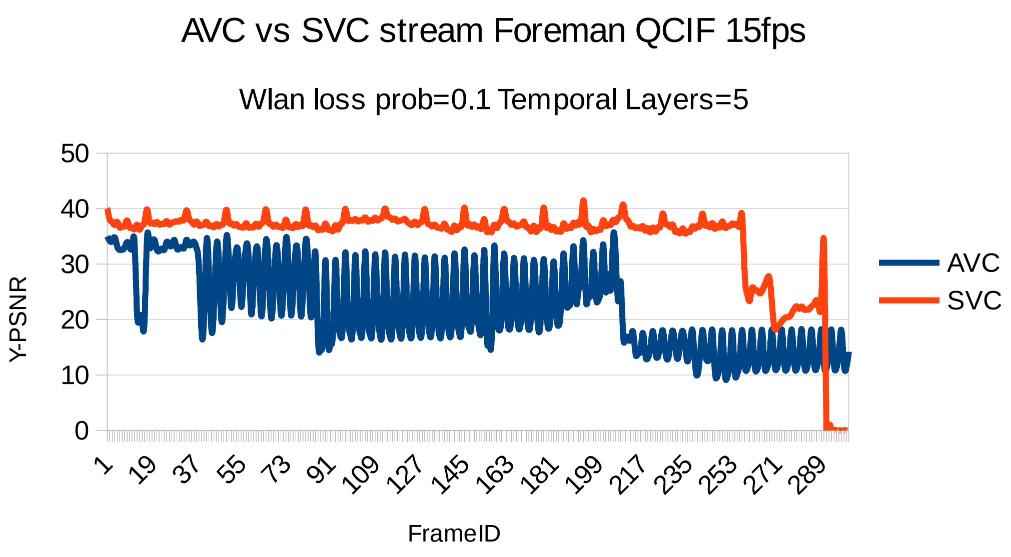


Figure 29: AVC vs SVC stream, QCIF 15fps ,wlan loss prob=0.1

b) Wlan loss probability 0.2

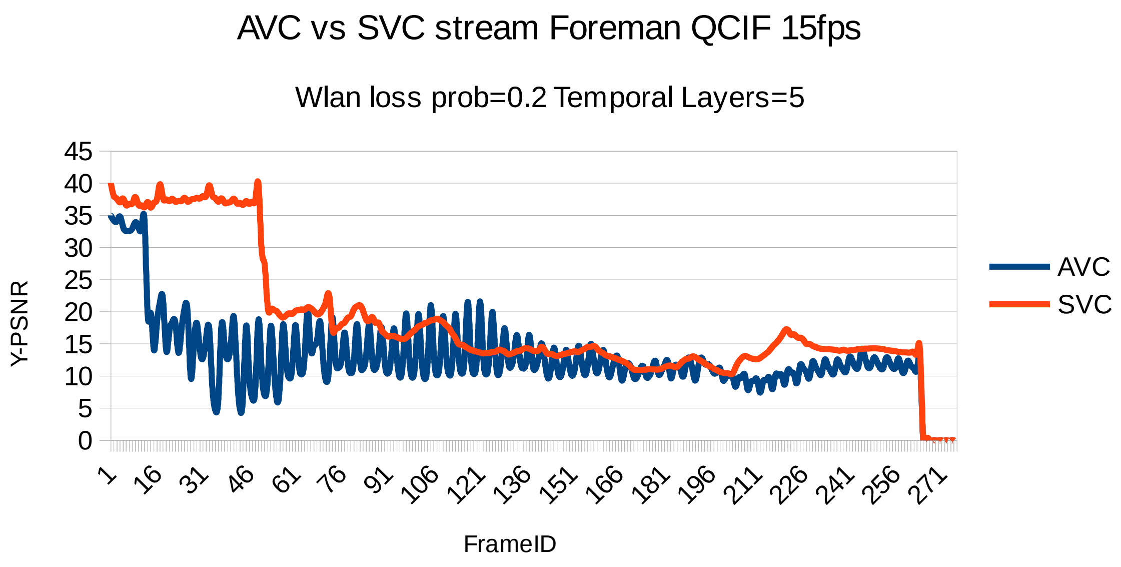


Figure 30: AVC vs SVC stream, QCIF 15fps ,wlan loss prob=0.2

9.3 Comparison of CIF sequences

In this section we will present the result of different simulation scenarios we've run in order to compare the performance of AVC and SVC video sequences regarding the algorithm presented on previous section. The video sequence tested here is FOREMAN video on 15fps and 352x288 (CIF) resolution.

9.3.1 Experiment 1 (2 Temporal Layers L0,L1)

Sub-streams given as input:

3. SVC sub-stream that contains the layers L0,L1 with frame rate 1.875Hz. The sub-stream is derived from the main svc stream produced after SVC encoding. The initial raw .yuv video file given as input to the JSVM encoder had 2000 frames. Considering this the svcSubStream1 will contain $\frac{300*1.875}{15}=37$ frames.

4. The corresponding AVC stream with 37 frames.

a) Wlan loss probability 0.1

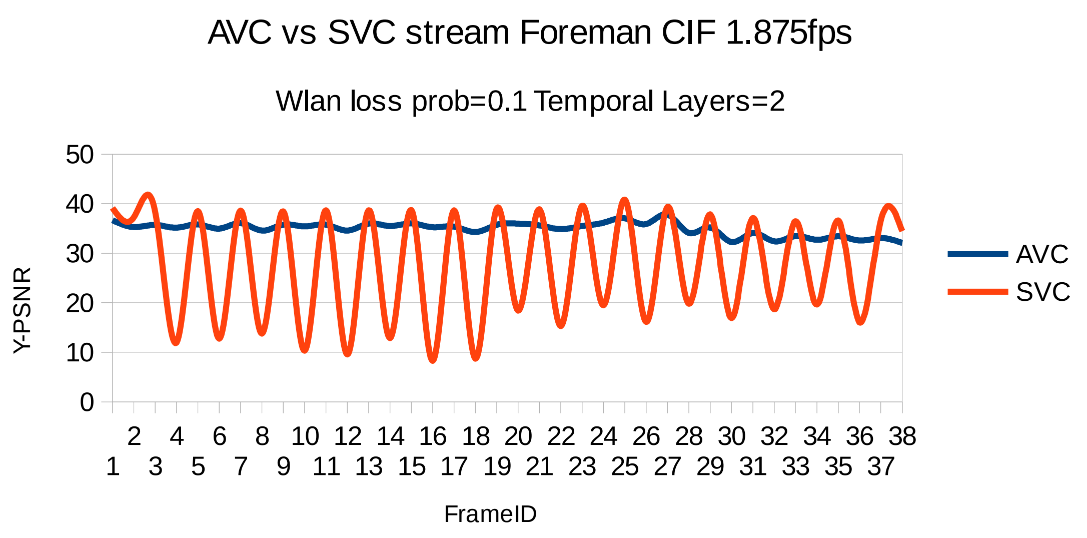


Figure 31: AVC vs SVC stream, CIF 1.875fps ,wlan loss prob=0.1

b) Wlan loss probability 0.2

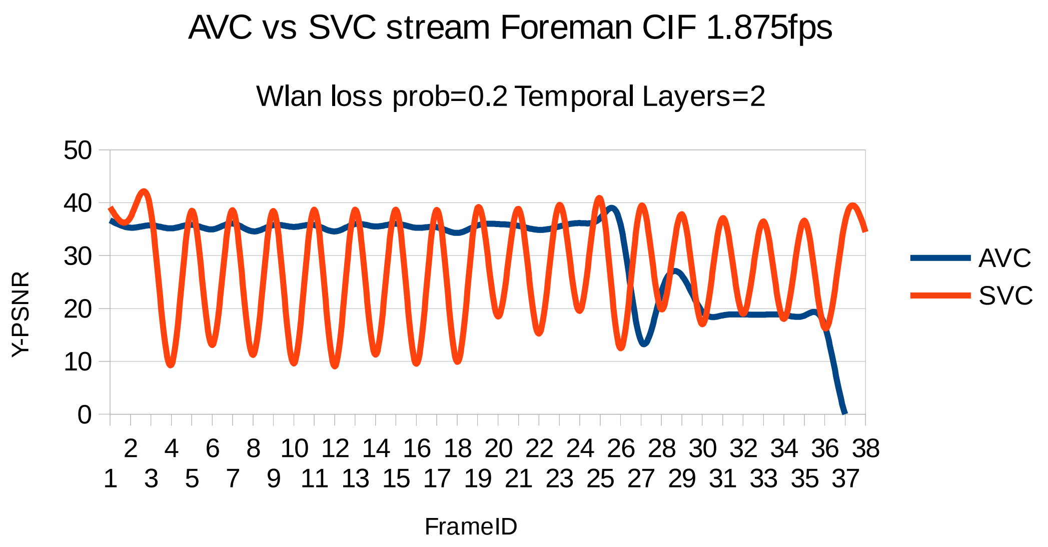


Figure 32: AVC vs SVC stream, CIF 1.875fps ,wlan loss prob=0.2

9.3.2 Experiment 2 (3 Temporal Layers L0,L1,L2)

Sub-streams given as input:

- SVC substream that contains the layers L0,L1,L2 with frame rate 3.75Hz. The svcSubStream2 will contain fr $\frac{300*3.75}{15}=75$ ames.
- The corresponding AVC stream with 75 frames.

a) Wlan loss probability 0.1

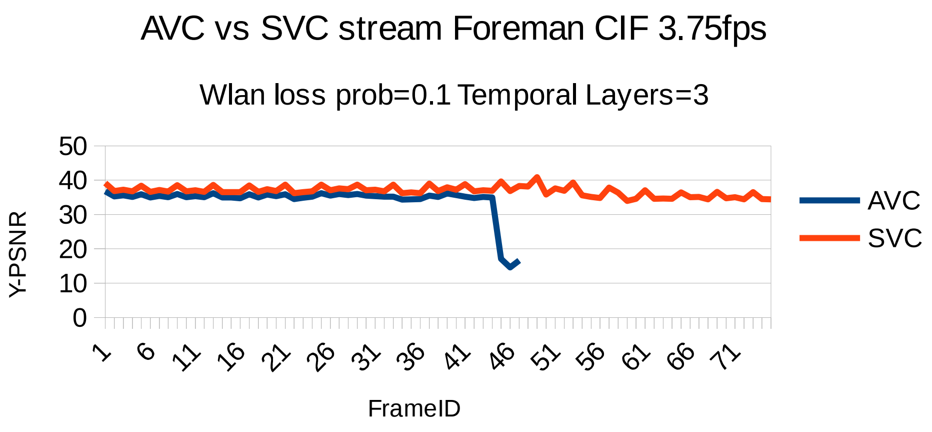


Figure 33: AVC vs SVC stream, QCIF 3.75fps ,wlan loss prob=0.1

b) Wlan loss probability 0.2

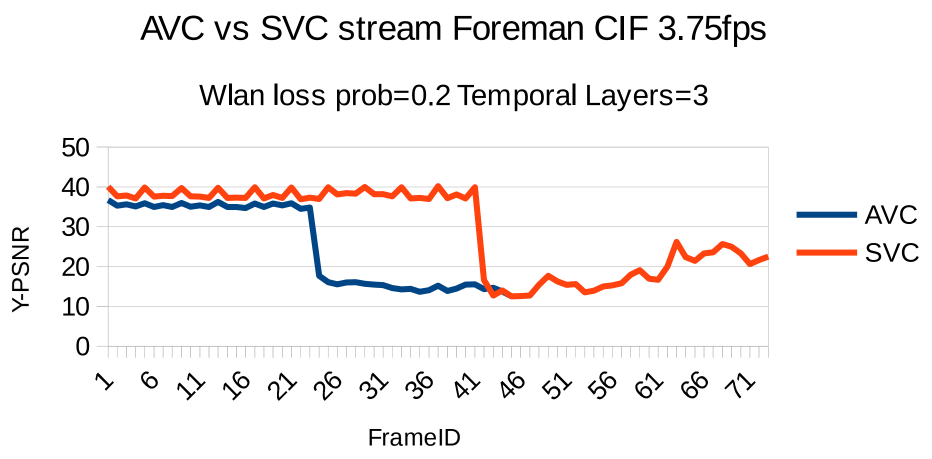


Figure 34: AVC vs SVC stream, CIF 3.75fps ,wlan loss prob=0.2

9.3.3 Experiment 3 (4 Temporal Layers L0,L1,L2,L3)

Sub-streams given as input:

- SVC sub-stream that contains the layers L0,L1,L2,L3 with frame rate 15Hz. The svcSubStream3 will contain fr $\frac{300*7.5}{15}=150$ frames.
- The corresponding AVC stream with 150 frames.

a) Wlan loss probability 0.1

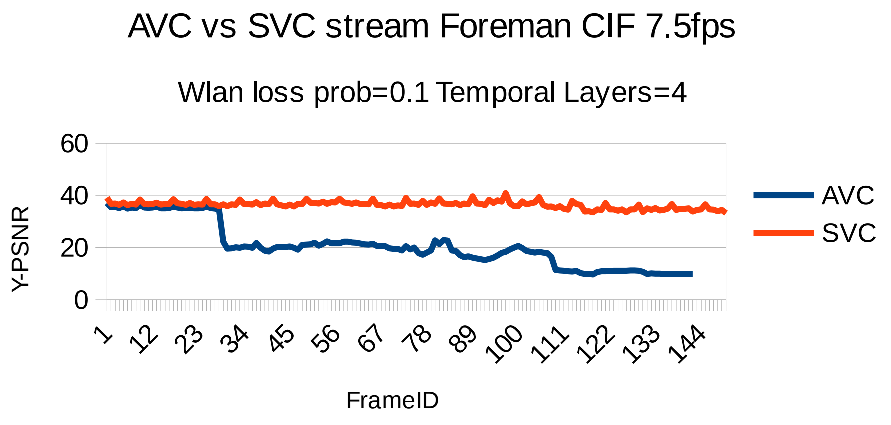


Figure 35: AVC vs SVC stream, CIF 7.5fps ,wlan loss prob=0.1

b) Wlan loss probability 0.2

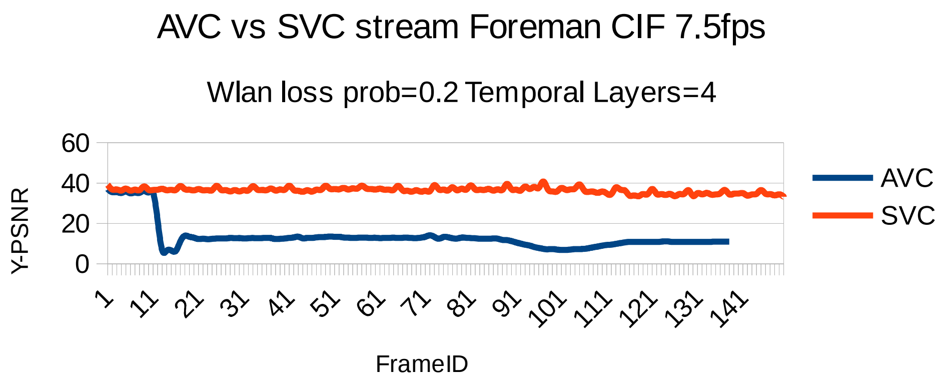


Figure 36: AVC vs SVC stream, CIF 7.5fps ,wlan loss prob=0.2

9.3.4 Experiment 4 (5 Temporal Layers L0,L1,L2,L3,L4)

Sub-streams given as input:

1. SVC sub-stream that contains the layers L0,L1,L2,L3,L4 with frame rate 30Hz. The svcSubStream2 will contain 300 frames.
2. The corresponding AVC stream with 300 frames.

a) Wlan loss probability 0.1

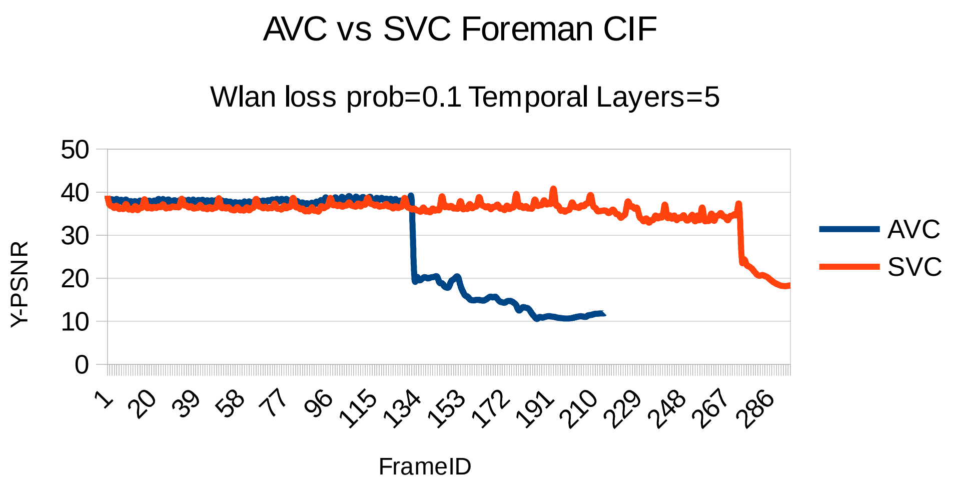


Figure 37: AVC vs SVC stream, CIF 15fps ,wlan loss prob=0.1

b) Wlan loss probability 0.2

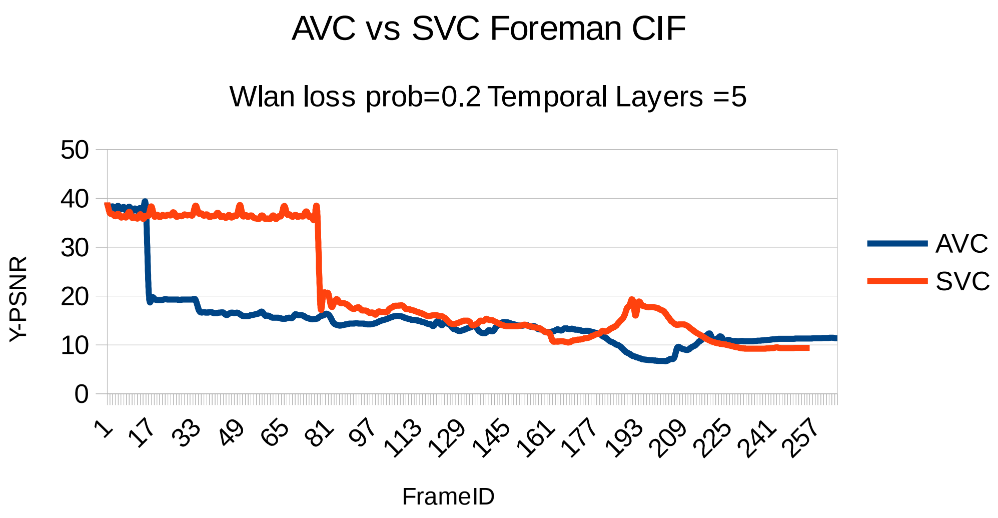


Figure 38: AVC vs SVC stream, CIF 15 ps ,wlan loss prob=0.2

10 Conclusions

The scope of this thesis was to investigate the performance of network coding for a SVC coded video transmitted in a group of users with proximity to each other in a mesh topology. According to the results taken from the experiments described on the section above we've concluded that, regarding temporal scalability, SVC streams can achieve better performance comparing to corresponding AVC substreams. The wireless probability doesn't affect the performance of AVC over SVC streams (SVC either on that case outperforms AVC), however the performance for both of the streams is decreasing while loss probability grows, as expected. Comparing the graphs obtained by the two streams reveals, that the early frames of the video sequence have similar quality for both of the two streams either using scalable stream of 2 or 3 or 4 or 5 layers. However, looking the frames that follow across the video sequence, SVC streams achieve better quality comparing to those of AVC streams. This can be attributed to the way each standard organizes its frame sequence. In AVC each video frame can be either an intra-coded (I), forward predictive coded (P) with motion compensated prediction from the preceding I frame or P frame, or bi-directionally predictive (B) based to its position in a group of pictures, (GoP) structure. With classical B frame prediction, a B frame is encoded with motion compensated prediction not only from the preceding but also from the succeeding I or P frame. Based on the aformentioned ,which is default in H.264/AVC, a B frame is not used as prediction reference for another B frame [12].

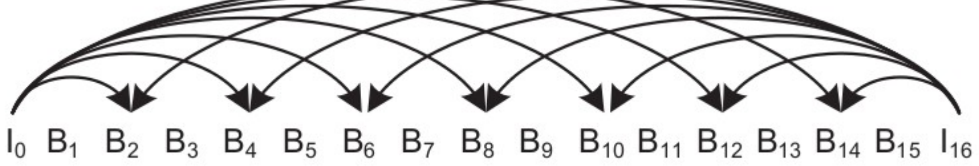

I_0 B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8 B_9 B_{10} B_{11} B_{12} B_{13} B_{14} B_{15} I_{16}

Figure 39: Classical B frame prediction used by default in H.264/AVC

On the contrary, in H.264/SVC standard, the restriction regarding the prediction of B frames has been lifted through its generalized B frame concept that permitted B frame blocks to be used as reference for the motion compensated prediction of blocks of other B frames. SVC employs the hierarchical B frame prediction.

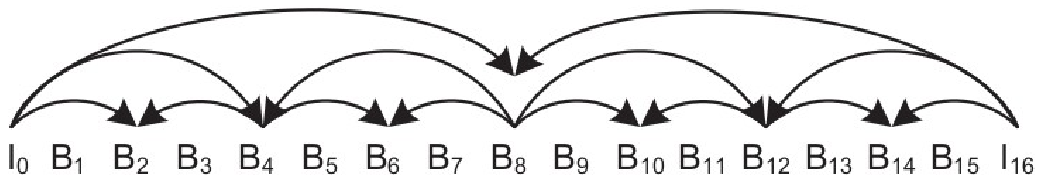

I_0 B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8 B_9 B_{10} B_{11} B_{12} B_{13} B_{14} B_{15} I_{16}

Figure 40: Hierarchical B frame prediction:default in H.264 SVC

Concluding, SVC bit-streams are less sensitive on losing a B frame, because of the layering that exists, when compared to AVC. Hence, the more layers are added on a SVC bit-stream sequence, the less critical is to lose a B frame and that's the reason why SVC bit-stream performance is better comparing to it's corresponding AVC. Moreover, it is concluded that the resolution size does not affect the performance of SVC over AVC (SVC outperforms over AVC). However in the cases where the SVC stream contains only 2 temporal layers, the performance is worse comparing to the AVC stream. In general, as the number of temporal layers increase, the SVC outperforms AVC stream for the reasons mentioned above.

11 Future Work

Scalable Video Coding is a new, promising video coding standard and becomes a highly attractive solution to the problems posed by the characteristics of modern video transmission systems. It can easily adapt to the various needs of preferences of end users as well as to the varying terminal capabilities or network conditions.

In our testing scenarios we tried to compare the performance of the transmission of a SVC stream, with multiple temporal layers, to its corresponding AVC stream, among a set of users with proximity to each other, using network coding. The parameters we've took under consideration were, the number of temporal layers of the SVC bit-stream and the loss probability regarding the transmissions inside the wireless local network. However it could be part of future job to check the performance of SVC while changing the GoP size which probably affects the performance or take into consideration the effect of other parameters like the number of users. An other thing that maybe interesting is to perform the same tests using as input spatial scalable SVC streams ,or SNR scalable SVC streams. Finally, regarding the network coding part ot the experiments, we can check the effect of the size of each segment. We can select the packets that belong to one generation (segment), based on the characteristics of each NAL-Unit packet (dependency_id, temporal_id, quality_id), instead of having standard sized segments without taking into account the characteristics of the packets that the segment consists of.

12 Literature

[1] R W Yeung S.-Y. R. Li and N. Cai. Linear network coding. IEEE Transactions on Information Theory, 49(2):371–381, February 2003.

[2] T. Ho, M. Medard R. Kotter, J. Shi M. Effros, and D. Karger. A random linear network coding approach to multicast. IEEE Transactions on Information Theory, 52:4413–4430, October 2006.

[3] Philip A. Chou and Yunnan Wu Network Coding for the Internet and Wireless Networks Microsoft Research Banff International Research Station June 2007.

[4] Y. Wu P. A. Chou and K. Jain. Practical network coding. In In Proceedings of 51st Allerton Conference on Communication, Control and Computing, September 2003.

[5] L. Keller, A. Le, B. Cici, H. Seferoglu, C. Fragouli, A. Markopoulou, “Microcast: Cooperative Video Streaming on Smartphones in ITW (Sept 2012), ACM MobiSys (June 2012), ACM HotMobile (Feb. 2012), ITA Workshop (Feb. 2012).

[6] C. Fragouli and E. Soljanin. Network Coding Fundamentals. Now Publishers Inc, Delft, The Netherlands, June 2007.

[7] https://www.dropbox.com/s/1lxc5d35qecwd2s/JSVM_CVS_2011-08.zip.

[8] Cisco Visual Networking Index: Forecast and Methodology, 2014–2019. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf

[9] A Tutorial on H.264/SVC Scalable Video Coding and its Tradeoff between Quality, Coding Efficiency and Performance | InTechOpen, Published on: 2011-06-24.

[10] Performance Analysis of SVC Mathias Wien, Member, IEEE, Heiko Schwarz, and Tobias Oelbaum.

[11] Overview of the Scalable Video Coding Extension of the H.264/AVC Standard Heiko Schwarz, Detlev Marpe, Member, IEEE, and Thomas Wiegand, Member, IEEE.

[12] Overview of the H.264/AVC Video Coding Standard Thomas Wiegand, Gary J. Sullivan, Senior Member, IEEE, Gisle Bjøntegaard, and Ajay Luthra, Senior Member, IEEE. IEEE transactions on circuits and systems for video technology, vol. 13, no. 7, July 2003.

[13] Simulator source code http://vapashos.github.io/SVCNetworkCodingSimulator/