Scalable Video Streaming with Prioritised Network Coding on End-System Overlays

Michele Sanna

School of Electronic Engineering and Computer Science

Queen Mary, University of London

This dissertation is submitted for the degree of

Doctor of Philosophy

Supervisor: Prof. Ebroul Izquierdo

March 2014

Acknowledgements

First and foremost I'd like to thank my supervisor Prof. Ebroul Izquierdo, for the constant support in the roller-coaster ride which has been writing this thesis, for giving me the chance and endowing me with the confidence to explore new ideas, and exploit my own skills.

My warmest thanks go as well to the Reverie FP7 project partner, for without the extremely valuable confrontation and discussions this thesis would have not been the same.

Thanks to all my lab mates who have alternated during the years, and that have made the work place a fun place to endure the journey towards the PhD.

Last but not least, I would like to dedicate this to my family – my dad, my mum, and my younger brother – for always supporting and loving me, although without begin able to see me as often as I would have wanted, to my far but always close friends because without them all this would have been a journey to nowhere, and especially thank You, who believed in me so much that I couldn't help but having to top that by believing in myself too.

Abstract

Distribution over the internet is destined to become a standard approach for live broadcasting of TV or events of nation-wide interest. The demand for high-quality live video with personal requirements is destined to grow exponentially over the next few years. End-system multicast is a desirable option for relieving the content server from bandwidth bottlenecks and computational load by allowing decentralised allocation of resources to the users and distributed service management. Network coding provides innovative solutions for a multitude of issues related to multi-user content distribution, such as the coupon-collection problem, allocation and scheduling procedure. This thesis tackles the problem of streaming scalable video on end-system multicast overlays with prioritised push-based streaming. We analyse the characteristic arising from a random coding process as a linear channel operator, and present a novel error detection and correction system for error-resilient decoding, providing one of the first practical frameworks for Joint Source-Channel-Network coding. Our system outperforms both network error correction and traditional FEC coding when performed separately. We then present a content distribution system based on end-system multicast. Our data exchange protocol makes use of network coding as a way to collaboratively deliver data to several peers. Prioritised streaming is performed by means of hierarchical network coding and a dynamic chunk selection for optimised rate allocation based on goodput statistics at application layer. We prove, by simulated experiments, the efficient allocation of resources for adaptive video delivery. Finally we describe the implementation of our coding system. We highlighting the use rateless coding properties, discuss the application in collaborative and distributed coding systems, and provide an optimised implementation of the decoding algorithm with advanced CPU instructions. We analyse computational load and packet loss protection via lab tests and simulations, complementing the overall analysis of the video streaming system in all its components.

Contents

List of Figures

Nomenclature

Matrix Operators

A	Source Scatter Matrix
A_I	Source Identity-Scatter Matrix
B	Destination Scatter Matrix
G	Block code generation matrix
K	Adjacency Matrix
M	Network Transfer Matrix

Data Delimiters

δ	Network code redundancy
E_l	Estimation interval length
h	Maxflow / n. received packets
k	information block length for FEC coding
w	Packet length
λ	Number of packets in a generation partition or priority class
n	coded block length for FEC coding
$N_{inn,l}$	N. innovative received packets per data segment/layer
$N_{ni,l}$	N. non-innovative received packets
$N_{recv,l}$	N. received packets per data segment/layer

ω Source code size / number of packets in a generation

P_g Playback point

P_l Priority region length

Performance indicator

η_{cod} coding efficiency

η_{gp} goodput efficiency

μ_{skip} skip rate

Other Symbols

I(m,n) Pixel luminance value

L Number of priority classes

m Bit length of GF field elements

q Galois Field size

r_{FEC} Block Coding rate

T Sink nodes set

Superscripts

(g) Generation index

Subscripts

l Priority class of data segment

t Sink node

Variables

b Source Data Block (row vector)

d Encoded Data Block (row vector)

x Source Network Codeword (column vector)

y Received Network Codeword (column vector)

z Network Error Vector

Acronyms / Abbreviations

3GPP 3rd Generation Partnership Project

4CIF Quadruple CIF, spatial resolution 704×576

ALM Application Layer Multicast

AVC Advanced Video Coding

CABAC Context-based Adaptive Binary Arithmetic Coding

CAVLC Context-based Adaptive Variable Length Coding

CDN Content Delivery Network

CGS Coarse-Grain Scalability

CIF Common Intermediate Format, spatial resolution 352×288

CPU Central Processing Unit

DASH Dynamic Adaptive Streaming over HTTP

DCT Discrete Cosine Transform

DVB Digital Video Broadcasting

ESM End-System Multicast

FEC Forward Error Correction

FGS Fine-Grain Scalability

GF Galois Field

GOP Group of Pictures

GPU Graphics Processing Unit

HD High Definition

HEVC High Efficiency Video Coding

ICN Information-Centric Networks

IETF Internet Engineering Task Force

IP Internet Protocol

ISO International Organization for Standardization

ISP Internet Service Provider

LC Layered Coding

LCM Linear Code Multicast

LOC Linear Operator Channel

MANE Media-Aware Network Element

MDC Multiple Description Coding

MDS Minimum Distance Separable

MGS Medium-Grain Scalability

MPEG Moving Picture Experts Group

MSE Mean Squared Error

NAL Network Abstraction Layer

NC Network Coding

NEC Network Error Correction

NEWT Network Emulator for Windows Toolkit

OSI Open System Interconnection

P2P Peer-to-Peer

PSNR Peak Signal-to-Noise Ratio

QCIF Quarter CIF, spatial resolution 176×144

QoE Quality of Experience

QoS Quality of Service

RLF Random Linear Fountain

RTP Real-time Transport Protocol

SIMD Single Instruction Multiple Data

SOTA State Of The Art

SSE Streaming SIMD Extensions

SVC Scalable Video Coding

UEP Unequal Error Protection

ULP Unequal Loss Protection

URT Unequal Recovery Time

VCL Video Coding Layer

Chapter 1

Introduction

Recent advances in packet video streaming techniques and the succesful introduction of internet connectivity into every house and every device of daily use (TV sets, laptops, tablets, handheld devices) has accelerated the demand for higher quality personal video services. In 2012, two important milestones have been laid by arguably the biggest internet-based TV and live on-demand video platform. First, with the live streaming of the London Olympic games through NBC, YouTube LLC brought the live streaming of all sport events via the Google servers throughout the United States, reaching 225 million streams provided globally and with peaks of 500 thousand concurrent connections [5]. Later the same year, the memorable dive of Felix Baumgartner from outer atmosphere, was transmitted live to the whole planet via the YouTube portal, reaching a peak of 8 million concurrent connections at the instant of the jump [5]. Such high numbers are indicative of an exponentially growing trend, which is likely to soon stroll up beside traditional terrestrial and satellite TV services.

In a globalised scenario of video services, several target devices and services fall into the same scope of interest of one video provider. Alongside DSL home access with full-HD and 4K screens, mobile and nomadic users may request the same content via 3G and 4G, WiMax or WiFi, with computational and screen capabilities limited by the portable device. Dynamic Adaptive Streaming over HTTP (DASH) is the open source and research standard for firewall-friendly adaptive streaming, competitor of HTTP Live Streaming (HLS, Apple), HTTP Dynamic Streaming (HDS, Adobe Systems), and Smooth Streaming (Microsoft). DASH uses the existing infrastructure for content retrieval over HTTP, and streams stored video encoded at multiple rates via single-rate encoders to users, depending on the service subscription. Scalable Video Coding (SVC) on the other hand is a seamless multi-rate video coding option where reusability of the stream is embodied by the layered coding approach, and aims at providing better and more dynamic adaptability to several streaming conditions

as well as differentiated classes of service. A scalable encoded video has multiple decoding options embedded in the stream. Any scalable stream can be reduced to a lower resolution, frame rate or SNR-quality by removing parts of the stream, or augmented with additional enhancement data to increase the resolution, frame rate or SNR-quality starting from the already received stream.

Flash-Crowd effects resulting from sudden increase of content popularity are cause of overload of streaming server. To avoid the high number of connections to a single server and due to the difficulty in deploying multicast at the IP layer in a dynamic environment, providers employ a Content Delivery Network (CDN) with cache nodes which aggregate the users of a geographical region to a local access point. Deploying a CDN infrastructure involves replicating computing facilities at remote locations, hiring bandwidth from local Internet Service Providers (ISP), and implementing a content distribution system on the private backbone. Motivation for a new form of delivery called *peer-assisted* delivery is the realisation that during flash-crowd events the same content is available not only at the caching nodes, but also across the consuming users. End-users have at the same time the data cached locally, and possess idle resources that can be employed to redeliver it to other users. The idea that millions of users, consuming the same content at the same time, could provide a "pool" of servers in huge numbers to connect to, not only is promising and extremely powerful, but could be implemented with very little effort with all existing technology on nowadays Internet.

Pushing the content delivery duties and upstream burden to end-users might be the solution to fulfil an ever growing demand with a non-exponential increase of resources, both at large scale such as the YouTube case as well as medium and small case, e. g., national or small independent broadcasters. Content distribution may be realised via collaborative streaming from multiple sources, without any need to redesign the existing internet infrastructure or expand the serving facilities. End-system multicast might also take small scale personal and commercial communication to a whole new level, where people or companies can share or stream live material without the support of dedicated infrastructure or external services. Peer-to-Peer (P2P) file sharing protocols, although under an infamous reputation, have demonstrated the power of file sharing via mutual exchange among final users. Building upon the reliable BitTorrent protocol [6], Tribler was one of the most succesful systems for video sharing [7, 8] and is still nowadays growing the customers span. In the industry world, a french company and the french national television have also teamed up to provide a browser-based peer-assisted solution for broadcasting the football World Cup 2014 [9]. Finally, the European project P2P-Next [10], which Tribler is also part of, put under de-

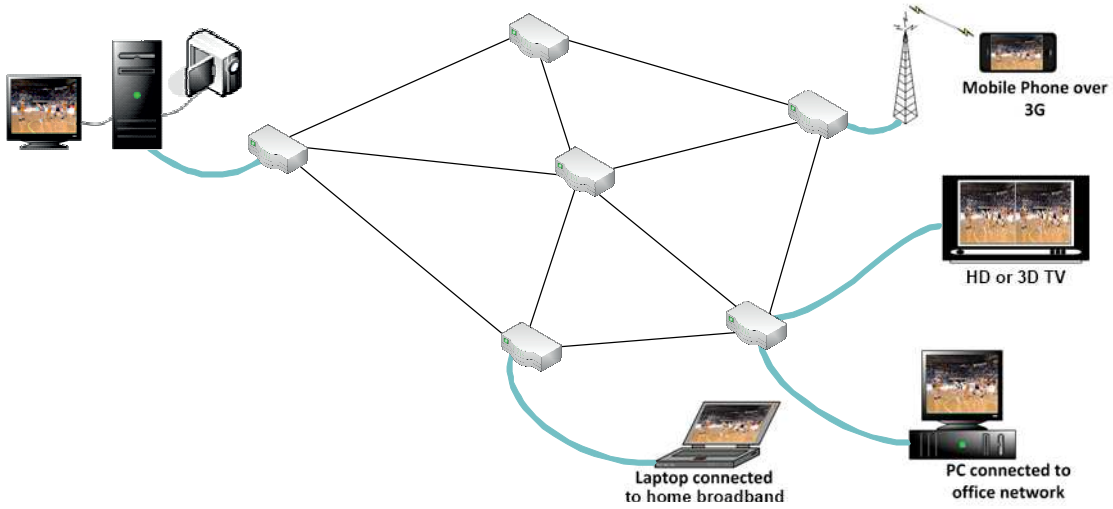

Fig. 1.1: An SVC application scenario with stream adaption for differentiated services.

velopement an internet television standard based on P2P, which marks a high and ethical recognition of P2P technologies for future multimedia communication.

In this thesis we analyse a content distribution system of novel conception, entailing coded collaborative multicast of scalable video over an end-systems mesh overlay. Prioritised network coding will be introduced as a fundamental tool for distributed coding and for optimised chunk delivery. We'll firstly highlight the motivations for our system design choices, then illustrate in detail our contributions.

1.1 Motivation

Our work is driven by the need for better personal video streaming services, available everywhere and for all kind of devices and satisfying a broad range of user needs in terms of Quality of Service.

Scalable Video Streaming

Traditional video codecs are designed to encode for a specific target bit rate and visual configuration. The variety of receivers in interconnected networks requires flexibility in terms of coding and delivery of the data. Scalable Video Coding (SVC) is a video coding paradigm that exploits the variety of the transmission channels and users requirements. SVC allows partial decoding of the video at reduced resolution, frame rate or quality from the same

embedded bit-stream. Several alternatives exist, the standard H.264/Scalable Video Coding (SVC) being based on block Discrete Cosine Transform spatial coding, and alternatives based on spatial and temporal wavelet transforms. Multirate delivery such as in the scenario in Fig. 1.1 would allow mobile devices with low computational power and small displays to receive a compact stream with reduced quality and resolution, whereas home users and companies connected via optic fibre and with availability of big screens can receive a higher quality video without the need for the cache nodes to store different versions of the video. In datagram-based and feedback free channels, with multiple paths and that are prone to losses and errors, or in low delay applications, channel coding plays a critical role, and SVC has been employed for superior resilience via joint source-channel coding. Unequal Error and Loss Protection (UEP/ULP) are coding techniques tailored for layered coding data, where the amount of redundant data for protection is tuned to minimise the visual distortion, based on the importance of the data layers. Multiple Description Coding (MDC) and Prioritised Encoding Transmission (PET) on the other hand aim at providing the best decoding compromises with partially received data, by distributing data in several representations in a way that allows partially decoding te video from a subset of received packets.

Content Delivery and Peer-to-Peer Networks

The situation in Fig. 1.1 illustrates a scalable video distribution system based on a CDN infrastructure. The concern that multicast at IP layer would not provide the necessary support in terms of network scalability and support for error, flow, and congestion control lead to the exploration of Application Layer Multicast (ALM) options. Infrastructure with application layer functionalities such as caching and multicast is put in place to replicate data over wide backbone links while providing a geographically localised access point. A special case of application layer multicast, referred to as End-System Multicast (ESM), is when all ALM nodes are End-Systems or final users and potentially recipient of the transmitted information. In a relatively not so recent work it was argued that End-System multicast would provide the solution to all shortcomings of IP multicast and additionally provide all desired properties in a CDN such as: Self-organisation, overlay efficiency, self-improvement, and adaptation to network dynamics [11]. Fig. 1.2 shows the difference in employing the IP layer functionalities to deliver data to multiple receivers, with multiple unicast sessions (a) or native multicast (b), or node communication at application layer, providing ALM (c) or End-System Multicast (d).

Multiple platform and types of devices are nowadays connected to the internet, putting

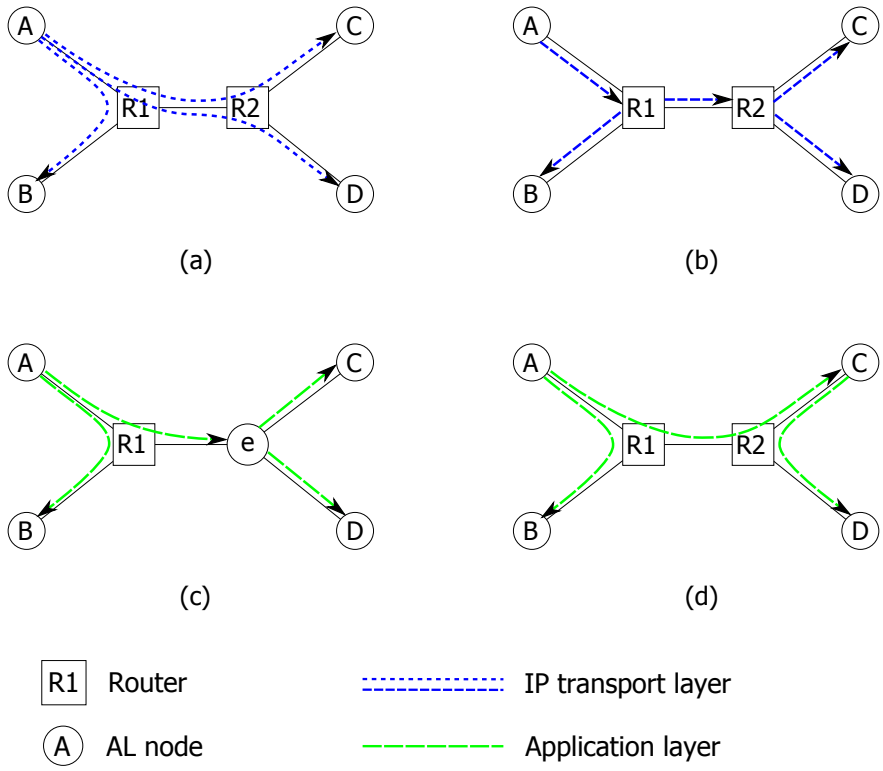

Fig. 1.2: Examples of multicast options: naive IP unicast (a), IP multicast (b), application layer multicast (c), and end-system multicast (d).

deeply different display requirements, reception conditions and Quality of Service (QoS) demands behind the same basic internet service offered by ISP. Peer-to-Peer (P2P) technologies, working without a serviced infrastructure and relying only on the spontaneous aggregation of users, have been successfully employed for live and on demand multimedia delivery, thanks to the inter-user communication that allows a closer interaction and an impromptu allocation of resources to the users needs. Scalable video delivery over hybrid network of peers with a content streaming server, known as peer-assisted delivery, will be topic of discussion of this thesis.

Network Coding

Network communication as a whole, but also specifically P2P systems as we'll shortly explain, are going through a stage of renovation with the introduction of Network Coding (NC). NC was pin pointed into research literature with the seminal article by Ahlswede *et al.* in [12] where the possibility of encoding information at the intermediate network nodes, as opposed to traditional relaying, is envisioned as the ultimate way to achieve the theorised

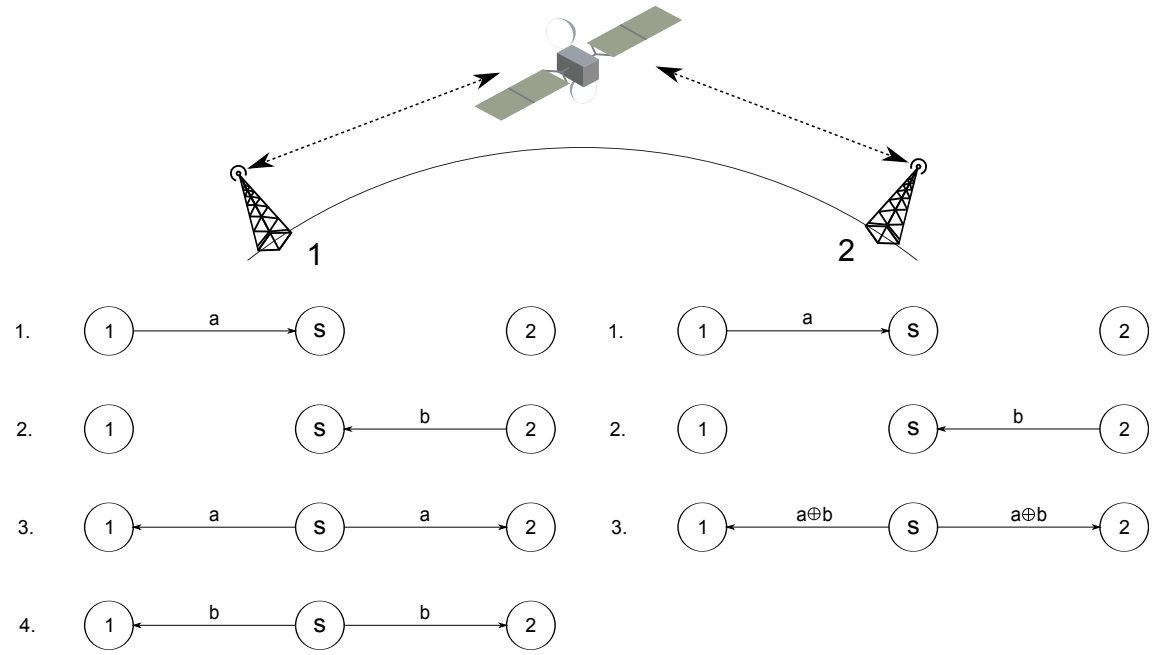

Fig. 1.3: The satellite example: with traditional relay (left) and with coding (right).

network capacity, achievable with routing only in particular cases.

Network coding was initially studied in the scope of satellite communication, where the idea of broadcasting encoded information rather than transmitting the exact signal needed by one receiver would increase the system performance at virtually no price, as shown in Fig. 1.3. A network equivalent of the satellite example, where a unique source is added, the satellite bottleneck has been made explicit, and both ends have been made recipient of the data, is the butterfly network shown in Fig. 1.4. In this classic example, one can note how only one packet among a and b can be forwarded via the middle link, allowing only one of the two sinks receiving both data symbols at a time (Fig. 1.4,a), whereas network coding would allow both sink nodes to receive an encoded version of the data via the XOR in the middle link. This allows decoding the information at both receivers with less bandwidth usage and with less delay (Fig. 1.4,b). Network Coding has proven to bring a number of advantages, such as:

1. Maximising the throughput and reach the network capacity.
2. Minimising the latency.

Network coding has introduced a new field of coding theory, one of the most interesting results being the Network Error Correction (NEC) topic, which studies the network coding characteristic to allow forward error correction at NC level.

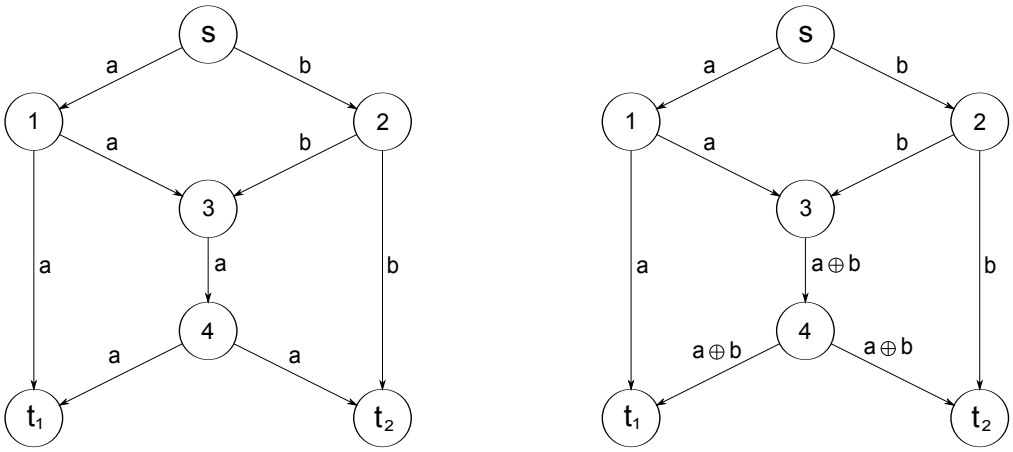

Fig. 1.4: The two receivers multicast butterfly network example with traditional routing (a) and with network coding (b).

Although influential authors argue that P2P does not perform network coding the way it has been theorised, several advantages are to be found when employed in P2P networks in the way data chunks are exchanged, both in file download and media streaming. Elegant solutions to the well-known *coupon collection* and the *missing chunk* problems can be implemented. Fig. 1.5 shows how the approach to data distribution changes with the introduction of network coding and the use of scalable data. In traditional P2P every data piece is unique. Every user identifies portions of data as either received or not received. Packet scheduling, chunks requests, prioritisation policies for "rare" pieces are necessary to make sure that each individual data chunk is delivered to each user (Fig. 1.5,a). On the other hand, with network coding, data bits are not a commodity anymore. Users can exchange encoded version of the buffered chunks, and, by these simple means, deliver a useful piece of information that builds up to succesful decoding of the information, with virtually no inefficiencies (or very little, e. g. in randomised settings), but sensibly reducing the management and coordination procedures.

In this work we aim at designing components and strategies for scalable video streaming systems via End-system multicast, possibly with P2P logic. We analyse advantages and implementation possibilities of network coding, implementing priority encoding techniques, a scalable data exchange method for P2P networks, and an optimised implementation of data recovery from fountains of encoded data.

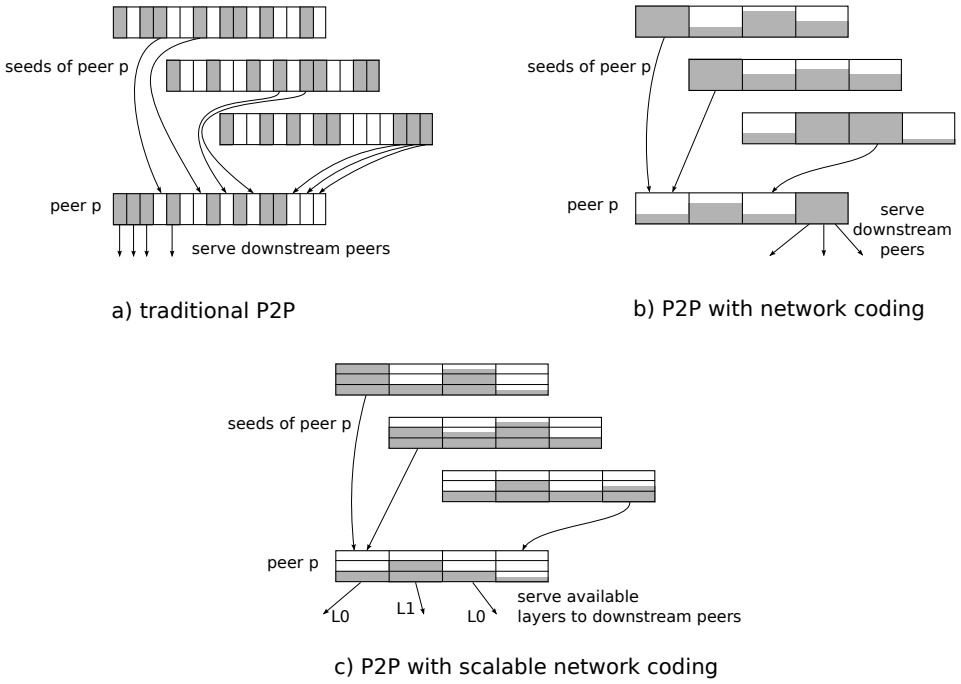

Fig. 1.5: Comparison between traditional pull-based P2P protocols (a) and network coding, push-based P2P systems in traditional (b) and scalable (c) implementation.

1.2 Contribution and Thesis Organisation

This thesis is structured as follows: We firstly give on overview of network coding, both algebraic and applied to data relay problems, and scalable data transmission. We then proceed to describe the main achievements of this project through three main contributions.

State of the art

Since network coding is the aspect that has the most influence in the design of the data flow and coding, Chapter 2 will review the basic formulation of both the algebraic approach to and the practical framework of network coding, as well as chunk exchange scheduling techniques on P2P. This will allow us to then introduce the decoding via detection and deletion method in Chapter 4 as well as the overlay data delivery in Chapter 5. To complement this necessary survey, Appendices B and C will review the coding bounds and network weights from network error correction theory, and deterministic and randomised construction techniques for standard and error correcting network codes, respectively. As far as scalable data coding and transmission is concerned, Chapter 3 will introduce scalable video coding and channel coding techniques for prioritised transmission. We clarify firstly how a scalable stream is formatted and is to be decoded, and then we describe the current state of the art in rateless unequal loss and error protection, as well as multiple description coding techniques.

By introducing multiple description coding we lay the foundation to describe our first contribution of prioritised coding over a network coding channel in Chapter 4, whereas describing the fountain codes and variants with unequal protection is again functional to introduce our data streaming system between overlay nodes (Chapter 5), and to explain design choices, similarities and differences with our fountain coding transmission in Chapter 6. Appendix A will then briefly describe the video coding architecture of SVC used in our experiments.

Contributions

The contribution of this thesis can divided into three main points.

Chapter 4 presents a method for coding data against channel errors, combining the error detection aspects of network error correction and the prioritised encoding transmission framework applied to scalable data, providing to the authors' best knowledge one of the first practical implementations of Joint Source-Channel-Network Coding. The algebraic formulation will help us at this first stage, and will be used to study the transmission over the network channel as a linear operator, also known as matrix channel. We will design network codes based on directed graphs, and not make any assumptions on the underlying network protocols, except assume randomised code construction and non coherent transmission, in order to analyse the error rate at the receiver-end application layer.

Chapter 5 will then introduce our P2P live video delivery application. We'll describe an application that can be run in distributed nodes. We will describe the implementation of prioritised random coding with a push criterion which realises a practical network coding framework for scalable data, and the rate estimation problem arising from distributing scalable data in a collaborative fashion. We'll describe a proactive water-filling rate allocation technique applied to live buffering and streaming, and describe and analyse the result of transmission experiments on the network simulator (ns2), with realistic network dynamics.

Finally, we will characterise the prioritised coding for exchanged data among the peers as a random fountain code over a finite field in Chapter 6. We'll recall the definition of state of the art rateless codes and understand differences and similarities with the premise of a distributed coding system like the one described in the other chapters, and we'll characterise our coding for push-based P2P with forward error correction properties for packet loss protection. We'll explain fast implementations of the decoder and dimensioning of the system for high rate streams, demonstrating the feasibility of such implementation in a real setting and virtually linear variance of end-to-end delay against network latency, which makes it particularly suitable for data exchange in live streaming systems.

Chapter 7 discusses the outcome of this study with considerations about future research direction. Appendices A, B and C conclude this thesis.

Chapter 2

Network Coding for Content Delivery

Network Coding (NC) was first introduced in 2000 [12]. The promise of Network Coding was that of being able to reach the network capacity, calculated theoretically but never reached with traditional routing, e.g. in multicast scenarios. NC has attracted interest in many applications, such as wireless networking, network security, data sharing and storage. It has promised to provide enhanced performance in terms of end-to-end delay, energy consumption, storage, impact of channel errors and data persistence. It has also allowed to find algorithms that reduce the well known intractable problem of finding optimal multicast trees to a solvable problem [13, 14].

The research community has already started exploring these topics and providing applications employing network coding as a system design component, while theoretical aspects of NC are still being studied and developed. NC is undergoing two parallel research paths, one pursuing the theoretical coding aspect of NC, and another one developing practical applications.

In this chapter we tackle both aspects, first describing the fundamental theory of linear network coding, and then analysing the well-established guidelines that allowed putting NC into practical systems. A survey of network coding and error correction theory, which was published as a journal paper, is partially covered in this Chapter and in Appendices B and C.

2.1 Linear Network Coding

The novel idea of NC is to allow network nodes to forward information encoded from the received one, instead of just relaying it like in traditional routing. The main advantage of this approach is that the theoretical capacity of any network can be reached with some form of coding, although the coding solution for any kind of network is far from being found.

Linear coding includes all those information mixing techniques, functions and maps, for which the following properties, defined with broad sense, are valid:

- Additivity: $f(x+y) = f(x) + f(y)$
- Homogeneity of degree 1: $f(\alpha x) = \alpha f(x)$

Coding with linear functions has demonstrated to be sufficient to achieve the upper capacity bound in multicast problems, with single or multiple sources, however it fails to do so in the general case (multi-source, multi-sink, and with arbitrary demands) [15].

Linear network coding was introduced by Li *et al.* [16]. Consider a communication network as a directed graph $\mathscr{G} = (V,E)$, composed of a vertex set V and a directed edge set E. Edges in the graph are considered to have integer capacity. In linear network coding the transmitted messages are linear combinations of the messages at the input edges of each node (Fig. 2.1). A Linear Code Multicast (LCM) is a set of linear encoding functions corresponding to each edge of E, which can convey an information flow from a set of sources $S = \left[s_1, s_2, \ldots s_{|S|}\right]$ to a set of sink nodes $T = \left[t_1, t_2, \ldots t_{|T|}\right]$. We will from now on only restrict our study to single source networks.

Given a non-source node i with input edges set $In(i)$ and output edges set $e \in Out(i)$ we refer to i as $tail(e)$, $e \in Out(i)$ and $head(d)$, $d \in In(i)$. Let $U_d, d \in In(i)$ be the messages in the incoming edges to i, then the message U_e transmitted on edge e can be written as:

$$U_e = \sum_{d \in In(e)} \beta_{d,e} U_d. \tag{2.1}$$

The coding coefficients $\beta_{d,e}$ constitute the *local encoding kernel* of an edge $e, \forall e \in E$, from the incoming edges to its tail node. The *global encoding kernel* of an edge are result of local coding from encoding from source to a given edge e in upstream-to-downstream order and can be written as:

$$k_e = \sum_{d \in In(e)} \beta_{d,e} k_d. \tag{2.2}$$

In practical applications, the messages in hand are are considered as symbols in a finite field $\mathbb{F}_q$ of size q, e. g., a Galois Field (GF) of size $q = 2^m$ or a ring of polynomials of maximum degree $\log_2(q) - 1$, where m is a chosen integer usually corresponding to the bits field.

2.1.1 Algebraic Approach to Acyclic Networks

The algebraic approach to model the network transfer characteristic was proposed in [17]. In this approach, as well as many coding-theoretic studies, the network is assumed to be

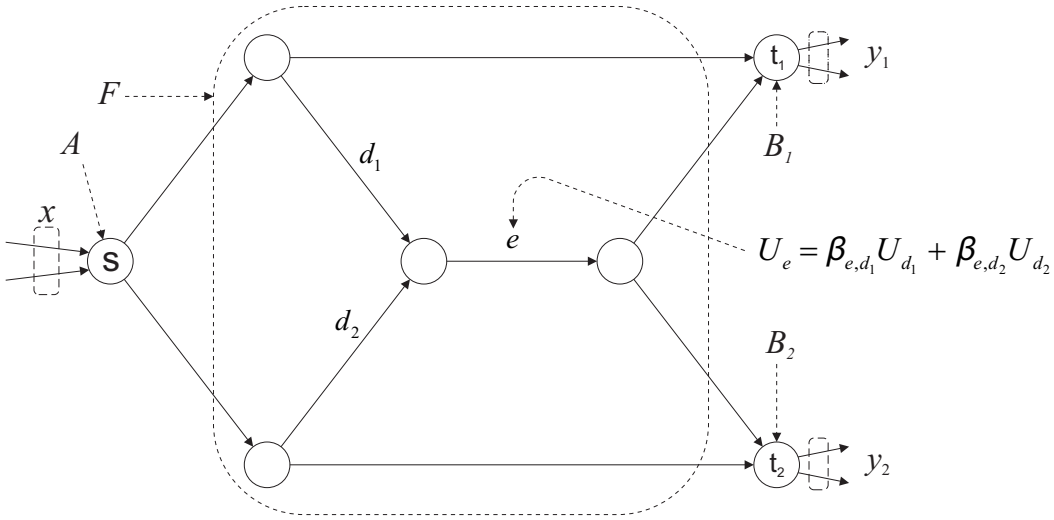

Fig. 2.1: Algebraic modelling of network coding with Linear Coding Multicast (LCM).

without cycles and delay-free. Although seemingly non realistic, it's practically possible to assume this is ensured by other mechanisms, as discussed later on.

An adjacency matrix K is defined as an $|E| \times |E|$ matrix:

$$[K]_{e,d} = \begin{cases} \beta_{d,e} & \text{if } tail(e) = head(d) \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

The network transform characteristic can be then modelled by the following system matrix:

$$M_t = A(I - K[\beta])^{-1} B_t \tag{2.4}$$

In the above equation, A is an $\omega \times |E|$ matrix which represents how the source s routes the ω source symbols into the network edges E. It follows that $A(i, j) \neq 0$ only if $j \in Out(s)$. Similarly, B is an $|E| \times h$ matrix representing how the receiver t maps the information on the incoming edges into h output symbols. I also holds that $B_t(i, j) \neq 0$ only if $i \in In(t)$. Finally I is an $|E| \times |E|$ identity matrix.

Consider a network with a max-flow h defined as the minimum among the max-flows to all destinations $h_t, t \in T$. A source codebook is a vector space $\mathscr{C} \subseteq \mathbb{F}_q^h$. It's important to note that we already introduce the notation where h is the max-flow and $\omega \leq h$ is the source code dimension and information rate for coherence with the network error correction notation used later on.

Linear coding across the network induces a linear transformation on the source message

vector (or codeword) $\mathbf{x} = [x_1, x_2, \ldots, x_\omega]^T \in \mathscr{C}$, with symbols $x_i \in \mathbb{F}_q$. The codeword received by a sink node is a vector of messages $\mathbf{y}_t = [y_1, y_2, \ldots, y_{h_t}]^T$, obtained by means of the linear coding as:

$$\mathbf{y}_t^T = \mathbf{x}^T M_t. \tag{2.5}$$

In light of these definitions, one can find the physical meaning of the network transfer characteristic in the quantities ω and h. Similarly to traditional channel coding, ω is the information rate in terms of information-carrying symbols that are injected in the network at each time slot, whereas h is the actual flow of symbols including random ones generated for loss protection. Network coding introduces the factor of recombination of such h packets at each stage of the transition from source to network. It retains though the meaning of global flow across multiple paths. The actual throughput can be calculated as the product of the number of symbols transmitted during a time slot h by the number of transmissions per second.

The receiver can decode the source messages by inverting the transformation, i. e. by solving a system of linear equations in a finite field ($GF(q)$). In order to deliver decodable information, the global encoding vectors along each path must retain linear independence. To achieve this condition, a subset of ω of the h global encoding kernels, identified by the columns of the matrix M_t in Eq. (2.5), must be linearly independent, i. e., must be a valid base for the coding space. In other words, the system matrix M_t has to have rank equal to ω. It has been demonstrated that an LCM is sufficient to reach the max-flow rate in single-source multicast networks, and such LCM exists if the base field is large enough [16].

Algorithm for construction of the code can follow either the flow approach or the matrix rank verification. Although theoretically they achieve the same result, in general the specific situation dictates whether one algorithm is applicable or not (e. g., whether centralised construction is allowed). This is discussed in Appendix C.

2.1.2 Network Error Correction

Network Error Correction (NEC) was proposed to use the network transfer characteristic for error control purposes [18, 19].

The algebraic model of transmission can be extended by considering errors as random alterations of the symbols on the edges and erasures as symbol cancellations. This alteration is considered as an additive $1 \times |E|$ error vector $\mathbf{z}$ as:

$$\mathbf{y}_t^T = (\mathbf{x}^T A + \mathbf{z})(I - K)^{-1} B_t. \tag{2.6}$$

Particular accent is put on the source codebook $\mathscr{C}$ being an ω-dimensional subspace of the vector space $\mathbb{F}_q^h$ spanned by an LCM, with $\omega < h$. Network Error Correction extends all the knowledge of classic coding theory, such as coding distance, weight measures and coding bounds considering the LCM as a coding operator. The network code can be given similar properties to traditional coding in terms of number of correctable errors and erasures depending on the code length, if similar distance properties are respected at destination, with definitions of distance and bounds given by a number of new measures (see Appendix B).

In broad terms, NEC assesses what is needed for the LCM to sustain a flow ω in presence of errors. Given an LCM, the code redundancy is defined as:

$$\delta_t = h_t - \omega. \tag{2.7}$$

Under certain conditions, and assuming a definition of network Hamming weight as:

$$W_t^{msg}(\mathbf{x}) = W_t^{rec}(\mathbf{x}M_t) = \min\{w_H(\mathbf{z}) : \mathbf{z}F_t = \mathbf{x}M_t\}. \tag{2.8}$$

and network coding distance as:

$$D_t^{msg}(\mathbf{x}_1, \mathbf{x}_2) = W_t^{msg}(\mathbf{x}_1 - \mathbf{x}_2) \tag{2.9}$$

as given in Appendix B, a minimum distance of the LCM can be calculated as

$$d_{\min,t} = min\left\{D_t^{msg}(\mathbf{x}_1, \mathbf{x}_2) : \mathbf{x}_1, \mathbf{x}_2 \in \mathscr{C}, \mathbf{x}_1 \neq \mathbf{x}_2\right\}. \tag{2.10}$$

For a Minimum Distance Separable (MDS) code it holds that

$$d_{\min,t} = \delta_t + 1. \tag{2.11}$$

With such definitions and in accordance with classic coding theory we can define a code as *l-error-correcting* if the receiver can see a consistent coding space of dimension $\omega = \dim(C)$, in presence of an error pattern with weight at most l, and if it is a Minimum Distance Separable (MDS) code, it also holds that:

$$l \leq \frac{\lfloor d_{\min,t} - 1 \rfloor}{2} \tag{2.12}$$

Construction of the network code with these characteristics can be done with various techniques. Despite the remarkable results within the theoretical framework of network coding,

many of the proposed code construction techniques are applicable to networks or graphs where these assumptions apply:

- Cycle- and delay-free network
- Coherent transmission
- Edge capacities known

These conditions also assume the knowledge of the optimal packing of spanning trees. Packing Steiner trees in traditional multicast is the task of finding the minimum cost tree connecting the source to a set of receivers. In network coding finding the trees is also a fundamental task, because it reduces the network to a directed and cycle free graph, and it's necessary to exploit transmission possibilities across multiple paths. Additionally, it relaxes some of the constraints that made the packing problem an intractable optimisation task in the routing case. Finding the trees is not anymore a task of finding separate paths, since superposing multicast trees is allowed by coding, and reduces the optimisation task to a tractable problem. However, the issue of how to build the trees with practical algorithms remains open, since an approach to finding such paths which is optimal and distributed is hot topic of research.

Randomised algorithms as well as deterministic and iterative construction algorithms based on the aforementioned assumptions are discussed in the next section and in Appendix C.

2.2 Practical Network Coding

Chou's model for practical transmission has been proposed in [3, 20, 21] and introduces a number of techniques to effectively implement coding techniques at intermediate network nodes, including introduction of *generations*, and a packet format with appended coefficients. This model abstracts from the previous assumptions of unitary and synchronised links, thus assumes that there can be random delays and losses throughout the network, variable link capacities as well as unknown broadcast capacity and max-flow to the receivers. The main technique for transmitting/decoding with randomised network coding is to include the encoding kernels as an extension of the packet payload. When packets are re-encoded at intermediate nodes, the coefficients are encoded as well. The receiver can build the decoding matrix by parsing the portion of data prepended to the payload.

2.2.1 Randomised Construction

In most situations, performing a centralised construction of the network code is very impractical. Additionally, coherent transmission assumes that all nodes encode according to the centralised design, which would require an unrealistic amount of coordination between the nodes and the senders and receivers. Distributed and randomised approaches allow intermediate nodes to handle, encode and retransmits data without knowledge of the global status of the network or what code is applied by other nodes, still effectively providing network coded transmission.

In order to achieve this, any forwarding node can chose randomly and autonomously the multiplicative coefficients to use to combine the outgoing packets or symbols. As a consequence, sink nodes receive a randomly encoded version of the source data by means of a random transformation operated all together by the network. Little a priori information is required, however the source should acquire an estimate of the sustainable flow of information to avoid rank deficiency at the receiver, whereas knowledge of the matrix channel characteristic is to be retrieved by the receivers to allow decoding. One additional constraint for the forwarding nodes is to know the flow traversing the node itself, which allows to respect the overall network flow. With unknown bandwidth restrictions one can as well turn to partially ranked approach to prioritise data, as explained in Section 3.2.2 and in [3]. We will however take a rateless approach which aims at exploiting every transmission opportunities, as described in Chapter 5.

As far as the receiver is concerned, the message can be decoded if the matrix channel conserves full rank of the source coding space. This is however, a random event that depends on many factors. With a coherent coding approach, the probability of having a valid random network code is bounded by a limit that depends on the field size, and the number of intermediate nodes and receivers. A sufficiently large base field is usually enough to ensure the existence of the code and the possibility of successfully decoding the transmission in any network [22]. Such choice of system design is then what determines the probability of failure of the transmission. A Table of probability bounds is given in Appendix C.

Additionally, in presence of link failures or errors the source information can also be retrieved correctly, with additional constraints. If the source codebook has a degree of redundancy $\delta_t = \text{mincut}(t) - \omega, \forall t \in T$, where $\omega = \dim(\mathscr{C}) \leq h_t$, the minimum distance can be eventually equal to $\delta_t + 1$, like for traditional error correcting codes, with some probability in randomised code or if deterministically constructed. In randomised coding minimum distance is a random variable and the probability of being equal to $\delta_t + 1$ has been found to be also dependent on network characteristic [23]. A survey on code construction

with randomised approaches is discussed in Appendix C. We will justify certain choices for randomised coding by the degree of redundancy δ_t and the error detection/correction properties that we want to achieve.

2.2.2 Packet Format

In a packet network we assume that all codewords in a packet are subject to the same encoding at the intermediate nodes. A non-coherent model can be assumed, meaning that the packets don't come necessarily from disjoint paths, but rather from any of the incoming edges and most probably at different time instants.

Let's introduce the notation of a source packet $\mathbf{b}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,w}]$ which includes every i-th symbol of w successive codewords $\mathbf{x}_1, \ldots, \mathbf{x}_w$. Equivalently, we can define a received packet as $\mathbf{d}_i = [y_{i,1}, y_{i,2}, \ldots, y_{i,w}]$, resulting from several recombinations of a set of ω packets that can be expressed mathematically as follows:

$$\begin{bmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_h \end{bmatrix} = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,w} \\ \vdots & \vdots & \ddots & \vdots \\ y_{h,1} & y_{h,2} & \cdots & y_{h,w} \end{bmatrix} = M_t^T \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_\omega \end{bmatrix} = M_t^T \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,w} \\ \vdots & \vdots & \ddots & \vdots \\ x_{\omega,1} & x_{\omega,2} & \cdots & x_{\omega,w} \end{bmatrix}, \tag{2.13}$$

where we recall the definition of the $\omega \times h_t$ matrix M_t from Eq. (2.4). This Equation is also the transposed of Eq (2.6).

Network coding transmission is defined coherent when the receiver knows the network topology and the coding functions, and non-coherent otherwise. When the coding functions are randomised and change at every transmission of a batch of packets, one has to assume the non-coherent model. However, in-band communication of the global encoding kernels ensures that the transfer characteristic can be deduced at the the receiver.

This is done by pre-pending a vector of length ω to each packet at the source, which has only one unitary element, and has zeros elsewhere. Each packet's pre-pended vector has the unitary element in a different position, so that a generation of packets packed into a matrix such as the one in Eq. (2.13) can also be rewritten as follows:

$$\mathbf{Y}_t = M_t^T \begin{bmatrix} 1 & & 0 & x_{1,1} & x_{1,2} & \ldots & x_{1,N} \\ & \ddots & & \vdots & \vdots & \ddots & \vdots \\ 0 & & 1 & x_{\omega,1} & x_{\omega,2} & \ldots & x_{\omega,N} \end{bmatrix} = \begin{bmatrix} & y_{1,1} & y_{1,2} & \cdots & y_{1,w} \\ M_t^T & \vdots & \vdots & \ddots & \vdots \\ & y_{h,1} & y_{h,2} & \cdots & y_{h,w} \end{bmatrix}. \tag{2.14}$$

The presence of the identity matrix (see also Fig. 3.8 for a visual representation) helps

understanding how the encoding kernels, result of coding operations at the core network nodes, are communicated in-band as a product of network coding, and are used by receiver to construct a linear system of equations and recover the source information. The cost of this scheme is an overhead of ω encoding factors per packet. This overhead is a small percentage of the total bandwidth for sufficiently large packets and no higher than average ordinary throughput losses. On the other hand, protection against packet losses can be achieved by arbitrarily increasing the number of produced packets. The probability that the received packets are enough for decoding $\mathbf{b}_1, \ldots \mathbf{b}_\omega$ is acceptably high (above 95%) for field sizes greater than 2^8 [3] and increases when $h > \omega$ packets are collected. This probability was calculated in [24] and [25] as being equal to:

$$P_{\Omega,H}(\omega, h) = \prod_{n=0}^{\omega-1} \left(1 - \frac{1}{q^{h-n}}\right), \tag{2.15}$$

which expresses the exact probability of decoding ω packets from h randomly encoded in a field of size q. The rateless nature of randomised coding has been pointed out as one of the advantages of this approach against packet losses [26]. We will reprise this technique in Chapter 6 and analyse packet loss protection properties and computational load.

2.2.3 Buffering Model: Generations

In real networks, the packets traverse the network with random delays. In order to synchronise the encoding operations on subset of packets of reasonable size, the concept of *generation* is introduced. Packets are grouped in generations and assigned an identification number. The number of packets in a generation is indicated with ω, with clear implication of the code dimension. At the intermediate nodes the packets are buffered until a transmission opportunity arises. A new packet is encoded from and only from the buffered packets of a specific generation with random coefficients, before being transmitted to the next hop. A small field in the packet header is used to distinguish the generation to which the packets belong to. Due to the randomised nature of the code, a small probability exists that an incoming packet is linearly dependent from the packets that have already been buffered. Packets that are linearly independent from the current buffered packets in the same generation are called *innovative* and are added to the buffer. *Non-innovative* packets are discarded. Gauss-Jordan elimination is traditionally used to check for non-innovative packets. This also progressively decodes the incoming data, so that when the last incoming packet from a generation is decoded, the original information is available to be consumed at the receiver.

Buffer flushing policies have to be implemented to keep the network on a reasonably updated status. We implemented a specific policy for our video streaming data.

2.3 Coded Multicast in Overlay Networks

To embrace network coding in existing packet networks one must think of the implications in the network protocol stack. It is very unlikely that NC will be implemented at transport layer of the stack, due to the well-established IP routing mechanisms and protocols. In the deployment of multimedia and content distribution networks, implementing routing and multicast functionalities has moved towards the application level to address most problems typically associated with IP multicast. Quality of Service (QoS), adaptivity to network dynamics, and ability to be self-organising are a few of the desired properties of a Content Delivery Network, addressing many of the limitation of IP multicast [11]. Additionally, handling the information at application layer, is an effective option to perform media-aware routing, and by re-encoding the information, also network coding. IP legacy systems can offer the options of packing multiple unicast session and IP multicast. Such options are known for having a very high throughput load for the source node, and limitations in terms of scalability and latent reaction to changes in the topology. With an overlay topology, where some of the nodes are capable of handling information at application layer (some of which might be end-user systems), multicast can be handled by the End-Systems and it might additionally rely on a number of intermediate nodes to manage multicast functionalities. End-system multicast can be seen as case of Application Layer Multicast where all multicast nodes are end systems.

The content delivery system described in this thesis is based on an application implemented at the top levels of the stack, handling coding and data exchange between the network nodes at application layer. A first problem in efficiently delivering information, both in traditional networks and coding-aware networks, resides in the process of network discovery. This information allows then to build paths to reach each destination. Two approaches can be implemented:

Tree based systems In this approach routing trees are calculated, spanning separate paths between source and receivers, optimised to achieve either shortest paths, maximum fairness, or maximum bandwidth. Although the problem of packing Steiner trees is traditionally considered computationally intractable [27], the network coding assumption generalises the problem and reduces it to a manageable optimisation problem [14, 28, 29]. Collaboration

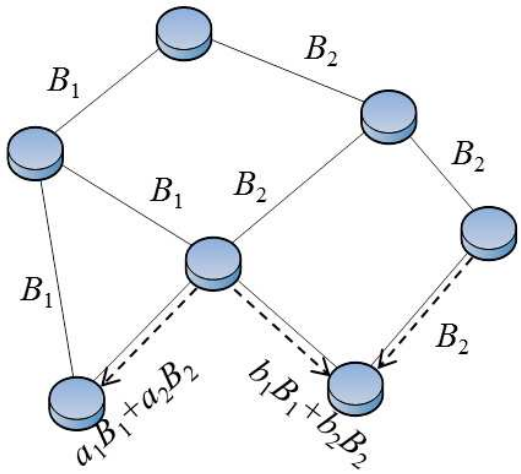

Fig. 2.2: Distribution of data in a coded overlay.

between nodes belonging to parallel paths is difficult to exploit, thus the data flow rate suffers from the bottleneck links on the chosen path.

Cooperative mesh systems Mesh networks have the advantage of using all the available connections to discover possible download opportunities to retrieve the data. The drawback of cooperative systems in the traditional case is that, missing the big picture, bandwidth efficiency is lower, nodes may be subject to reception of a highly redundant stream and might run into the problem of chunk rarity. Coding on cooperative mesh overlays seems a viable option to reduce inefficiencies and the missing-chunk problem. Concurrent downloads enhance coding and path diversity [26]. This has been the point of strength of peer-to-peer protocols, whose distribution mechanism is exemplified in Fig.2.2.

2.3.1 Mesh Based Architectures: Peer-to-Peer

A Peer-to-Peer (P2P) network is an overlay network which, as opposed to CDNs or other infrastructure networks, is composed by nodes that spontaneously join and leave and that are generally all final consumers of the data (whereas in CDN, core nodes are only cache points). P2P does not need to build paths, instead all nodes share their cached data with other connected users. P2P protocols can be classified depending on the approach they take to distribute the chunks of data:

Pull-based systems allow nodes to request specific chunks of data. Users need to communicate to each other the respective availability of data chunks.

Push-based approaches allow senders to decide which chunks to push to other users, based on some scheduling criterion.

Introducing network coding in P2P systems has the benefit of reducing the need to transmit specific chunks of data, improving time efficiency as well as shortening scheduling procedures. Any coded packet is potentially *innovative* for the receivers, which don't have to receive a specific chunk of data, but can instead progressively decode the data and at the same time contribute with their portion of encoded chunks (See Fig. 2.2). In dynamic systems, where either the nodes can depart or fail at unexpected times or the network can have shortages, congestions and bottlenecks, some chunks might remain unrecoverable, impeding most of the nodes, if not all, to recover the data. This is well known in file-share systems as the *missing chunk* problem. Distributing randomly encoded data can instead make sure that the whole data is distributed in different forms across the whole network, so that even if the source departs from the network, the other nodes can exchange their buffered data to recover the source data.

The Avalanche system, developed by Microsoft, was the first research product that demonstrated improved performance with respect to traditional system, in terms of reduced download times and better usage of resources. Peer-to-peer systems might be the architectures that can benefit the most from random network coding and will be the fundament of our content delivery system proposed in Chapter 5.

Both CDNs and P2P networks are a first step towards a class of communication networks known as Information-Centric Networks (ICN) [30]. With ICN, object retrieval mechanisms are based on content rather than location, allowing the network to choose the best way to cache and delivery such objects upon request. Similarly to the idea initiated with the aforementioned architectures, caching and localisation becomes a tool for transparent delivery of content identified by the naming convention rather than the original source, optimised with regards to the location of the requesting user. Network coding natively provides the means for efficient content caching across the ICN, as well as the possibility for users to retrieve collaboratively from different rendezvous points, with a distributed yet efficient system to manage objects at content level rather than a packet or chunk level [31].

2.4 Conclusions

In this chapter we have highlighted the fundamental results in network coding that allow us to implement and experiment data transmission and streaming on a complete network

environment. Although being a young discipline and several new mathematical tools being still studied, practical systems exploiting network coding have demonstrated validity and potential for future developments, especially peer-to-peer systems. The benefits of advanced processing in the network come with a price. This entails additional computational complexity for coding and decoding, some fixed overhead for in-band signalling and variable overhead due to random coding, as well as the need to carefully plan the data flow in order to not fall into the rank deficiency problem. With these premises, the focus of our research is to implement and experiment streaming of scalable data on a peer-to-peer environment with network coding, providing a solution for rate allocation with overhead awareness (Chapter 5) as well as for sustaining the computational complexity (Chapter 6).

Chapter 3

Scalable Coding and Prioritised Streaming

Scalable Coding, also referred to as Layered Coding (LC) responds to a multitude of challenges imposed by media broadcasting. Looking at systems employing dynamic streaming (such as Youtube, and DASH and industrial alternatives) it becomes clear how providing a Scalable Video Coding (SVC) service can optimise the use of resources and the overall quality of service on the platform. Conventional codecs only target one single configuration of video in terms of quality, resolution and frame rate. While one configuration may be a good fit for one target system (e. g., full HD at 30 frame per second with fibre optic broadband) the same configuration is probably not good for another terminal (e. g., laptop with Wi-Fi connection, hand-held device over 3G). Dynamic Adaptive Streaming over HTTP (DASH) and other dynamic streaming protocols target this issue by storing multiple encoded versions of the same video. The downsides of this approach are mainly two:

- The server has to decode and re-encode the live video in different versions, often without any smart transcoding.
- Caching nodes, or peers in the P2P case, need to store all available versions of the video.
- Once the segment is downloaded the user cannot reduce the quality on the fly (for example because of corrupted stream or sudden lack of computational resources) and can only request and download the segment again at a different quality, or wait and switch for the next segment.

Scalable coding provides a way of gradually switching between video configurations by decimating the stream at any point to reduce the quality. Conversely, by retrieving additional

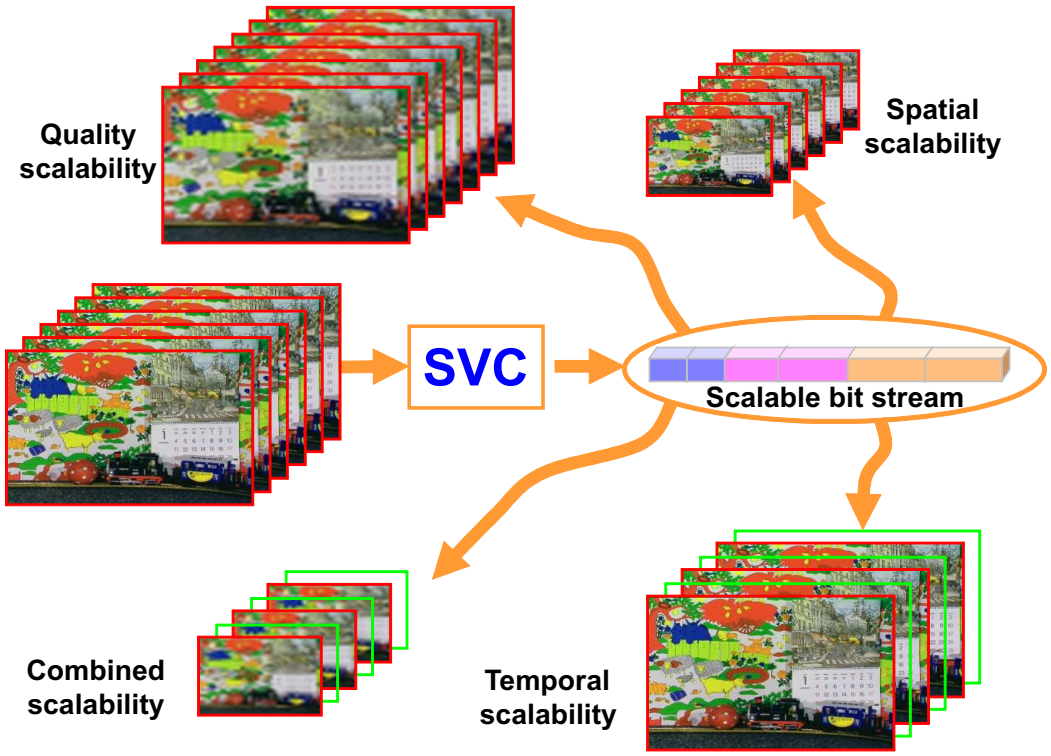

Fig. 3.1: Example of partial decoding for temporal, spatial, quality, or mixed scalability.

enhancement data, the device can reuse all data retrieved for lower rate video and play back an enhanced quality video.

A scalable extension of the H.264/AVC, known as H.264/SVC, has been standardised and is now the *de-facto* state-of-the-art in scalable video coding and standard reference for other codecs. Since shortly after the introduction of the High Efficiency Video Coding (HEVC), follow up to the AVC and most efficient codec to date, exploration has started to provide the scalable extension to HEVC in the near future. Among the alternatives, the Wavelet-SVC codec (also known as aceSVC or MMV-SVC) provides comparable performance with an approach based on wavelet transforms instead of discrete cosine transforms [32, 33].

The main objective of the layered coding has been to enable multiple decoding possibilities from a single encoded bitstream. This might be the case of data to be retrieved from local storage with the possibility of selecting directly the desired configuration. Or it can be the result of an adaption performed by a network element. For instance, in a hierarchical structure like the one shown in Fig. 1.1 a group of edge nodes caching the full rate video can stream different versions of the video to the connected users depending on the requested quality. In case of partial reception due to packet jams, reduced bandwidth, or partial data

corruption, the stream can be decimated and decoded up to the lowest layer received intact, without interrupting the playback.

Joint Source-Channel Coding (JSCC) is a key technique to get the best out of scalable encoded video. In scalable coding, errors and losses of data on enhancement layers don't interfere on the decodability of lower layers, allowing to limit distortion to fine-granularity visual impact, confine corrupted data on less important data and keep cross-layer interference limited. Unequal allocation of redundant bits promotes the correct and error free reception of portions of data that have higher impact on the visual quality, such as the base and first enhancement layers of video, or the I-frames. This allows ad-hoc boosting of the overall performance in terms of objective quality, continuity and subjective quality of service.

This chapter introduces channel coding approaches to scalable data for channels prone to erasures and without retransmission protocols. We'll describe the latest approaches to scalable video delivery in order to introduce then our network coding approach for scalable error-control coding. This chapter won't describe the scalable coding as such – based on discrete cosine transform as opposed to wavelet transform – which is commented in Appendix.

3.1 Scalable Video Coding and Transmission

The idea behind a layered coding approach is to have decoding flexibility depending on device or variable channel conditions. Large and heterogeneous networks, where receivers experience diverse reception quality, can benefit from differentiating the delivered video without having to produce and store several encoded versions on the video, like it's currently done with DASH.

The encoded video is segmented in the transform domain, where various levels of a transformation hierarchy can be identified. For instance Dynamic Cosine Transform (DCT) coefficients or wavelet transform subbands form the coding hierarchy for frame reconstruction in the spatial dimension, whereas the I-P-B frame hierarchy or the lifted wavelet transform provide a hierarchy in the temporal dimension, and truncated entropy coding in the SNR domain. These components are organised into units that can be transmitted independently and decoded at the receiver on condition that the segments are decompressed respecting the coding dependencies. For example spatial scalability can be encoded into a first stream containing the information of the video at low resolution (QCIF, 176×144). A first enhancement stream can follow. When decoded together with the base stream, this produces the video at CIF resolution (352×288). A last enhancement stream can carry the

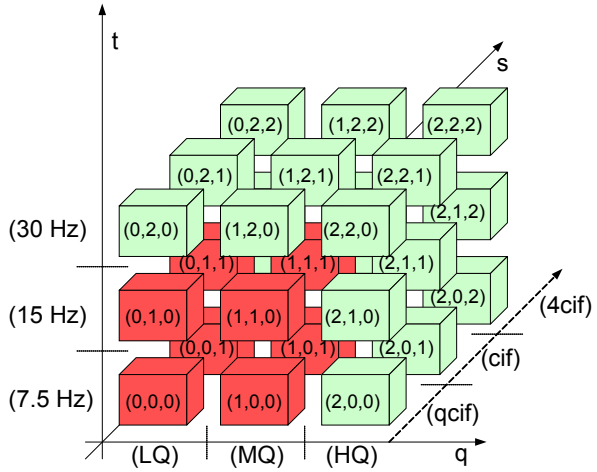

Fig. 3.2: Representation of atoms of an SVC stream with 3 levels of temporal, spatial and quality scalability. A possible extraction is highlighted [1].

additional data for 4CIF resolution (704×576). The standard scalable video coder and its alternatives are called fully-scalable, because they provide scalability in terms of temporal resolution (frame rate), spatial resolution and quality (SNR). A stream with scalable video is shown in Fig. 3.1.

The encoded layers can be seen as a multidimensional hierarchic structure as shown in Fig. 3.2, where the coding units that need to be collected in order to decode a target layer are highlighted. Extraction of the necessary units, or atoms, can be performed by parsing the bit stream (Fig. 3.3) and reconstructing a stand-alone decodable stream including all dependencies from base layer to desired configuration.

3.1.1 Joint Source-Channel Coding

The fundamental theorem of source-channel separation states that source and channel coding can be completely independent, provided the freedom of using arbitrarily long block coding length [34]. By stating that they can be performed regardless one another, source coding parameters such as quantisation step etc. only depend on the information rate of the channel, whereas channel coding only has the role of transmitting reliably the source rate by using a sustainable channel rate for source protection.

However, it has been argued and demonstrated that in the case of limited resource availability, including a limited block code length, joint design of source and channel coding yields to the best performance [35]. In this work we are interested in performing channel and network coding with source coding awareness in order to globally improving the

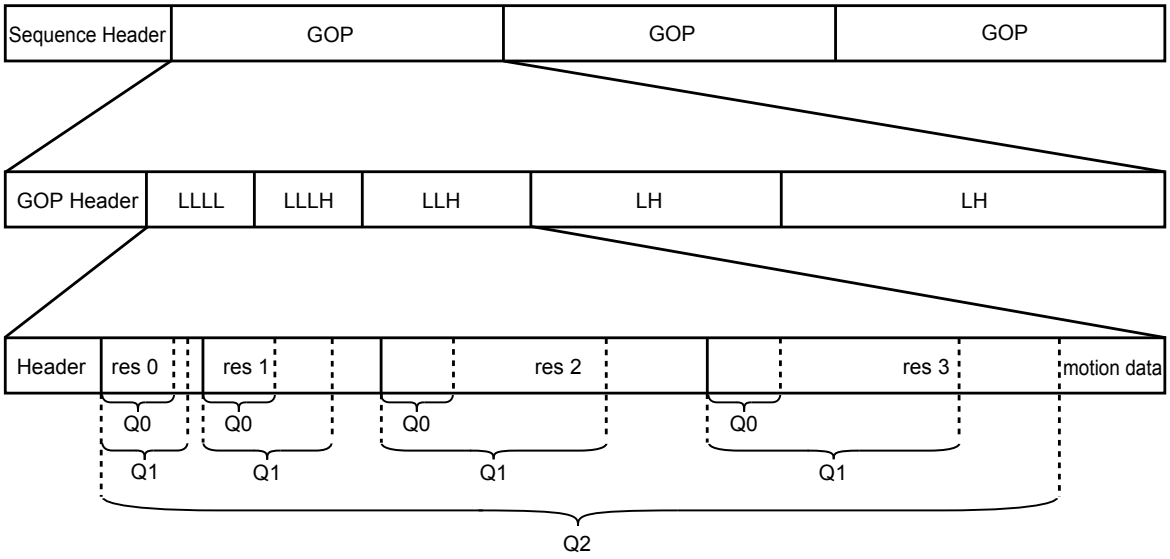

Fig. 3.3: Organisation of the embedded bit stream with highlighted atoms for temporal, spatial and quality scalability [2].

transmission from the point of view of the visual quality and the resilience against channel degradations [36, 37]. With layered coding one can rely on techniques such as Prioritised Encoding Transmission (PET), where segments of data which have more impact in the visual quality, or that other segments of data are dependent on, are assigned higher priority, and encoded and delivered with higher loss and delay resilience.

We introduce now a notation that will be used in the remainder of this thesis. We classify the scalable data in priority classes, assigning lower indices to *high priority* classes and higher indices to indicate *low priority* classes. We also assume that any class of priority $i > 0$ will need all of the priority classes $j < i$ to be received, before we can decode i. With reference to Fig. 3.4, where blocks $\mathbf{b}_i, i = 1, \ldots, \omega$ are ordered by coding layer and priority, we define L priority classes $0, \ldots, L-1$ in terms of non overlapping coding sets of length

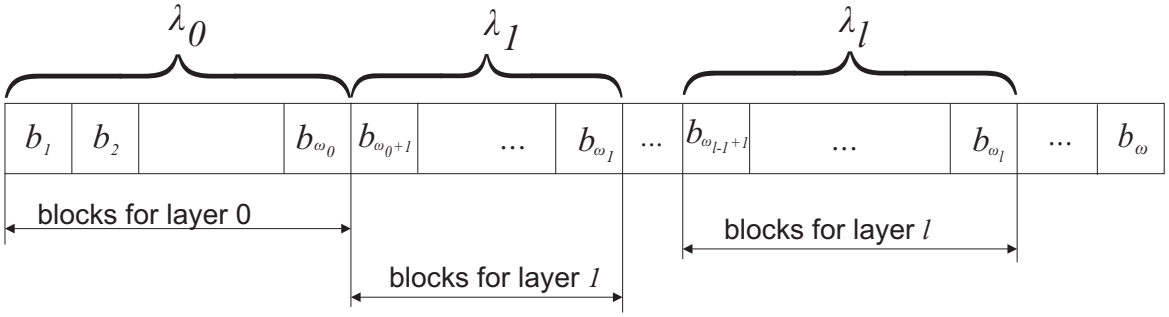

Fig. 3.4: Arrangement of blocks in the buffer from layers of decreasing importance.

$\lambda_0, \ldots, \lambda_{L-1}$, composed of blocks $\mathbf{b}_i$ as follows:

$$\begin{aligned} S_0 &= \{\mathbf{b}_1, \ldots, \mathbf{b}_{\lambda_1}\}, \\ S_1 &= \{\mathbf{b}_{\omega_1+1}, \ldots, \mathbf{b}_{\omega_2}\}, \\ &\ldots \\ S_l &= \{\mathbf{b}_{\omega_{l-1}+1}, \ldots, \mathbf{b}_{\omega_l}\}, \end{aligned}$$

where λ_l is the number of packets in a stream containing the data of priority class l, and

$$\omega_l = \sum_{i=0}^{l} \lambda_l$$

is the cumulative number of packets from base layer to layer l, also including all the data from classes of higher priority necessary to decode layer l. The definition of generation follows as the union of all the classes, so that the generation length in number of packets can be expressed as:

$$\omega \equiv \omega_{L-1} = \sum_{i=0}^{L-1} \lambda_l.$$

We note how ω_i expresses the cumulative set of blocks from the beginning of the generation to either layer l, or if the subscript is dropped, until the end of the generation. It's important to notice that λ_i can represent, in presence of FEC precoding like in the MDC schemes, not only the number of source coding packets for stream i, but it can also include redundancy packets encoded with traditional block codes from data of class i.

Although the term Unequal Error Protection (UEP) is often use to refer to any kind of scalable or weighted channel protection, we can identify specific ways the benefits are perceived by the receiver.

- Unequal Error Protection (UEP) refers to channel coding that provides, in different measures, robustness against *error and erasures*.
- Unequal Loss Protection (ULP) refers to differentiated robustness against *erasures* of information, like *packet losses* on networks.
- Unequal Recovery Time (URT) refers to the possibility of decoding higher priority classes *earlier* than others.

All three features are desirable in our data delivery system, and we will tackle them by means of different aspects in the following chapters.

Although traditional block codes still play an important role in packet-based communi-

cation, in the remainder of this chapter we will describe first a new class of codes, called *fountain codes*, that rely on randomness and rateless-ness to combat very efficiently packet losses, and in a different measure packet errors as well. We also briefly analyse multiple description coding based on block coding, which is the base for our coding system proposed in the next Chapter.

3.2 Prioritised Rateless Coding

Classic block codes have been studied for unequal protection purposes, either via partitioned generation matrices [38], with concatenation [39], or by shortening and puncturing [40]. The main limits of block approaches are: Decoding complexity and limitation in the block length. Decoding algorithms, even in their fast implementation, are usually complex and have to be implemented in hardware to reach satisfactory performance. Deterministic codes also have a limit imposed by the number of blocks that can be encoded, e. g. Reed-Solomon codes can encode only up to $q-1$ blocks, being q the size of the Galois Field (GF). When one value of q is not enough (i. e., 256 for elements of size 1 Byte) one has to increase the field size to, i. e., to 16 bits, further increasing the encoding and decoding cost.

Additionally, since our goal is to applied coding to a distributed setting and perform network coding, deterministic codes fail in provide the necessary diversity. For the sake of clarity let's imagine two nodes that have stored the blocks A and B, and that have to deliver both data to a third receiver, ideally each one providing one packet. In absence of any coordination, both nodes would have chances to pick the same encoding configuration (A, B, or c_1A+c_2B with the same coefficients c_1 and c_2), delivering to the third node potentially redundant information. Fountain codes embrace the following characteristics which have found very positive applicative outcomes:

- Randomness
- Ratelessness
- Linear complexity

In the following we describe state-of-the-art rateless codes, and optimisation of fountain codes for unequal protection coding. We'll evaluate the extent and possibility of application to our content delivery mechanism. It's worth noting that when possible we'll use the same notation introduced for network coding.

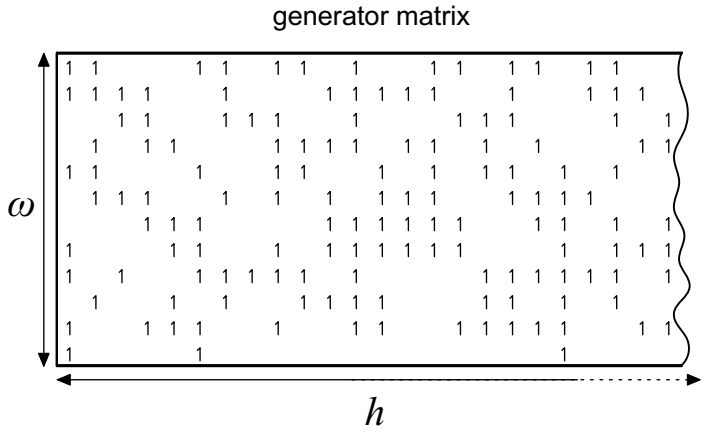

Fig. 3.5: Generation matrix of a binary fountain code

3.2.1 Digital Fountain Codes

The *digital fountain* idea entails the concept that, of the multitude of encoded data produced by the sender for a set of ω blocks, a receiver needs to receiver any ω and only slightly more than ω of these packets to decode the source data, without need for receiving specific packets or reordering them. Most of these codes have been proposed for erasure correction, which is the main channel impairment one has to deal with on the internet, together with random delays. This is a characteristic that we keep in the design of our coding framework, i. e., fountain codes in $GF(q)$, as well as the need to transmit in-band the coding characteristic.

Random Linear Fountain A random linear fountain produces packets encoded by combining together a number of source block grouped in random sets. An outgoing packet is encoded from the ω source packets according to:

$$\mathbf{d}_n = \sum_{k=1}^{\omega} M_{k,n}\mathbf{b}_k$$

where M is coefficients matrix whose columns define random (or pseudo random) coding sets of input packets [41] and, in most fountain and low-density parity-check codes implementations, are binary and have a degree of sparseness, as shown in Fig. 3.5. Randomness and rateless streaming are the key of the random fountain, as new packets can arbitrarily be generated and the source blocks can be decoded by any independent subset of size ω. Decoding can be performed via Gaussian elimination, whose complexity is in the worst case limited by $\mathcal{O}(\omega^3)$.

We will base our rateless coding for P2P delivery on a Random Linear Fountain over

$GF(q)$. This will be necessary in the network coding scenario, where coding diversity is essential to increase the probability of succesful decoding at the receiver. We will however now briefly describe state of the art fountain codes in order to introduce the use of expanding windows for scalable coding. We will distinguish between binary fountain codes and fountain codes over $GF(q)$, implying that the latter will also have dense coefficients.

Non-scalable Fountain: LT and Raptor codes Luby-Transform (LT) codes are derived from the digital fountain, and they make use of a degree distribution and an easy decoding algorithm to reduce the computational cost of decoding to $\mathcal{O}(\omega \log(\omega))$. The main difference with the random linear fountain is the sparse coefficients, meaning that only a relatively small number of blocks are combined together in an outgoing packet, and a degree distribution, called Soliton distribution, which is the probability distribution of the number of packets encoded together in an outgoing packet. The degree distribution allows decoding via the Belief-Propagation (BP) algorithms. This starts from decoding the symbols with degree equal to 2 from those with degree equal to 1 (which are know) then it proceeds to decode the degree equal to 3 and so on [42].

Raptor codes improve over LT codes by adding a precoding stage which feeds into the LT code symbols generated with a systematic Low-Density Parity-Check (LDPC) code [43]. This allowed to reduce the input degree of coded symbols and thus the complexity, while ensuring that all source symbols are covered in the coding process and thus resolvable. The decoding complexity of raptor codes is $\mathcal{O}(\omega)$ which allows to decode great amounts of data all with software implementations. Both LT and Raptor codes are designed to achieve a target performance with a relative overhead ε. This overhead includes additional packets to be received in order to be able to decode the source set of packets with high confidence.

Although being the best-performing erasure correcting codes, retransmission before complete decoding is essential in our distributed coding architecture, whereas Raptor decoding takes place once $\omega(1+\varepsilon)$ packets have been received. Adequate diversity when re-encoding raptor encoded symbols is also needed. Distributed LT codes have been proposed for simple scenarios [44]. Our network coding system will rely on progressive decoding instead, to allow early retrieval of base layers and additionally identify non-innovative packets. We argue thus that Gaussian elimination decoding is necessary and raptor codes have difficult application in cooperative prioritised network coding.

Scalable Fountain via Weighted and Expanding Window Codes Binary fountain codes supporting scalable data transmission via UEP can be implemented in two different ways.

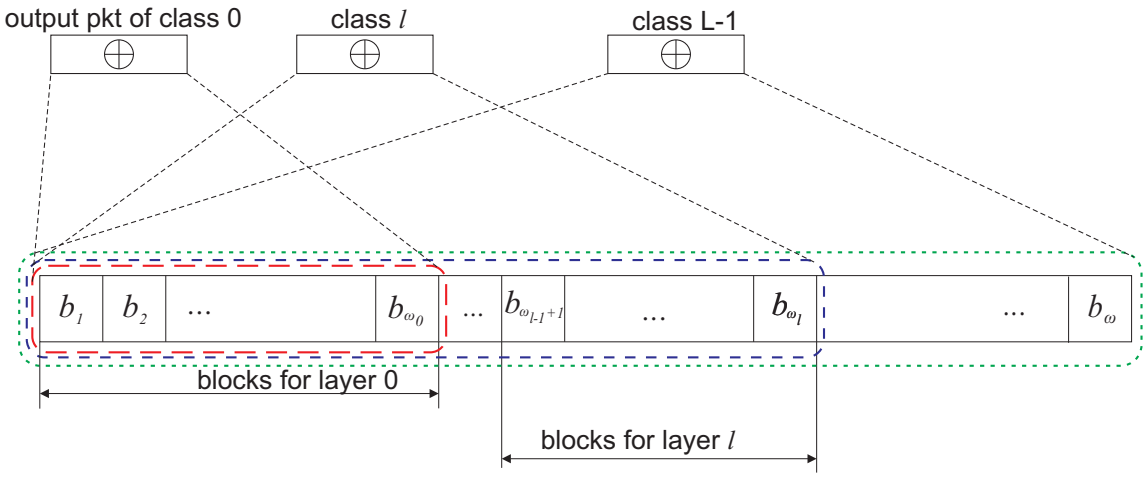

Fig. 3.6: Expanding windows to create priority coding block sets.

One first approach is to have weighted degree distribution of the coded symbols with regards to the symbols to combine [45]. Expanding Window Fountain (EWF) codes instead use an approach similar to the one we propose in the network coding overlay in Chapter 5, shown in Fig. 3.6. An outgoing packet from a specific class is encoded by combining all packets with the highest priority (lowest layer) to the one that is being transmitted, while the other blocks are masked with zeros. With the degree-distribution driven approach, and especially in the first method where the persistence of data in each layer is randomised, the successful decoding of a layer i from a subset of $\omega_i + \varepsilon_i$ packets might need large values of ε. Prioritised transmission is a process of random selection of windows, driven by a selection distribution which minimises the impact of losses on the probability of decoding the lower classes [46, 47].

In all these methods, usually erasure protection is done by considering whole packets are symbols, so that an outgoing packet is one only outgoing symbol, and in case of one packet loss, just one encoded symbol is lost. Our method is very similar to EWF in the choice of windows and generation of multiple packets except that ours is non-sparse and uses Gaussian elimination for partial decoding of sublayers.

We will explain in detail the use of Fountain Rateless Coding over $GF(q)$ with network coding overlay in the following Chapters.

3.2.2 Multiple Description Coding via Channel and Network Codes

Multiple Description Coding (MDC) has been proposed to allow partial decoding with graceful degradation under random loss of information without need of prioritisation [48].

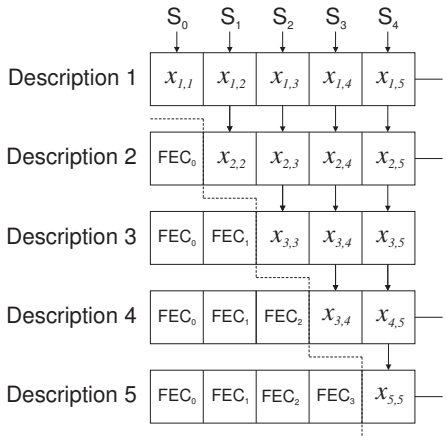

Fig. 3.7: Scheme for Multiple Description Coding via Forward Error Correction and for unequal packet-loss protection.

Although the concept best applies to specifically designing source compression coding to fit the task (i. e., channel splitting, even/odd frames separation, etc.), MDC on simply scalable codecs is possible.

We describe now two MDC techniques based exclusively on channel or network coding, the first based on Forward Error Correction, and the second on random network coding, which share the idea of prioritising by interleaving different classes with variable rate codes, and constitutes the base for the detection and deletion method presented in the next Chapter.

FEC-based Multiple Description Coding A method to generate multiple descriptions of a scalable stream via variable rate codes is shown in Fig. 3.7 [49]. Firstly, blocks of equal length are obtained by applying erasure codes with variable rate to each priority class stream. If a stream of class i is composed of k_i symbols, a FEC code is applied to it, with a rate so that $k_i r_{FEC}^{(i)} = w\lambda_i$, where λ_i is the number of packets that stream i is divided into, after FEC coding, and w is the packet length. The encoded block length n is fixed, and variable FEC rates ensure that the following is true:

$$\lceil \frac{\lceil k_0/r_{FEC}^{(0)} \rceil}{\lambda_0} \rceil = \lceil \frac{\lceil k_1/r_{FEC}^{(1)} \rceil}{\lambda_1} \rceil = \ldots = \lceil \frac{\lceil k_{L-1}/r_{FEC}^{(L-1)} \rceil}{\lambda_{L-1}} \rceil. \tag{3.1}$$

The base layers are encoded with a heavier code than enhancement layers, with rates:

$$r_{FEC}^{(0)} < r_{FEC}^{(1)} < \ldots < r_{FEC}^{(L-1)}. \tag{3.2}$$

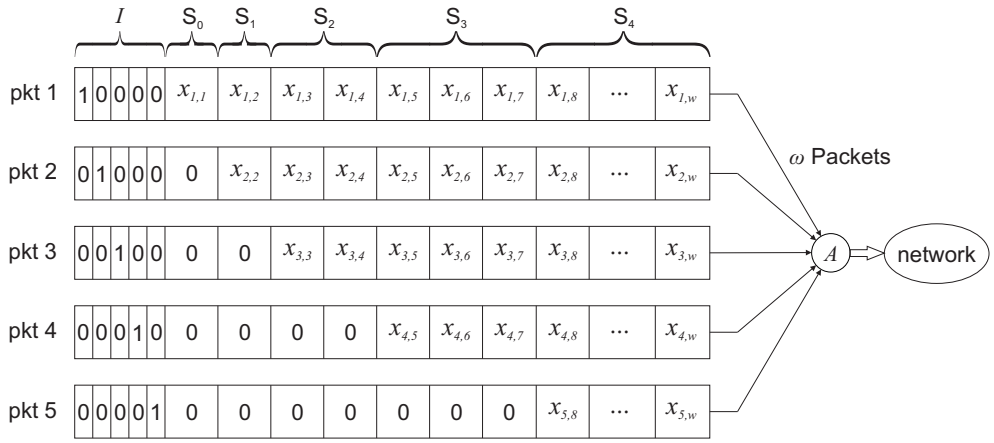

Fig. 3.8: Source vector partitioning and redundancy for Priority Encoded Transmission with network coding [3].

When packetised, the streams are interleaved allocating into each packet a fraction of the $w\lambda_i$ encoded blocks from each stream, as shown in Fig. 3.7, for a total of $\omega = \sum_i \lambda_i$ packets.

At the receiver side, erasure decoding is performed after de-interleaving the streams. With reduced throughput, when part of the packets are lost, the sub-streams are truncated to fewer data symbols to decode the source data from. When the remaining data is at least equal to the information rate in the coded stream, erasure decoding is still possible. Since the sub-streams carry a different information rate, the cut point will be different for each sub-stream. Namely, when l packets are lost, erasure decoding is possible from the $\omega - l$ remaining packets, for all streams i that verify th following:

$$l \leq \omega - k_i.$$

Partial decoding of the video can then be performed up to the highest video layer for which all his dependencies are also decodable (all layers from the base one to the one under consideration). If Eq. (3.2) holds, then the previous statement is true for any decodable stream.

In the following Chapter we will explain the use of this MDC construction in conjunction with network coding to build our Detection/Deletion method for error correction.

Priority Encoding Transmission via Network Coding Similarly to the previous approach, one can use an interleaved scheme as shown in Fig 3.8, with randomised network coding. One might pad the streams with zeros instead of applying correction codes, and allow random coding to produce ω representations out of k_i source symbols. The scheme assumes that every GF element $x_{i,j}$ in the packet will be encoded consistently with the whole

packet, including global encoding kernel, in the process of re-encoding and relaying information at each hop throughout the network. At the receiver a random code will have been applied to the generation of packets, where the probability of having rank equal to k_i will depend on the field size, and whenever a receiver collects a rank k_i system, it will be able to decode the video up to stream i. This allows receivers with reduced download capacity to partially decode the video, whereas those recovering all packets will be able to decode the full rate.

These approach have overhead arising from mixing information from different layers across the packets. Windowed coding and prioritised transmission trades in the need to allocate rate a-priori for the overhead, as we will discuss in Chapter 5.

3.3 Conclusions

Scalable video coding brings great advantages over fixed rate codecs, allowing seamless switch, transcoding and partial decoding between different video configurations. Most of the coding tools from the previous non-scalable standard have been extended, allowing a deployment flexibility with little effort and especially small decrease of efficiency with respect of single rate encoders (See Appendix A). The exploitation of such property is both at the receiver and at the sender, where a new version of scalable DASH [50], as well as blind prioritised coding, can most exploit the adaptivity of SVC to different situations and differentiated users demands.

Fountain codes have proven to be the most efficient form of erasure protection. As we analyse possible extensions of this model to multi-source coding, network coding, and prioritised coding, a solution which is optimal on all fronts is difficult to identify. We'll argue in Chapter 6 that rateless fountains over $GF(q)$ are still the best solution for our scalable content delivery system presented in Chapter 5.

Chapter 4

A Detection/Deletion Method for Scalable Error Protection

In this Chapter we present a study of end-to-end coding characteristic of the network channel, thus relying upon an abstract model network, referred to as Linear Operator Channel (LOC) or also Matrix Channel. We will describe the Detection/Deletion system, a scheme derived from the FEC-based MDC described in the Chapter 3, which uses the network code to detect errors occurring at any location of the packets, and a traditional block code Forward Error Correction (FEC) to perform erasure correction. We will provide numeric results of error probabilities on exemplary networks.

4.1 Coding and Transmission Model

This section describes the coding and packetisation scheme that aims at detecting and correcting errors on packetised and coded streaming. Let's consider the network as a directed and acyclic graph, with a source node where the scalable data is originated and packets are injected into the network, and a set of receivers $t \in T$. Let's abstract from the network protocols and assume that nodes forward the data according to a predefined routing scheme, and apply a random code to the packets in transit. Although a coherent transmission model easily describes the transfer function of the matrix channel, we don't need to assume a flow path characteristic with packed trees, or synchronised/coherent forwarding taking place at the intermediate nodes. A practical buffering scheme like Chou's coding (See Chapter 2) and randomised coding is sufficient to infer that the algebraic transmission model holds. Additionally we assume that no decoding, partial or complete, is performed at intermediate nodes, as well as not either any error control process, except at the receiver, and that

prepending unitary vectors to the packets at the source allows in-band communication of the overall coding characteristic. Errors occurring on a single link are also propagated according to the transfer characteristic. Therefore we'll make use of the algebraic formulation of the channel as a linear transfer function.

Let's consider an $\omega \times w$ input matrix $\mathbf{X}_F$, consisting of ω packets produced at the source, each packet made of w elements in $GF(q)$. Let's also consider the following matrices: an $\omega \times |E|$ matrix A as the source routing matrix, where $|E|$ is the number of edges in the networks, which expresses the injection and coding of the packets into the outgoing edges from the source, a matrix of edge adjacencies w, and an $|E| \times h_t$ scattering matrix B_t, for receivers $t \in T$. Let's assume that these matrices have non zero values in accordance with the network graph. $\mathbf{Z}$ is a $w \times |E|$ matrix of errors occurring at specific edges. Having w rows, it also expresses at which positions of the packets the errors are injected. The overall transmission characteristic can be expressed as:

$$\mathbf{Y}^T = (\mathbf{X}_F^T A + \mathbf{Z})(I - K)^{-1} B_t, \tag{4.1}$$

Consistently, $\mathbf{Y}$ is an $h_t \times w$ matrix composed by the h_t packets received by sink node t. Let's also assume that A is calculated as follows:

$$A = G * A_I$$

where G is an $\omega \times h$ block coding matrix, and A_I is an $h \times |E|$ routing matrix with only one element equal to 1 on each row, and corresponding to an outgoing edge from the source node. Injecting using a block coding generation matrix makes sure that the symbols at the outgoing edges from the source, i. e.

$$\mathbf{X}_G = G^T \mathbf{X}_F,$$

are coded according to a traditional block code, e. g., a Reed-Solomon code. Eq. (4.1) can be rewritten as:

$$\mathbf{Y}^T = (\mathbf{X}_G^T A_I + \mathbf{Z})(I - K)^{-1} B_t, \tag{4.2}$$

The Detection/Deletion (D/D) system uses NEC code characteristics to detect erroneous symbols along the columns of $\mathbf{Y}$, which are the received codewords, and uses this information to perform erasure decoding along the rows, which also conveniently correspond to the received packets, therefore combining error-control along space and time dimensions.

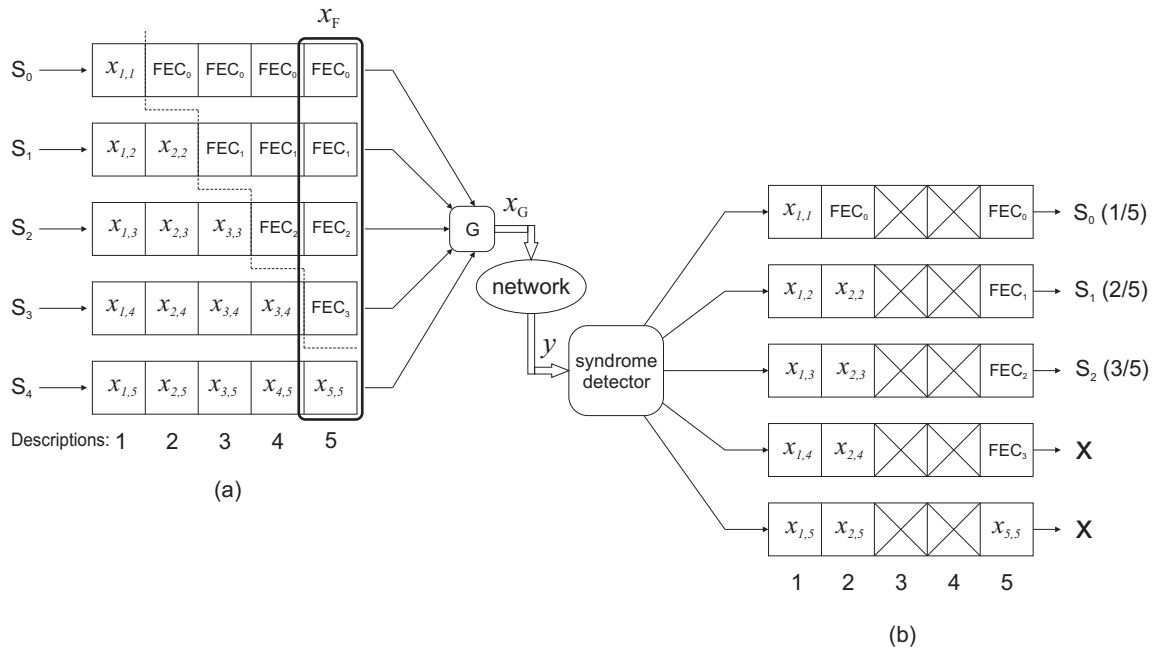

Fig. 4.1: Implementation of Detection/Deletion scheme with Pre-Coding (a), transmission, and scalable decoding (b)

4.1.1 Pre-Coding Scheme

The pre-coding allows to format the source data into a matrix $\mathbf{X}_F$ and block-encode it via G. We defined a generation as a segment of data which is encoded together according to Eq. 4.1. The length of a generation in number of blocks or packets (ω) depends on the packet length w, the size of the generation and the FEC rate applied during precoding.

We identify then with k_i the amount of data for each priority class i of the scalable video expressed in number of $GF(q)$ elements for each generation, with $r_{FEC}^{(i)}$ the coding rate applied to the stream i, and with λ_i the number of packets carrying data of stream S_i. As a result, priority segments of length k_i GF elements are encoded into sequences of $n_i = \lceil k_i / r_{FEC}^{(i)} \rceil$ coded sybmols, and divided across λ_i packets of length w. From this it follows the following allocation of data in the precoding stage scheme:

$$\lceil \frac{\lceil k_i / r_{FEC}^{(i)} \rceil}{\lambda_i} \rceil = w, \quad i = 0, \ldots, L-1. \tag{4.3}$$

This constraint ensures equal length of coded data blocks for efficient packetisation as exemplified in Fig. 4.1 (a). It follows from these last considerations that λ_i is now the number of packets for stream i, including information and redundancy. The Forward Error Correction encoding can be performed by means of any block codes, similarly to traditional FEC-based

Multiple Description Coding described in the last chapter.

The $\omega = \sum \lambda_i$ pre-coded packets in $\mathbf{X}_F$ are at this stage *augmented* to h packets via the generation matrix G, therefore the introduction of G and A_I as illustrated earlier. By producing a Minimum Distance Separable code in the classic sense we ensure that the source network codewords have minimum distance $h - \omega + 1$. However, since the network coding is randomised, the distance property is not automatically preserved along the transmission paths. Still, an MDS generation matrix G and an identity source routing A_I, instead of a completely randomised A, increase the probability of a higher distance at the receiver. Once injected into the network we can see the h packets as being streamed along each one of the h paths that constitute the multicast max-flow from source to receivers. However, the flow paths are not necessary to explain the transmission model, as long as h is a representative quantity of the flow of packets from the source to the receivers, therefore the augmentation can be also seen as an inter-packet FEC coding phase.

4.1.2 Detection/Deletion Decoding

Packetised transmission, coding, and retransmission at intermediate nodes make sure that all packets along the way receive the same network code. As also discussed for the practical coding framework in the Chapter 2, all columns of $\mathbf{Y}$ at the receiver are encoded via the same global kernels. We assume that random errors occur on network edges, and affect the packets travelling in the network by means of the additive factor $\mathbf{Z}(I-K)^{-1}B_t$ in 4.1.

Traditional decoding of a NEC code is based on a complete distance-decoding, i. e., the receiver tries to recover the ω original packets by consistently deducing them out of the h received ones. If the network code is a MDS, the receiver can reliably recover $\mathbf{x}_{F,i}$, in case an error pattern with network weight at most $\lfloor \frac{h-\omega}{2} \rfloor$ has occurred on the error vector $\mathbf{z}_i$. Certainty of having an MDS code comes only from deterministic construction, which is in most cases highly impractical. Additionally, although receivers deduce the transfer characteristic from the prepended coefficients, they cannot always know or deduce the network topology, which makes efficient decoding (e. g., via Maximum Likelihood) difficult if not impossible. Randomised coding implies having randomised distance properties as well, as documented in App. C.

Let's consider each individual linear systems from Eq. 4.1, representing the linear transformation M_t on each codeword:

$$\mathbf{y}_i = M_t^T \mathbf{x}_{F,i}, \quad i = 1, \ldots, w. \tag{4.4}$$

M_t is known from the packet headers, and, by transmitting a number of redundant symbols on each codeword via G, the number of equations h will be greater than the number of variables ω, producing an over-determined linear system. There are two possible situations with an over-determined system: either the solution is still unique, which allows straightforward decoding of $\mathbf{x}_{F,i}$, or there is no solution of $\mathbf{x}_{F,i}$ for given $\mathbf{y}_i$. To better elaborate on the second situation, one could find two sets of ω indices between 1 and h, corresponding to two non over-ranked linear systems of ω equations from Eq. 4.4 giving different solutions of $\mathbf{x}_{F,i}$. While the system is not solvable, this situation allows us to assume that errors have happened on the codeword i, and that all packets are affected by an error at the corresponding position i. We can't say which packets have the error, but we can assume that potentially any of the h packets have it, and more importantly we can precisely identify at which position of the packet the error has occurred. Therefore the network code can detect the position i of the erroneous codewords in the received packets and flag all symbols at the i-th position of all h packets. A syndrome detector is used for this purpose efficiently and with little computational effort [51]. Consistently with classic syndrome decoding, a parity check matrix H_t is built from the system matrix GM_t. The code generator matrix at the receiver t can be written in systematic form as:

$$GM_t = [I_\omega, P]\,. \tag{4.5}$$

A syndrome matrix H_t can be defined as the matrix which, when multiplied by any codeword generated from GM_t, always give a null vector and results in a non-null vector for any other codeword out of the source vector space [52]. Thanks to linearity, any source codeword where additive errors have occurred, will also produce a non-null vector. Such matrix can be written as:

$$H_t = \left[P^T, I_{h-\omega}\right]. \tag{4.6}$$

For all network codewords transmitted without errors it holds that:

$$\mathbf{x}_F^T GM_t \cdot H_t^T = \mathbf{0}^T, \quad \mathbf{x}_F \in \mathscr{C} \tag{4.7}$$

This definition holds for any randomised transfer matrix M_t.

The detection stage in the decoding algorithm consists in checking all the codewords along the received packets, and flagging all codewords with syndrome not equal to $\mathbf{0}$ (Fig. 4.1). Of all the symbols of each packet, those corresponding to correct codewords can be used for erasure decoding.

It follows that on each packet, if l symbols are discarded, priority classes with information rate smaller than $(w-l)/k_i/\lambda_i$ can be decoded from the remaining symbols. We can

also set a threshold, saying that when l errors occur, and there's a pair of indices $i < j$ that verify $w - k_i\lambda_i \geq l > w - k_j\lambda_j$, all priority classes higher than i (i. e. with lower index) can be decoded, whereas the other streams will remain undecoded. In a multirate setting, the sinks can receive differentiated flows h_t, with differentiated detection capabilities.

4.1.3 Errors in the Coefficients

It should be noted that errors occurring in the bits carrying the global encoding kernels may severely affect the decoding performance of the system. Although not tackled in this study, we list a few options that can be considered to overcome the problem:

- One of them is perform a subspace analysis of the coefficients before using the D/D method (about subspace codes see [53–55]). The receiver can calculate which coefficient vector in the h received packets is not consistent with the ones in the other packets, and induces an over-ranked and undetermined system.
- Alternatively one can apply heavy correction codes to the coefficients part of the payload, provided that either the intermediate nodes decode the coefficient header before applying network coding, re-encode and re-append to the payload, or the correction code is a repetition code, in which case applying network coding as described so far will also produce the right effect.

The latter option would increase the payload overhead of few percentage points, but would provide a degree of reliability in recovering the encoding headers.

4.1.4 Network Error Correction Code Properties

The presented technique performs decoding on the temporal dimension jointly with the error detection on the spatial dimension. From theory we know that a network code can detect errors with weight up to $d_{min,t} - 1$ ($W_{\mathbf{z}}^{err} \leq d_{min,t} - 1$). Whether the minimum distance is sufficiently high for the error rate that we expect from the channel, is a random event, since the distance among codewords and therefore the code minimum distance are random variables.

As discussed in App. C, the probability that a random code with redundancy δ_t has minimum distance high enough for an expected degradation of the network is higher with a larger field sizes, and is a more probable event for small values of $d_{\min,t}$ [23]. In other words it's more probable to have $d_{min} = 1$ (enough to have a full rank transfer matrix) than $d_{min} = 2$ (necessary to detect error with weight 1). It comes by itself that it is also less

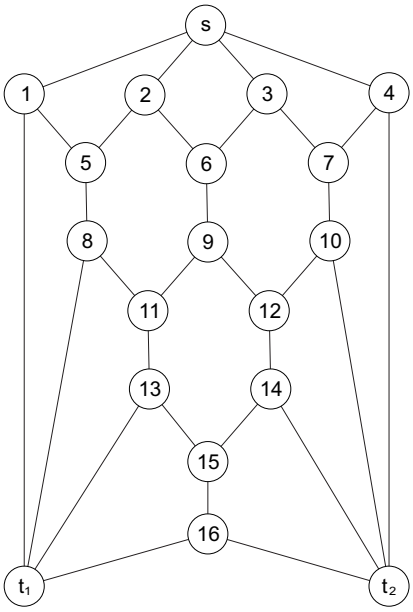

Fig. 4.2: Network topology used for the generation of the channel operator used in our numerical simulations.

probable to have $d_{min} = 3$ (with $\delta = 2$), and so on, due to the increasingly wider separation that the network code is required to keep between the codewords at the receiver. However, for the task of only detecting errors these limitations produce a lower risk than if we had to perform correction of network codewords by means of the NEC code. A distance $d_{min,t} = 2, \forall t \in T$ is required for detecting all errors with weight $= 1$. As pointed out, this value of Hamming distance is more probable than a distance $d_{min,t} = 3, t \in T$ which would be needed for complete decoding from the NEC code.

4.2 Results of Detection/Deletion over the Matrix Channel

In this section we analyse the transmission over a random matrix channel. We give numerical performance of error detection and correction under random channel errors. We generate the channel operator based on the the graph in Fig. 4.2. The graph has 19 nodes, of which one is the source and two are receivers. The edges have all identical bandwidth and the source-receivers max-flow is equal to $h = 4$ times the edge capacity, for both receivers. The flow dimensioning chosen for our sample network channel, is not unrealistic of the number of parallel paths one can find between two peers in any small-medium overlay network. Whilst one can encode several packets on the same paths and increase as such the coding dimension, network error correction was originally formulated on graphs for transmission

spanning multiple paths, therefore short codes arising from this formulation are preferrable to analyse the performance gap between the two systems employing detection and correction via the network code.

We inject errors with knowledge of the topology to emulate an adverse channel condition. However, coding at the source and at the intermediate nodes, and decoding at the receivers are performed without any such knowledge. Injection of errors follows a random process with uniform probability at the edges connecting the intermediate nodes with any other node. We compare the detection/deletion technique with a traditional NEC with maximum likelihood decoding, the first one built to correct all errors $\delta = 1$, and one the second one with $\delta > 1$. In both cases, including the NEC, we construct the code randomly. We also follow the stream partitioning criterion described in Chapter 3. We compare the decoding performance while using different field sizes, information and channel rates.

Detection is performed by means of a random network code with $\delta_t = 1$, for both receivers $t = 1,2$, and erasure correction is supported by a Reed Solomon block code with variable length N. We compare the detection/deletion (D/D) transmission technique with a transmission with a random network error correction code with $\delta_t = 2$. We'll try to correct all single errors at network level with this code. Since the network topology is assumed to be oblivious to the receiver, we apply the three stages syndrome decoding with the NEC code as proposed in [51], and apply traditional RS error correction decoding on the packets afterwards. The packet length length is 512 Kbytes, although RS decoding is performed on blocks of length N, therefore the packet length is not determinant.

The error injection processes on the links are independent and measured on the occurrence of errors at bit level. Equivalently, at the receiver the error rate is calculated on erroneous bits with respect to the original stream. Although coding is performed at symbol level, this allows to normalise and compare the performance among systems that use different field size.

4.2.1 Error Correction Results

Fig. 4.3 shows the error protection performance of a three priority classes stream of data, representative of a scalable video coding data stream. We generated a base stream and the enhancement streams with rates 1.7 and 2.8 times the base layer, respectively. We assigned FEC coding rates $15/31$, $17/31$, and $19/31$, respectively, when using the D/D method, and $28/31$, $26/31$, and $24/31$, respectively, when using the NEC decoding, and use a field size $q = 2^6$. The error slope of D/D in Fig. shows a steeper decrease of errors against the

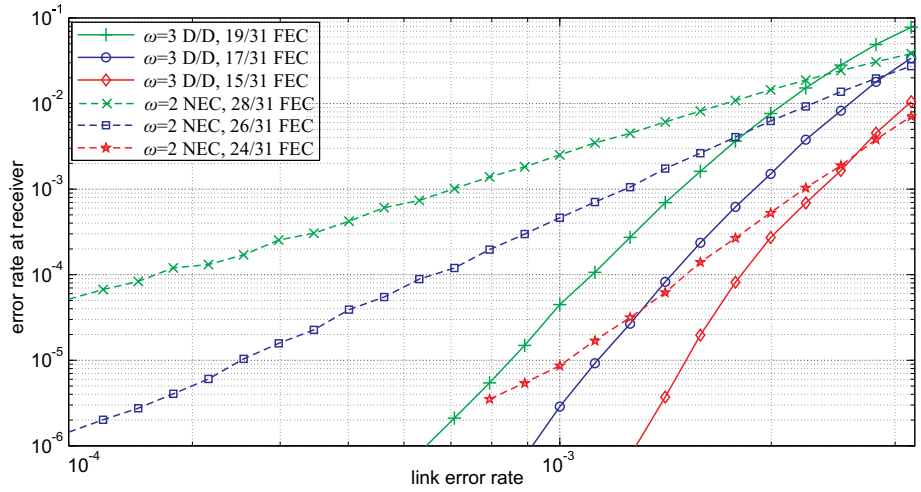

Fig. 4.3: Error rates for the three streams with Detection/Deletion by means of a network code $\delta = 1$, compared with a traditional NEC code with $\delta = 2$.

decrease in link error rate. The difference becomes particularly indicative below $P_e = 10^{-3}$ where we see nearly 2 orders of magnitude of difference. Below this threshold the number of errors approaches the error-free transmission, which is a desirable property for scalable video especially at lower layers. Fig. 4.4 shows the error protection performance of the D/D method and the normal NEC transmission with:

a) fixed field size $q = 2^6$
b) fixed block size $N = 31$
c) two link error probabilities $Pe = 10^{-2.7}$ and $Pe = 10^{-3}$

Fig. 4.5 shows the error protection performance of the D/D method and the normal NEC transmission with:

a) fixed field size $q = 2^6$
b) two different code block lengths $N_1 = 31$ and $N_2 = 63$
c) fixed link error probability $Pe = 10^{-2.7}$

Fig. 4.6 compares instead the error rates resulting from the two methods, under the following circumstances:

a) different field sizes and block lengths: $N_1 = 31$ symbols with $q_1 = 2^6$, and $N_2 = 23$ with $q_2 = 2^8$
b) fixed block size
c) fixed link error probability 10^{-3}.

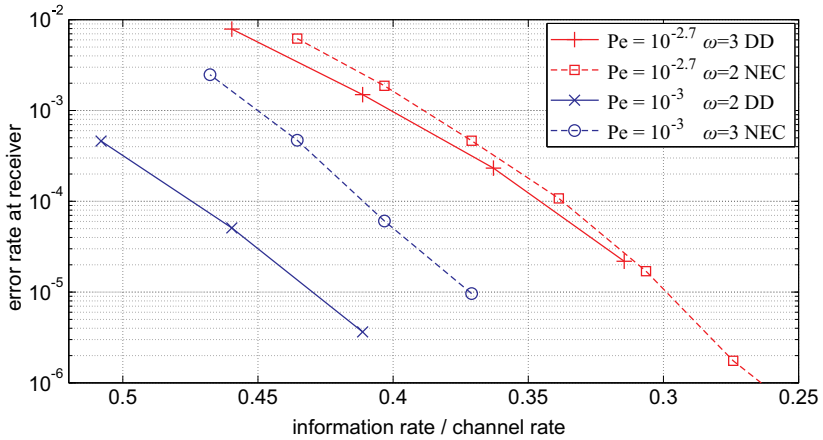

Fig. 4.4: Bit-error rate at the receiver of NEC and D/D method. Block length $N = 31$ symbols. Field size $q = 2^6$. Link error probabilities 10^{-3} and $10^{-2.7}$.

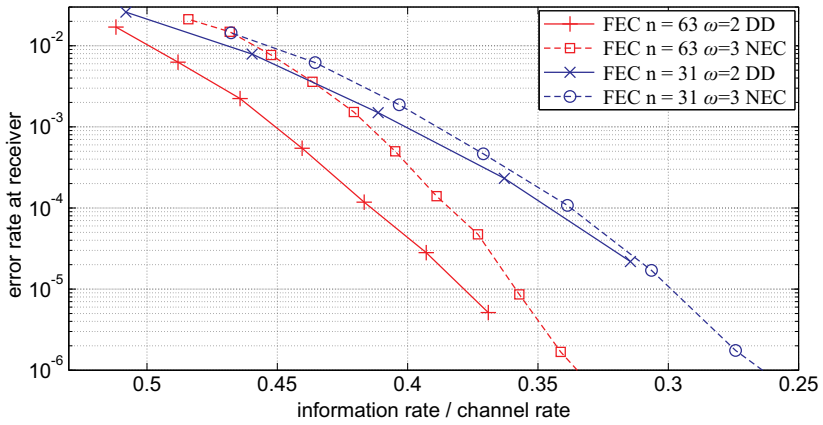

Fig. 4.5: Bit-error rate at the receiver of NEC and D/D method. Block sizes $N_1 = 31$, $N_2 = 63$ symbols. Field size $q = 2^6$. Link error probability $10^{-2.7}$.

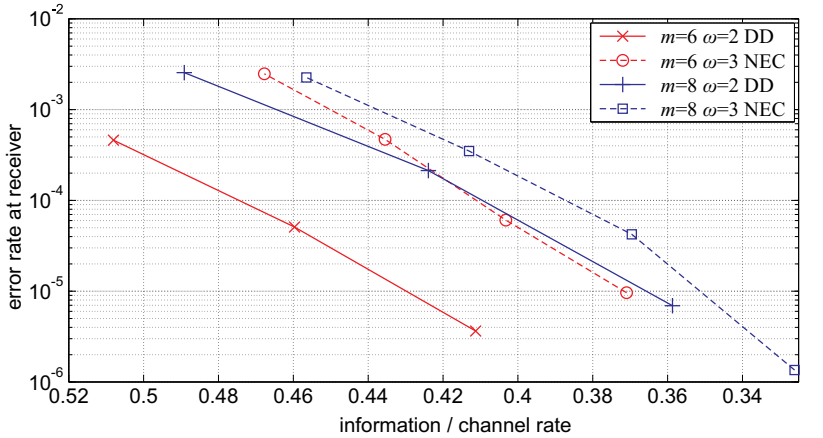

Fig. 4.6: Bit-error rate at the receiver of NEC and D/D method. Block length and field sizes are $N_1 = 31$ symbols with $q_1 = 2^6$, and $N_2 = 23$ with $q_2 = 2^8$. Link error probability 10^{-3}.

The coding rate is indented as the ratio between channel and info rate, considering both erasure coding and NEC code rate. In general, at low error rates the performance of the D/D method is always higher than the traditional NEC for all chosen coding rates, block lengths, and error probabilities. Larger block lengths for the erasure code, in terms of GF elements, as expectable has a better correction performance. The only limitation in this choice is given by computational complexity and mathematical limit for the RS block length. The maximum length of a RS code is in fact $2^m - 1$, which is 63 for $m = 6$ bits, equal to 48 bytes and 255 for $m = 8$ bits. With larger field sizes one can encode larger portions of data, at the expense of a heavier GF algebra.

Using larger field size increases the average distance of the network codewords. This increases the chances of performing better with the NEC code. On the other hand, having the NEC code for correction $1/3$ less rate available for FEC than the FEC for detection, shorter correction codes are used, and since after NEC decoding we perform normal RS error decoding rather than erasure decoding like with D/D, for the tested setting, higher error rates are still registered at the receiver.

Error performance of the two systems are comparable for the higher link error rate of $10^{-2.7}$, whereas for lower error rates, the D/D always performs better in our tested scenarios. We expect for larger block sizes and higher error rates, e. g. larger max-flow, the NEC to perform comparably if not better.

4.2.2 Discussion of Results

The D/D method performs better at low or controllable error rates, whereas, as anticipated, at higher link error rates the performance of the two systems become comparable, eventually yielding to lower error rates of the NEC system. Additionally, D/D also performs better when using smaller field sizes which yields to an overall less computational load. Much less complexity stands on the receiver with the D/D, thanks to both the simple detection method and the erasure decoding as compared to the complete decoding of NEC and block codes in the two separate stages.

4.3 Conclusions

Network Error Correction has been proposed as a way of exploiting a new channel characteristic such as network coding for error resilience. With the shortcomings of the assumption of coherent transmission one cannot fully benefit from such approach yet. However

our detection and erasure decoding is proposed as a valid alternative that exploits a code in the spatial dimension and one in the temporal direction. A better trade-off between redundancy symbols and correction capabilities is provided by jointly decoding network, and channel code, as well as the scalable video, providing a first practical implementation of Joint Source-Channel-Network Coding. This research was presented at one peer-reviewed conference and was afterwards published as a journal paper.

Chapter 5

Peer-assisted Prioritised Delivery of Scalable Video

This chapter presents our content distribution system based on End-System Multicast (ESM). There is an advantage in moving some of the uplink rate from the source to the end users [11]. Users streaming a live content can connect to other peers wich are in turn buffering the same information locally and exchange the data among themselves. An example of peer-assisted delivery is shown in Fig. 5.1. When compared with server-based streaming, the source of data needs to stream a fraction of the overall network demand. When compared to Content Delivery Networks (CDN), no specific infrastructure is employed and the resource necessary to re-upload the data is moved in great part towards the edges of the system.

End-system multicast is often implemented as a Peer-to-Peer (P2P) system. P2P allows decentralised management of resources, assuming that a tracking server only provides information about connected users. P2P additionally provides the means to dynamically create and modify groups of users, keep the state of the group at each peer, and allow them to decide upon establishing connections with other users. We also adopt a data distribution technique closer to P2P techniques, rather than the recent dynamic adaptive streaming approaches adopted by DASH, HDS, HLS, and Smooth Streaming. In the remainder of this chapter the terms *End-system multicast* and *P2P* will be used interchangeably.

We propose here an end-system multicast architecture, making use of a P2P application. Distributed collaborative streaming is achieved with synchronised buffering windows and prioritised network coding with rateless push-based streaming. We introduce a proactive rate estimation algorithm based on node status information, aiming at improving the network usage and increasing the continuity of the service. Video quality is adapted to the available resources, with bandwidth fairness among peers, and minimisation of overhead.

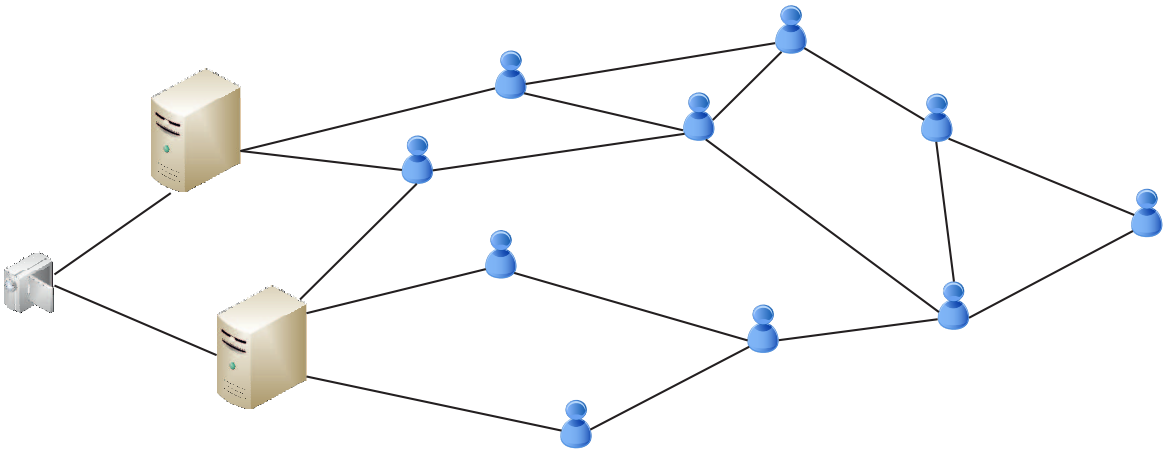

Fig. 5.1: Topology of a P2P television system.

In this chapter we'll use the same notation introduced in the previous chapters. We'll also use the following terminology:

- A *seeder* or *sender* refers to a node in the act or with the potential of sending buffered data.
- A *leecher* or *receiver* addresses a node which is considered as target destination for the data transmitted by a sender.
- The *server* or *source* is the node where the video is originated and is thus no leecher at any time.
- The terms *packet*, *block*, and sometimes *(linear) combination* (of packets or blocks) are used interchangeably when referred to a unit of data being transmitted by a peer, or being received and buffered.

We tested our system in a simulated environment. Results are presented at the end of this Chapter.

5.1 A Prioritised Push-Based System for Live Streaming

End-system multicast based on push strategies is an architecture that has shown great benefit from network coding. In file-sharing systems, encoding and pushing chunks of data to the other peers has proven to be a valid technique to quickly distribute information with small overhead, and with reduced delivery and decoding delays [26]. The file-swarming idea is based on the fact that, due to coding randomness, every packet pushed to a downloading peer is highly probable to be innovative for that peer. Peers can blindly forward

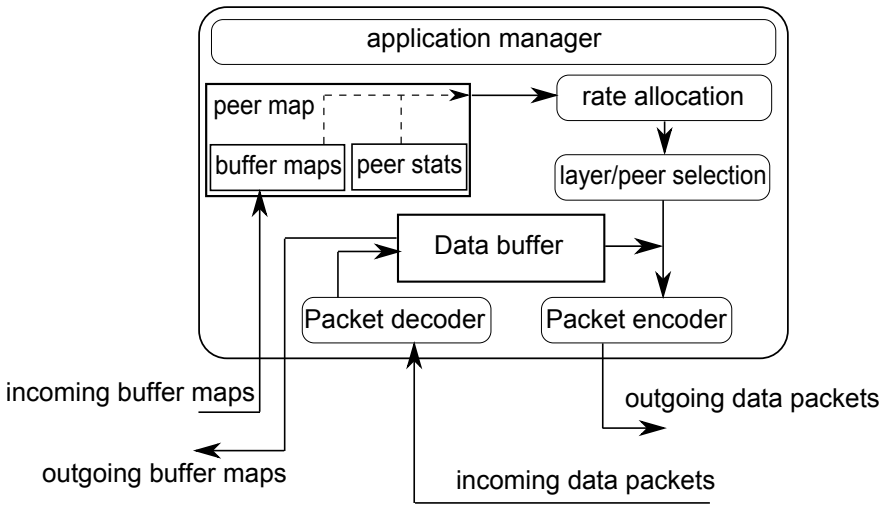

Fig. 5.2: Block diagram of our P2P application, highlighting the main components and functional dependencies.

packets, whereas all nodes in the network can decode the file from any set of received innovative packets. Such system was exposed to the research community under the name of "Avalanche" and more recently it was made available under the name of "Microsoft Secure Content Downloader" (MSCD), highlighting the increased security deriving from the information encoding, and marking the first exploitation of network coding in a commercial software release.

A push-based live-streaming system based on coded packets swarming was proposed in [56]. Such system called R^2 makes use of generation based streaming, introducing the use of a priority region and buffer status messages that allow decoding and consuming the video content at certain time instants, while streaming and retrieving data for the upcoming time instants. The status messages inform the peers in the network of the buffering status of other nodes, which allow the senders to recognise which other users would potentially benefit from retrieving packets from the present node. We developed our push strategies based on R^2-like time-framework, extending the node status information system to support rate estimation and resource allocation for scalable data.

Our streaming system is composed by an overlay of nodes running the application shown in Fig. 5.2. The main parts can be identifiable in: The data buffer, the buffer maps database, the data selection and forwarding component, the rate estimation component.

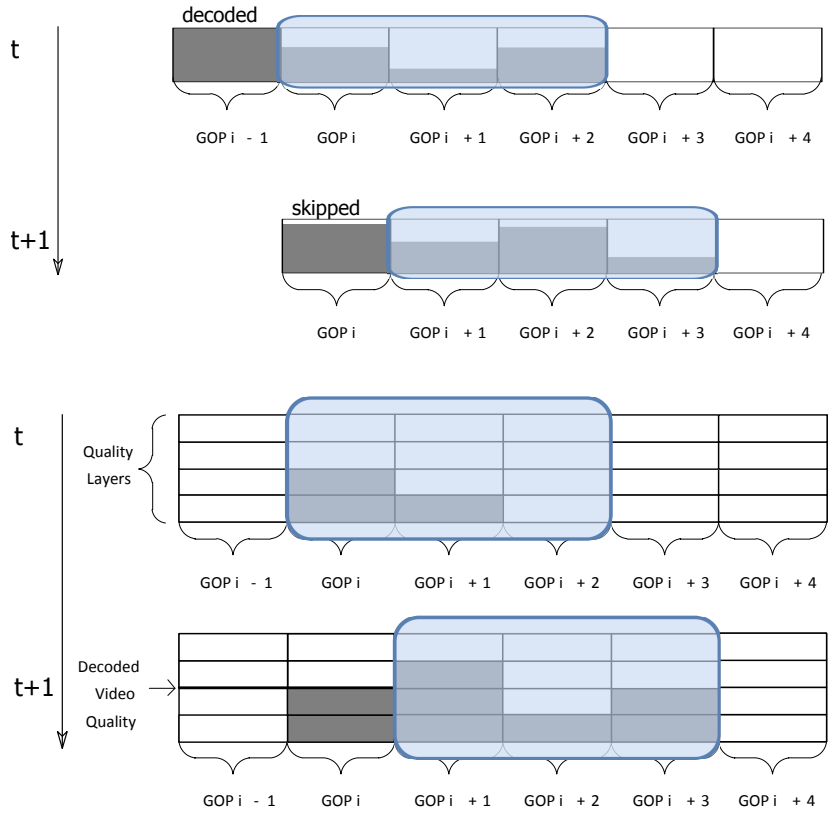

Fig. 5.3: Sliding window with non-scalable data (top) and scalable data (bottom), at two successive instants of time.

5.1.1 Data Exchange and Buffering

The data buffer stores the video before decoding. The video is divided into groups of pictures (GOP) and coding generations (as described later on).

5.1.1.1 Priority Region

A sliding window defines, at each instant of time, a priority region, or playback buffer (Fig. 5.3). Based on a synchronised clock and decoding/playback deadlines defined for every GOP, it is possible for peers to determine at every instant which GOPs are to be buffered and retransmitted. When the deadlines expire the buffered data can be decoded and played back. We define as P_g the playback point of the video, being the last generation of the last GOP that is being played back, and P_l as the length of the window in number of generations, so that at each instant in time the sliding window includes all generations with indices:

$$\mathbf{g}_{p-reg} = [P_g + 1, \ldots, P_g + P_l] \tag{5.1}$$

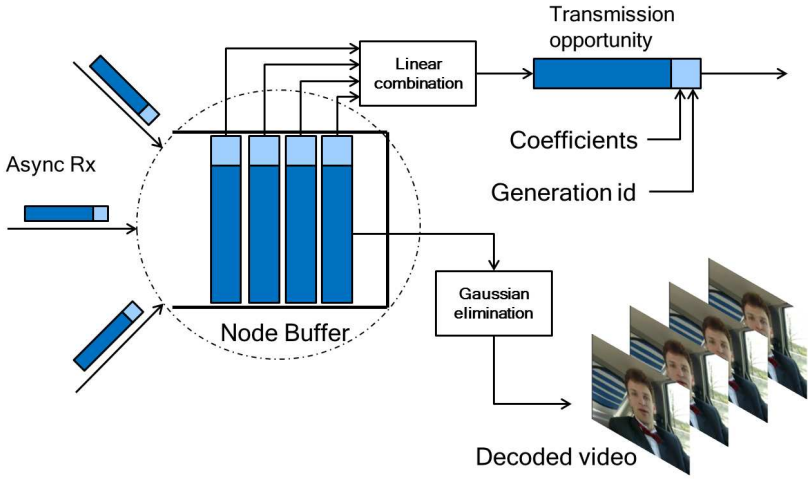

Fig. 5.4: Buffering, decoding and retransmission at peers.

Fig. 5.3 (top) shows a sliding window implementation for non-scalable video. When the deadline for the oldest GOP expires, the window slides forward. If the GOP has been completely received, it can be decoded and played back, otherwise it will be skipped, causing a buffering event. When scalable video is buffered (Fig. 5.3 – bottom), other than received or non received, it can also be partially decodable. When the window slides forward, the GOPs that are popped at the back of the window are decoded at the *maximum* level of quality that has been received. The data in the sliding window is also retransmitted by the seeders by re-encoding and forwarding of new packets. One of the goals of our system is to make sure that as much data as possible is buffered and delivered within the sliding window in order to maximise the quality of the decoded video.

The priority region implicitly defines an allowed playback delay, or buffering delay. Such delay is of the order of seconds, which is compliant to the specification of commercial adaptive streaming protocols (2 seconds for HLS, 10 seconds for Smooth Streaming). The buffer size also determines the occupation in the device memory, ranging from 400-700 KBytes for 2 seconds of HD and full HD H.264 video, respectively, to 2-3.5 Mbytes for 10 seconds of the same videos (See Table 6.1).

5.1.1.2 Packet Encoding

Fig. 5.4 highlights the operations performed by an active peer:

- Asynchronous reception of packets and buffering.
- Decoding via gaussian elimination and playback.
- Retransmission via random linear coding.

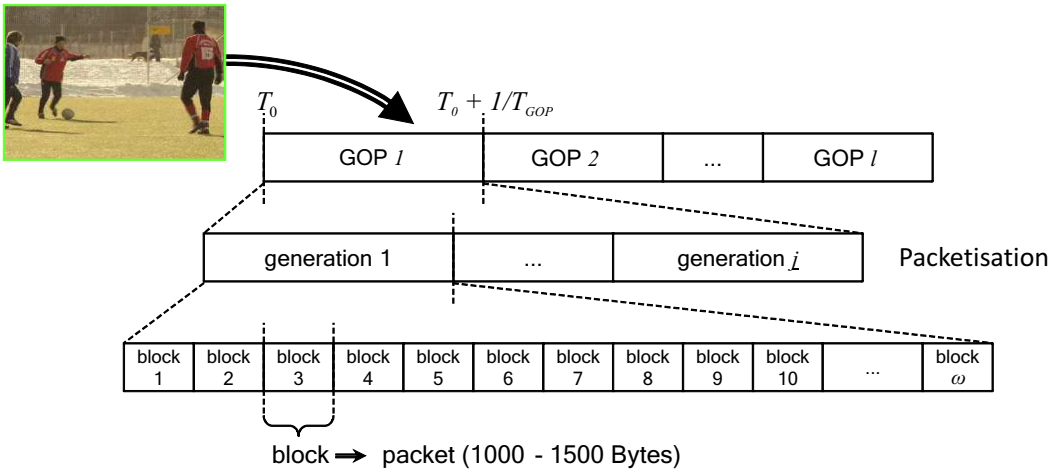

Fig. 5.5: Segmentation of video from GOPs to data blocks.

When a transmission opportunity arises the peer encodes a packet from the buffer with the aforementioned scheme and sends it to a chosen destination. The vector of coding coefficients and the generation ID are embedded in the packet header, to allow other receivers to build a matrix with the respective encoding vectors and retrieve the source data. Each time a new packet is added to the buffer, Gaussian elimination is performed. The coefficients of the buffered blocks will be reduced to row-echelon, which also allows to partially decode the scalable data.

Our network coding system as well as many other practical systems uses generation-based buffering and coding [3]. Fig. 5.5 shows the segmentation hierarchy from GOP level to data blocks. NAL units are grouped into non-overlapping segments of data, called *generations*. Each generation is then divided in blocks, which are combined to produce outgoing packets. In this chapter we assume that the coding field has size 2^8, so we define a packet length as w Bytes. Having chosen generation size of S and a packet size to accommodate blocks in User Datagrams Protocol (UDP) packets, a generation can be segmented into $\omega = \lceil S/w \rceil$ blocks.

The division into GOPs/generations is functional to the encoding process on the outgoing packets. One packet is generated by combining the the blocks belonging to one generation, with some coefficients. In other words, considering a generation composed of blocks $\mathbf{b}_j, j = 1, \ldots, \omega$, a new encoded packet can be written as:

$$\mathbf{d} = \sum_{j=1}^{\omega} \mathbf{b}_j c_j \tag{5.2}$$

Algebraic operations are performed in a Galois Field of size $q = 2^m$ ($GF(q)$), with $m = 8$, and $c_j \in GF(q), j = 1, \ldots, \omega$ are the random coefficients generated for the specific outgoing packet. Randomised dense coefficients are used for a number of reasons. As anticipated in Chapter 3 and discussed in details in Chapter 6, random rateless dense coding has advantages over sparse degree-distribution driven codes. The following characteristic are the strong points of our architecture:

Collaborative coding allows receiving data from multiple sources. While state the art fountain codes, such as Raptor codes, and the predecessors LT and Tornado codes, use binary coefficients to reduce the decoding complexity (Raptor codes have linear decoding complexity), high efficiency in collaborative systems is ensured by coding diversity. Dense coefficients in $GF(q)$, and possibly a large field size q, make sure that randomly encoded blocks are received as innovative most of the times by any receiver, regardless the source of the other buffered packets.

In a peer-to-peer environments nodes might be randomly join and leave. Traditional P2P networks are very prone to the "missing chunk" problem, occurring when the all seeders that had cached a specific chunk leave the network, and as a consequence the other nodes will not be able to retrieve it elsewhere. Transmitting in encoded form allows distributing data blocks with equal persistence of all original chunks, virtually eliminating the missing chunk problem. While in both traditional and coded P2P networks node departures suddenly decrease the throughput availability, in our case we consider this case transparently to the actual departure events. As explained in the next Section, the aggregate throughput is, in every instant, taken as an index and measure of the sustainable video rate. This balances out many diverse cases in which a high number of nodes upload at little rate, or few nodes upload at high rate, and includes the case of heterogenous upload rates across the peers. The collaborative and aggregate throughput can be calculated as:

$$R_{dl}(t_1) = \frac{R_{ul}(s) + \sum_{i=1}^{N_{nodes}} R_{ul}(t_i)}{N_{nodes}}, \tag{5.3}$$

where R_{dl} and R_{ul} are download and upload rates, respectively, of peer t_i, and s is the streaming server. This formula averages out the various contributions, allowing the rate allocation to estimate based on a measurable rate regardless the network situation. A transitory period is however to be taken into account for the algorithm to converge in the event of large changes in the aggregate throughput, i. e., sudden and simultaneous departure of a large number of peers.

Rateless streaming allows any sender to upload an arbitrary number of packets. This idea, coming from fountain codes, allows the sender to adapt the rate of outgoing packets, to accommodate packet losses on the channel, and makes sure that the receiver collects at least the minimum amount of packets necessary for decoding. In our collaborative streaming we could fix the upload rate, and fix a share of this rate for each receiver, counting on the fact that in average other senders will direct their packets towards the same peers. Alternatively there could be a stop condition based on receiver's acknowledgement or status information. In both cases, random coefficients ensure that packets have a statistically high rate of innovativeness, and can be generate in arbitrary number. The rateless streaming ensures that as many packets as possible are sent to overcome packet losses.

5.1.1.3 Prioritised Encoding

Prioritised encoding is performed with the goal of producing packets that carry the information to partially decode video and that can be cumulated with lower priority classes to decode at higher quality. Video data blocks can be classified into L priority classes, based on the layered coding hierarchy, as discussed in Chapter 3. In the remainder we refer to blocks with lower layers as high priority, and vice-versa. The priority segments inside a generation consist of, respectively, $\lambda_0, \lambda_1, \ldots, \lambda_{L-1}$ packets. Packets of class l are constructed by combining blocks corresponding to any class not higher than l. Therefore we can reformulate Eq. (5.2) for prioritised coding via windowing, by setting all coefficients of class lower than l to zero. In other words, priority-encoded blocks can be generated as:

$$\begin{array}{lcll} l=0 & : & \mathbf{d}^{(0)} & = \sum_{j=1}^{\omega_0} \mathbf{b}_j c_j \\ l=1 & : & \mathbf{d}^{(1)} & = \sum_{j=1}^{\omega_1} \mathbf{b}_j c_j \\ & & \vdots & \\ l & : & \mathbf{d}^{(l)} & = \sum_{j=1}^{\omega_l} \mathbf{b}_j c_j \end{array} , \tag{5.4}$$

Where we've used the cumulative number of packets :

$$\begin{array}{ll} \omega_l & = \sum_{i=0}^{l} \lambda_i \\ \omega_{L-1} & \equiv \omega \end{array} . \tag{5.5}$$

Receiving more than or exactly ω_l independent packets, all of class l, allows to decode the portion of the generation with the data blocks containing the data form the base rate segment $l = 0$ up to layer l. Receivers experiencing higher download rates can receive packets of lower priority classes and decode a higher resolution or quality. Using progres-

sive codes, rather than stacked codes [57] is motivated by the fact that greater kernel sizes $(\omega_0, \omega_1, \ldots, \omega_{L-1})$ increases the coding diversity of the overall system. Additionally, a high layer up in the hierarchy is useless if decoded without lower layers. Chunked or stacked codes is however a viable option to decrease the computational cost and is employed in the decomposition of the GOP in multiple generations, also discussed in Chapter 6.

In the next Section we present our rate estimation and allocation framework, which is an essential procedure to allow generation, rate, and peer selection (i. e., chunk/peer selection equivalent of traditional P2P).

5.2 A Collaborative Rate Selection Framework

Our rate selection mechanisms allows receivers to select a video layer with sustainable rate and subscribe to it. This is tightly coupled with generation and peer selection, which on the other hand ensures that random choices performed in a decentralised way are able to statistically collaboratively build up to a throughput that can sustain a level of service.

5.2.1 Peer, Generation and Layer Selection

The chunk selection algorithm is used to allow senders, whenever a transmission opportunity arises, to chose which receiver to send a new packet to, and which generation and priority class to encode the packet from. Such decision is to be supported by a system of information sharing through which the nodes learn about each other buffer status and can determine when a new packet is useful for another peer.

In R^2 the nodes exchange the information regarding which GOPs and generations have been already decoded, so that a sender in a transmission opportunity can independently determine whether receiver t_1 has decoded generation g_1. As far as the chunk/peer picking policy is concerned, the sender would randomly pick among all pairs (t, g) where g is not yet decoded by t. The collaborative nature of the system would statistically fill equally for all required data, reaching an acceptable level of overall efficiency. Without limitation to the approach that we are about to explain, we now present the roulette approach via the pseudo-code in Algorithm 5.1.

When it comes to scalable data, there is to consider different steps at which a generation can be decoded, therefore different classes of packets that can be transmitted to decode at either of the quality layers. We introduce the use of extended buffer maps carrying information on all priority classes received by each node as well as the channel efficiency experience

Algorithm 5.1 Roulette algorithm for generation and peer selection.

```
1: procedure SELECT [t,g](Receivers, Priority Region)
2:     Roulette = {}
3:     for all t ∈ T do
4:         for all g ∈ Priority Region do
5:             if t has not decoded g then
6:                 Roulette ← (t,g)
7:             end if
8:         end for
9:     end for
10:    return random (t,g) ∈ Roulette
11: end procedure
```

by the node. These allow nodes to exchange the following information:

- Detailed information about the buffer status, i. e., for each generation g in the priority region: the number $N_{inn,l}^{(g)}$ innovative packets received for layer $l = 0,\ldots,L-1$. Where $N_{inn,l}$ is the number of innovative packets received for layer l.
- Subscription to a specific layer for each generation, depending on the estimated sustainable rate (see next Section).

An example of buffers map recorded at a receiver node is shown in Fig. 5.6.

Before

	L0	L1	L2	
i	5	4	1	Layer 1 completely decodable
i+1	5	2	0	Layer 0 completely decodable
i+2	3	0	0	no decodable video

After

	L0	L1	L2	
i	5	4	1	Layer 1 **decoded**
i+1	5	4	3	Layer 2 completely decodable
i+2	5	3	0	Layer 0 completely decodable
i+1	5	4	1	Layer 1 completely decodable

Fig. 5.6: Example of buffer map in two instants of time, like in Fig 5.3. N. of packets in this example are 5 for L_0, 4 for L_1, and 3 for L_2.

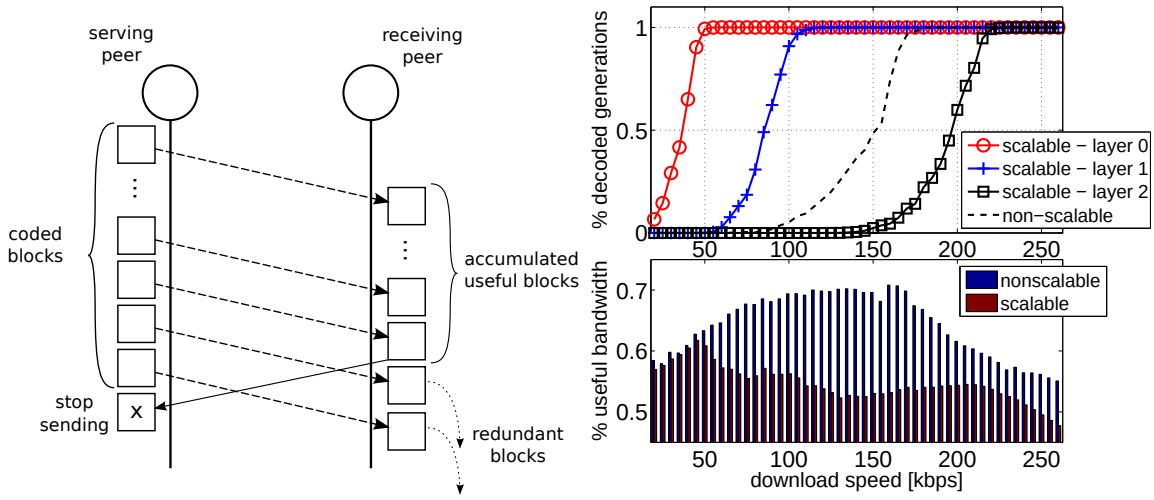

Fig. 5.7: Exchange of packets with layer switching after acknowledgement and braking overhead (left), and performance on a small network (right).

Buffer maps are exchanged at regular time intervals as well as upon complete decoding of a segment, allowing updating the status information as soon as possible. By collecting this information from the other nodes, each peer can build its own status information map and it can determine which layers have already been decoded by a particular peer, of all the generations in the priority region.

5.2.1.1 A conservative allocation approach

In a preliminary study of this system we experimented with senders delivering lower layers first, and making sure that these are received and decoded before proceeding to forward higher layers. While this technique allowed us to delivery partially decodable video under reduced bandwidth, it also produced large overhead. The so-called *braking* overhead occurs when the stream of packets is forwarded ratelessly, i. e. without a predetermined information or coding rate. All non-innovative packets that are sent to a receiver t between the instant the last innovative packet is sent from any seeder to t, and the time instant in which the buffer map is received by the senders and they update the status information, build up to *braking* overhead, as exemplified in Fig. 5.7. This quantity can have a considerable variance, and it's statistically influenced by:

- End-to-end delay between nodes: shorter transit times avoid large overheads.
- Size of the priority region: longer sliding window reduces the chances of selecting a generation and layer which is already decoded by the chosen receiver.

- Size of the peers group: larger number of peers to choose from reduces the chances of selecting a generation and layer which is already decoded by the chosen receiver.
- Congestion in receiving the acknowledgement messages.

While braking overhead was present in R^2 as well, introducing the layer hierarchy has amplified its effect. On the other hand, assuming a loss-prone network, overhead is to be introduced anyway to combat the erasure of packets. The main difference with adding redundancy in point-to-point transmission is that in the first case the overhead is determined by a FEC protection rate based on the estimated loss rate, whereas in our case the overhead is determined by the packets transmitted during the lapse of time required to receive the acknowledgment of decoding.

In the remainder of the chapter we present the rate estimation as the main solution to tackle both partial decoding and overhead reduction. In order to increase the chance of delivering independent packets, we allow the nodes to:

- Record the number of packets sent to each peer. Once the combinations sent to a specific node are equal to the number of buffered packets (retrieved via node status messages) the chances of an extra random packet being innovative are extremely low. The seeder will stops serving this node until its buffer is updated with a packet from another seeder. This enforces the distribution of only innovative packets. Measuring the delivery rate is important on lossy links.
- Record the number of packets received from every peer. We assume that the blocks buffered from a peer should not be sent back, so we allow to send only an amount of packets equal to the number of blocks buffered from nodes other than the potential destination.
- Forward packets only once enough blocks have been received. Coding from a small number of packets reduces the coding diversity. Buffered generations that can already ensure a higher diversity are distributed first.
- Monitor the received video rate and register the number of innovative and non-innovative packets received for each generation and layer to perform rate selection.

While we preserve the buffer map and push strategy, our rate estimation aims at cutting the braking overhead arising from progressive increasing the quality by targeting specific layers for each GOP and its generations. Receivers will deliver the necessary information to make sure a uniform random choice of all subscribed layers for all generations and receivers achieves the best performance.

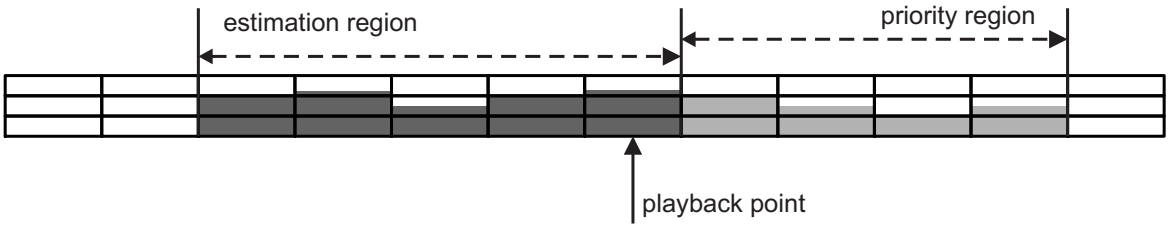

Fig. 5.8: Estimation window

5.2.2 Water Filling for Rate Allocation

Layer and rate selection is based on information dynamically collected by each node and aims at estimating a sustainable download rate and perform an appropriate allocation of video layers for the upcoming buffering interval.

Traditional point-to-point probing techniques or even more sophisticated multi-path probabilistic techniques are either unfit or overcomplicated for the purpose. We rely instead on a simple analysis of the rate received by the video decoder in the time instants preceeding the buffered video. For this purpose we define an *estimation interval* extending over a period of time before and including the playback point and we measure the average total packet rate as:

$$R_{tp} = \frac{1}{E_l} \sum_{g=P_g-E_l+1}^{P_g} \sum_{l=0}^{L-1} N_{recv,l}^{(g)} = \frac{1}{E_l} \sum_{g=P_g-E_l+1}^{P_g} \left[\sum_{l=0}^{L-1} N_{inn,l}^{(g)} + N_{ni,l}^{(g)} \right], \tag{5.6}$$

where E_l is the number of generations in the estimation interval. We consider this as a reliable estimate of the number of packets received by the application, where we can assume that coding and distribution dynamics are implicitly taken into account, and include both innovative and non-innovative packets in the calculation. We note the use of $N_{recv,l}^{(g)}$ to refer to the total number of packets of layer l received for generation g (innovative and non innovative), as well as $N_{ni,l}^{(g)}$ for the number of non-innovative packets received for generation g. We can additionally define the average innovative packet rate as:

$$R_{ip} = \frac{1}{E_l} \sum_{g=P_g-E_l+1}^{P_g} \sum_{l=0}^{L-1} N_{inn,l}^{(g)} \tag{5.7}$$

as the average rate of innovative packets calculated over the estimation interval E_l.

The rate allocation can be defined as: *Finding, for each generation in the priority region, a target layer, possibly the closest to* $L-1$*, provided that the collaborative rate that the receivers are expected to sustain is lower than* $R_{tp} * \alpha_{tp}$.

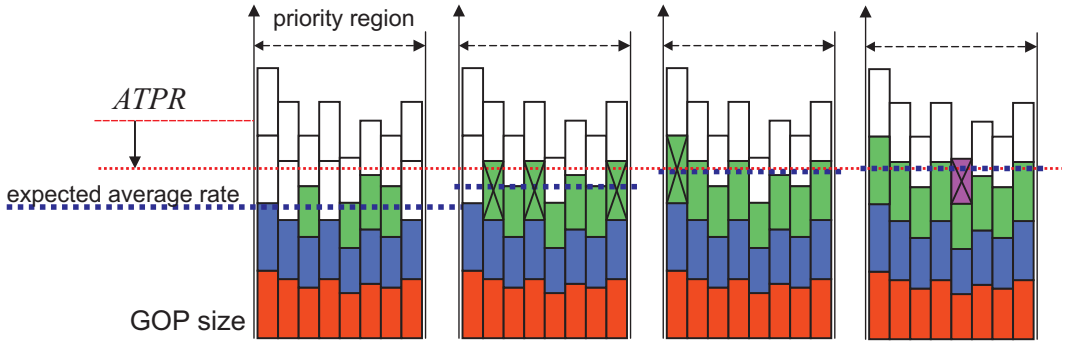

Fig. 5.9: Example of layer selection in the priority region with the water-filling algorithm.

We've now introduced an adaptation factor $0 < \alpha_{tp} \leq 1$ that allows targeting an information rate arbitrarily smaller than the effective rate of packets, in order to compensate for inefficiencies. The rate selection is performed every time the priority window slides forward, i. e., whenever the oldest GOP is popped out, and a new one is inserted.

The procedure for finding $\hat{\mathbf{l}} = [\hat{l}_g, g = P_g + 1, \ldots, P_g + P_l]$ starts from a set of initial values, $\hat{\mathbf{l}}_0$. This inital vector is calculated as the highest values of quality for each GOP for which the number of packets $\omega_l^{(g)}$ is below the total packet rate $R_{tp} * \alpha_{tp}$. Afterwards it proceeds to fill the priority region by increasing the quality layers for each generation, until the $R_{tp} * \alpha_{tp}$ limit is reached, and this is done by prioritising, in turn:

1. Higher classes segments.
2. Segments with less missing packets.
3. Older generations (closer to the deadline).

Fig. 5.10 shows typical stream sizes and PSNR of an H.264/SVC-encoded sequence. Fig. 5.11 also shows the periodic presence of GOPs which are bigger in size, although yielding to approximately the same PSNR of the other GOPs at the same quality layer. The large GOPs correspond to those containing the I-slice, and the periodicity is given by the ratio between I-frame period and GOP length. It should be noted that this is allowed in SVC by the presence of a key-picture in every GOP. However, in order to decode GOPs without I-frame at a higher quality than the neighbouring GOPs in the same I-frame period, one should make sure that the prediction configuration and the coding loops allow that.

The rate allocation algorithm via water-filling in the case of a single generation for each GOP is described by the pseudo-code in Alg. 5.2. The only change when the GOP is divided into multiple generations is to sort the GOP indices and consider all the generations in the GOP, and swap N_{inn} and $\omega^{(g)}$ with the sum over the GOP. An example of the algorithm execution is also shown in Fig. 5.9.

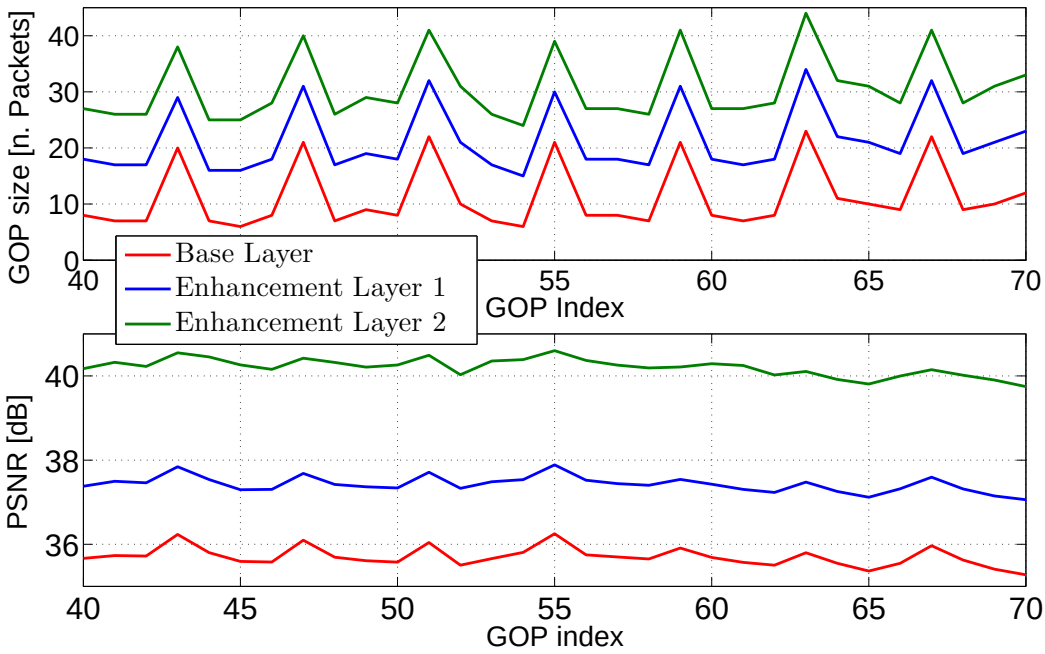

Fig. 5.10: Paris CIF video, H.264/SVC with Medium Grain Scalability: Size of GOPs (top) and PSNR (bottom).

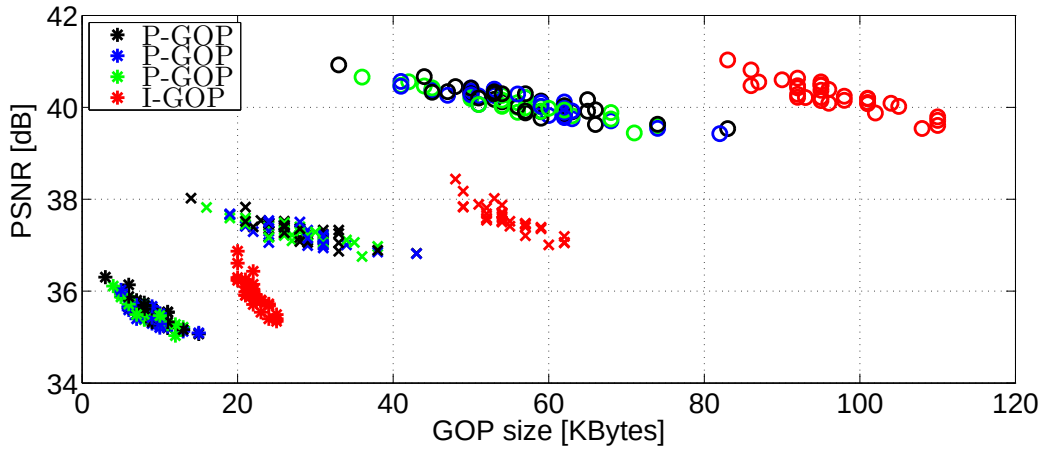

Fig. 5.11: Paris CIF video, H.264/SVC with Medium Grain Scalability: PSNR vs Size of GOPs for different classes of GOPs.

Once a specific layer has been allocated for each GOP in the priority region, the uniform randomised routine can be applied with one possible choice of quality for each GOP for each potential receiver and Alg. 5.1 can be applied trivially.

Algorithm 5.2 Prioritised Water Filling algorithm for single-generation GOPs.

function PRIOREGRATE(**l**)
    return $\frac{1}{P_l}\sum_{k=P_g+1}^{P_g+P_l}\omega_{l_k}^{(k)}$
end function

function SORTBYSIZE($\mathbf{g}=[g_1,\ldots,g_n],l$) ▷ Sort elements of $\mathbf{g}$ by size of $\omega_l^{(g)}$
    $\mathbf{g}_{sort}=[\ g_{j_1},\ldots,g_{j_n}\ \mid\ j_k\in\{1,\ldots,n\},\ k\in\{1,\ldots,n\}\ ]:$
    $N_{inn,l}^{(\mathbf{g}_{sort}(j_k))}-\omega_l^{(\mathbf{g}_{sort}(j_k))}\leq N_{inn,l}^{(\mathbf{g}_{sort}(j_k+1))}-\omega_l^{(\mathbf{g}_{sort}(j_k+1))},\forall j_k\in\{1,\ldots,n-1\}$
    return $\mathbf{g}_{sort}$
end function

procedure WATER FILL($\omega_l^{(g)},R_{tp},\alpha_{tp}$)
    Priority region: $\mathbf{g}_{p-reg}=\left[g_j|j=P_g+1,\ldots,P_g+P_l\right]$
    $\hat{\mathbf{l}}_0=\left[\hat{l}_j=\max_i:i=0,\ldots,L-1\ \text{ s.t. }\ \omega_i^{(g)}<R_{tp}*\alpha_{tp},j=P_g+1,\ldots,P_g+P_l\right]$
    $\hat{\mathbf{l}}_{piv}=\hat{\mathbf{l}}_0$
    for $l=0,\ldots,L-1$ **do**
        Prio. region filled $=false$
        $\mathbf{g}_{sort}=$ SORTBYSIZE $(\mathbf{g}_{p-reg},l)$
        for $g\in\mathbf{g}_{sort}$ **do**
            if (g,l): not decoded **then**
                $\hat{\mathbf{l}}_{\text{temp}}=\hat{\mathbf{l}}_{piv}$
                $\hat{\mathbf{l}}_{\text{temp}}[i]=\mathbf{g}_{sort}[i]$
                if PRIOREGRATE$(\hat{\mathbf{l}}_{\text{temp}})<R_{tp}*\alpha_{tp}$ **then**
                    $\hat{\mathbf{l}}_{piv}=\hat{\mathbf{l}}_{temp}$
                else
                    Prio. region filled $=true$
                end if
            end if
        end for
        if Prio. region filled $==true$ **then**
            break For cycle at the current l
        end if
    end for
    Return $\hat{\mathbf{l}}_0$
end procedure

5.3 Experimental Setup and Results

This section describes the experimental setup prepared to validate our streaming system. We first describe the P2P protocol which the application is based on. We then present the simulation environment and experimental results.

5.3.1 Architecture of our Application

Our application implements a new data delivery system over a P2P protocol described in an unpublished work and called *Zeta*-protocol [58]. ZetaSim extends the GnutellaSim [59] implementation for the network simulator (ns2 [60]) by using its messaging and socket structure, and it implements the Zeta protocol, a file download scheme based on the network coding principles of Avalanche. It uses the packet push scheme, UDP datagram for feedback-free transmission, and progressive decoding with Gaussian elimination.

We modified the ZetaSim suite to perform live streaming of scalable video. We introduced the use of generations, the synchronised sliding window, and prioritised coding, as well as a video data loader and parser for demonstration purposes. Finally we introduce packet forwarding, rate estimation and allocation, as well as the buffer map exchange, based on the prioritised streaming system described earlier in this Chapter.

ZetaSim defines a boot server which incorporates the functions of the web server in P2P protocols like bit-torrent. We incorporate the functions of tracker and boot server. Users who want to retrieve the video content join the network via the boot server. We have then three different type of nodes:

- Boot server/tracker
- Video server peer
- User peer

One node, the server, is the initial source of the live video. Both users and server authenticate to the boot server and stream the buffered video via the same exact procedures. We now describe the main authentication and bootstrap procedures.

P2P protocol procedure: *JOIN* The *JOIN* procedure is shown in Fig 5.12 both for a normal user and for a user with a video to stream (in our setup, the server). When a user joins, it is put by the boot server in the *presence list*, allowing the user to perform further actions. The video server then sends an *UPDATE_STREAMS* message informing the tracker

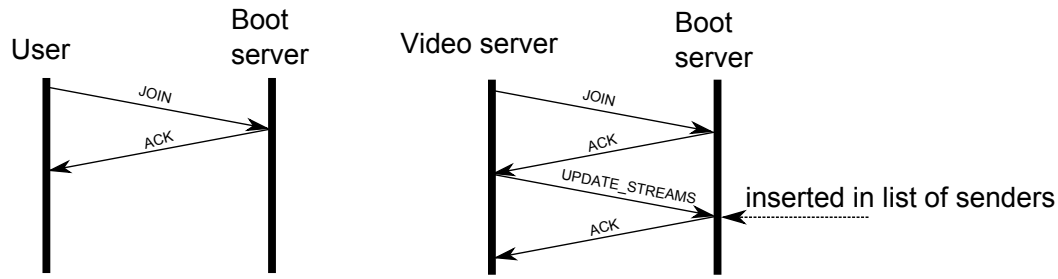

Fig. 5.12: P2P protocol *JOIN* procedure.

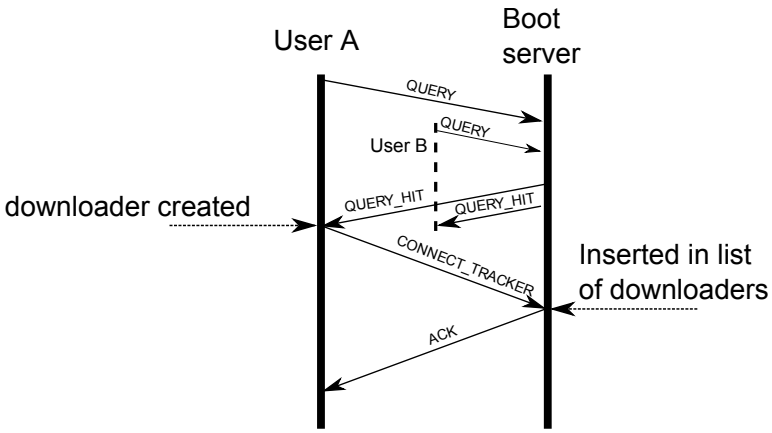

Fig. 5.13: P2P protocol *QUERY* procedure.

entity of the availability of the video to stream, after which the video server node is put in the list of the *SEEDING_PEERS*.

P2P protocol procedure: *QUERY* When a node that has joined the network wants to receive the video, it sends a *QUERY* message to the boot server. The query is either replied by a *QUERY_HIT* message, or it is put in the list of unfulfilled queries (in case there are no seeders for the requested video) and is replied at a later stage once seeders become available. The user will then create the downloader and inform the tracker, which will put the user in the list of downloaders. This procedure is described in Fig 5.13.

P2P protocol procedure: *START* Fig 5.14 show the stream bootstrap, where all peers in the network are notified of the lits of downloaders in the network and the video server is explicitly triggered to start the stream. In our system we concentrate on the live streaming of the video, so we perform the procedures of joining the network and exchanging the list of shared videos upfront the streaming simulation. Downloaders create in turn their seeding functionality once the seeding buffering threshold has been reached.

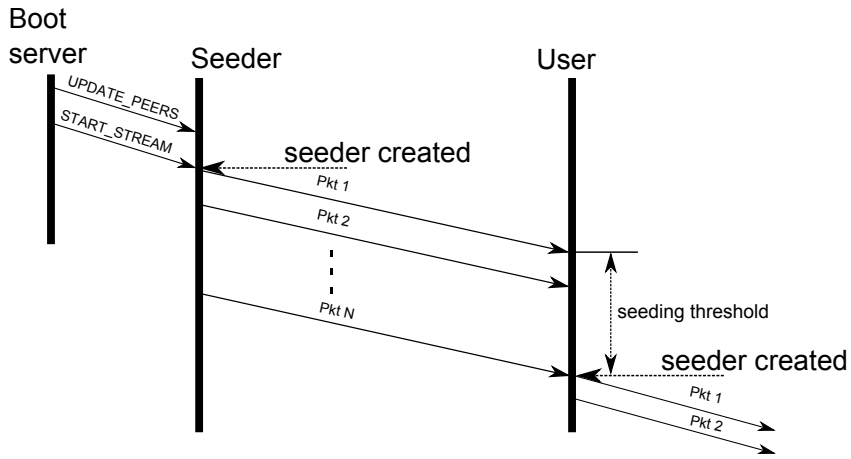

Fig. 5.14: P2P protocol *START* procedure.

5.3.2 Network Generation

For our experimental setup we generate a number of random topologies where we place an application at each node. We make use of a geographic model for network growth emulation, proposed as the Waxman model [61] and whose software implementation is available at [62]. The generation allows the creation of a flat plane, where a number of nodes are placed according to a 2D Poisson process. Links are then created between any pair of nodes u and v, with probability:

$$P(u,v) = ae^{-\frac{d(u,v)}{bL}} \tag{5.8}$$

where $a > 0$ and $b \leq 1$, d is the distance between u and v, and L is the maximum distance between any two nodes (given by the plane size). The parameter a drives the probability of having a link between any pair of nodes, whereas the parameter b increases the probability of having long edges with respect to the probability of having links between closer nodes. Fig. 5.15 shows a few examples of randomly generated networks, the top line showing 20 nodes networks, and the bottom line showing a 50 nodes network, as used in our experiments.

5.3.3 Simulation Results

In this section we present the simulation results of the proposed system.

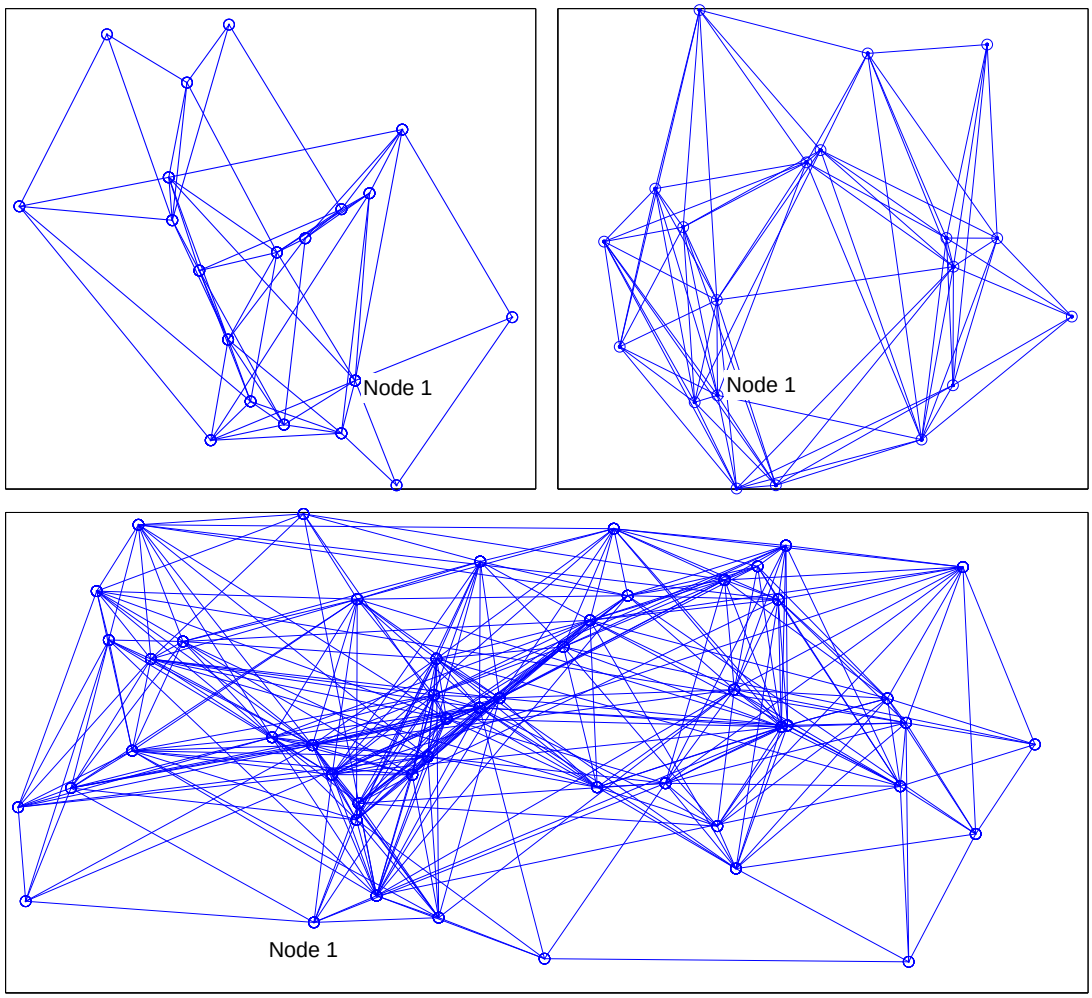

Fig. 5.15: Randomly generated network topologies. Top: 20 nodes; left: $b = 0.1$, $a = 5$; right: $b = 0.1$, $a = 10$; Bottom: 50 nodes.

5.3.3.1 Setup

In our scenario, we have a single source node, streaming a video encoded with H.264/SVC using Medium Grain Scalability (MGS) at CIF resolution. The source node is the only node that is not consuming the video. We make use of the Joint Scalable Video Model (JSVM) reference software [63]. Three priority classes are selected, exploiting quality scalability, where the base layer has QP equal to 30 and the enhancement layer has QP equal to 22 and MGS vector partitions equal to 6 and 10, for enhancement layer 1 and 2, respectively. We use the Paris sequence with CIF spatial resolution (352×288) and 30 frames per second. The compressed video streams account for an average bit rates and PSNR of:

	bitrate [KBps]	PSNR [dB]
Base layer	41.6881	35.6259
Enhancement layer 1	78.7615	37.3439
Enhancement layer 2	113.7042	40.1305

We use an I-frame period of 32 frames (corresponding to 1.067 seconds of video) and a GOP size of 8 frames (equal to 0.2667 seconds of video). The video GOPs have PSNR and sizes shown in Fig. 5.10, and every GOP is composed of a single generation. Multi-generation GOPs are treated in Chapter 6.

We generate a random network topology, with a fixed number of nodes. We let all the nodes join the network and get the list of peers, before starting the streaming from the video server. The peers are let download at a typical DSL speed, i.e., 2 Mbps, which is high enough to be considered not a bottleneck. The normal peers also upload at a variable rate between 80% to 130% of the full rate video. The video server will stream to all authenticated peers at an upload rate which is higher than the other peers, and changes between different experiments. Additionally, different values of the guard factor for the water filling allocation threshold are tested, ranging from 0.7% to 0.95%, to allow variable overhead. We also set the priority region to be 8 GOPs long, corresponding to 2.133 planned buffering delay. This also corresponds to the specifications of HLS. The estimation interval is also set to the same lenght.

5.3.3.2 Performance Measures

We collect rate and playback statistics the receiver ends. We record the arrival of non-innovative and innovative packets and decoding events of innovative packets, and the complete decoding of the GOPs. We then measures the following quantities:

Average PSNR The Peak Signal-to-Noise Ratio (PSNR) is defined as [52]:

$$PSNR = 20 \cdot \log_{10} \left(\frac{\mathrm{MAX}_I}{\sqrt{MSE}} \right), \tag{5.9}$$

where $I(m,n)$ is the intensity of the luminance components of the pixel with m,n coordinates, and MAX_I is the maximum value such intensity can take. Having 8 bits per component depth coding, the intensity max value is 255. The Mean Squared Error (MSE) is

calculated as follows:

$$MSE = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \left[\hat{I}(m,n) - I(m,n) \right]^2 . \tag{5.10}$$

While the PSNR can be calculated also for the chrominance components, our only PSNR measures will be taken only on the luminance component. The PSNR is calculated over the frames of the received video GOPs at all receivers, and then averaged in the time domain over the interval of observed video.

Playback skip rate A playback skip occurs when the decoding of a GOP is missed at any quality layer. The received video is observed during a time window of duration T, equal to an observation of $N_{GOPs}(obs) = \lfloor T * (GOP\_period)/(frame\_rate) \rfloor$ GOPs. The playback skip rate is calculated as:

$$\mu_{skip} = 1 - \frac{1}{N_{GOPs}(obs)} \sum_{g=g_0}^{N_{GOPs}(obs)} \mathrm{dec}(g,0) \tag{5.11}$$

where $\mathrm{dec}(g,l)$ returns 1 if generation g has been decoded at least at quality l, 0 otherwise. The skip rate is calculated on the number of skipped GOPs, not on the skipping events, i. e. we consider a number of consecutive skipped GOPs contributing individually.

Coding Efficiency, Goodput Efficiency The *coding* efficiency is calculated as the fraction of innovative packets over all the received packets. It is an index of coding diversity and overhead that is introduced by non-innovative packets, and it is calculated as:

$$\eta_{cod} = \int_{t_0}^{T} \frac{inn(t)}{recv(t)} \, \mathrm{d}t \tag{5.12}$$

where $(t_0, T]$ is the interval under observation for packet arrival events, and $inn(t)$ and $recv$ are functions which are in the form $\sum_{t_{arrival}} \delta(t - t_{arrival})$ ($\delta(t)$ is Dirac's function). $t_{arrival}$ are the arrival times of packets (innovative or non innovative). These can also be expressed as the sum of packets received for each generation over the period of time:

$$\eta_{cod} = \sum_{g=g_0}^{N_{GOPs}(obs)} \frac{N_{inn}^{(g)}}{N_{recv}^{(g)}} \tag{5.13}$$

While this is calculated over all innovative packets, the *goodput* efficiency is calculated over the innovative packets received for the layer that is finally decoded, disregarding innovative packets received for higher and non-decodable layers as overhead. The goodput efficiency is calculated as:

$$\eta_{gp} = \sum_{g=g_0}^{G} \frac{\sum_{i=0}^{l_{decoded}(\mathrm{GOP}(g))} N_{inn,i}^{(g)}}{N_{recv}^{(g)}} \tag{5.14}$$

where $\mathrm{GOP}(g)$ is the GOP generation g belongs to, and $l_{decoded}(\mathrm{GOP}(g))$ is the layer at which generation $\mathrm{GOP}(g)$ has been decoded, which is the minimum among all generations in the GOP, and could be smaller than 0 if the GOP is not decoded at all.

Of all these indicators, global measures are obtained by averaging the calculated values over all the receivers in the networks.

5.3.3.3 Numerical Results

Streaming is simulated on topologies shown in Fig. 5.15. The download rates for peers are set to be twice the full video rate, in order to allow receiving the full rate video. The upload rates are limited, in order to evaluate how much resources the users need make available to sustain the streaming, when the server uploads at a fixed rate. The tests are performed on the event-based network simulator (ns2) and we analyse separately the performance against the nodes upload rate, and against the adaptation overhead. In the remainder we refer to the estimation as *tight* for values of α_{tp} close to 100%, and *loose* otherwise. Performance is also compared to a non prioritised system, streaming the full rate video, similarly to R^2. Skipped GOPs are computed with PSNR equal to zero. The observations are taken after allowing an initial buffering time of 10 seconds. We show simulation results for a 20 nodes network, where we've set the server upload rate to 700 KBytes/sec, and a 50 nodes cluster, where the video source uploads at 1 MByte/sec. When compared to the network demand, these server uploads rates are approximately around one third and one fifth of the uplink capacity required in a normal client-server system, respectively.

Average PSNR Fig. 5.16 shows the average PSNR experienced by a group of 20 users streaming the Paris CIF video. Fig. 5.17 shows the average PSNR experienced by a group of 50 users, also streaming the Paris CIF video. The dashed line in both figures shows the PSNR with non prioritised streaming, whereas the solid line shows the PSNR of the prioritised system. For the non-scalable system, reaching the PSNR of the full-rate video is index of good reception. Degradation of PSNR is sign of missed reception and skipping of GOPs. In the 20 nodes network, the full-rate video is delivered with high probability (100%

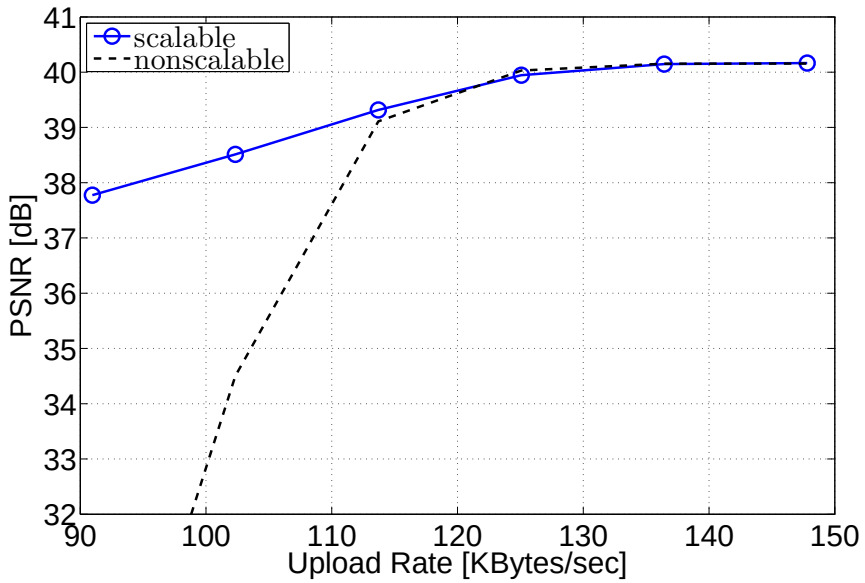

Fig. 5.16: Average PSNR vs upload rate. Network size: 20 nodes, server upload: 700 KBytes/sec, Paris CIF sequence.

of receivers) when the nodes upload at around the same rate of the video, whereas in the 50 nodes network, where the server uploads at 1 MByte/sec, we notice that the nodes need to upload at 120% extra rate to achieve the same results. In general we expect the nodes to have to put more of their own resources to achieve the same result if the server reduces its contribution, meaning a shift of the curves towards higher values of the x-axis, and vice-versa. This is also true when the number of nodes increases, meaning in general a higher need for resources either from server or users.

With rate estimation we notice how the quality of the video degrades gracefully, but does not drop, indicating that the senders will try to adapt the quality to a lower rate. For upload rates around 70% of the full rate video (around 90 KBytes/sec), in the tests over the small network, and 80% in the big network, a PSNR of around 37-38 dB means that in average the nodes are streaming the video mostly at the first layer of quality.

Fig. 5.18 shows the variation of PSNR with the adaptation factor α_{tp} in the 20 nodes cluster when the server uploads at 700 KBytes/sec. These statistics allow us to evaluate the impact of a thight estimation against one allowing room for overhead. A tighter estimation works better for higher upload rates, when the bandwidth is close or higher than the full-rate video. However, at lower rates, the tight estimation does not take into account the higher overhead generated by this data size at this uplink rate. The estimation of sustainable rate is too tight for the goodput that can be delivered reliably, and it fails to deliver at a good

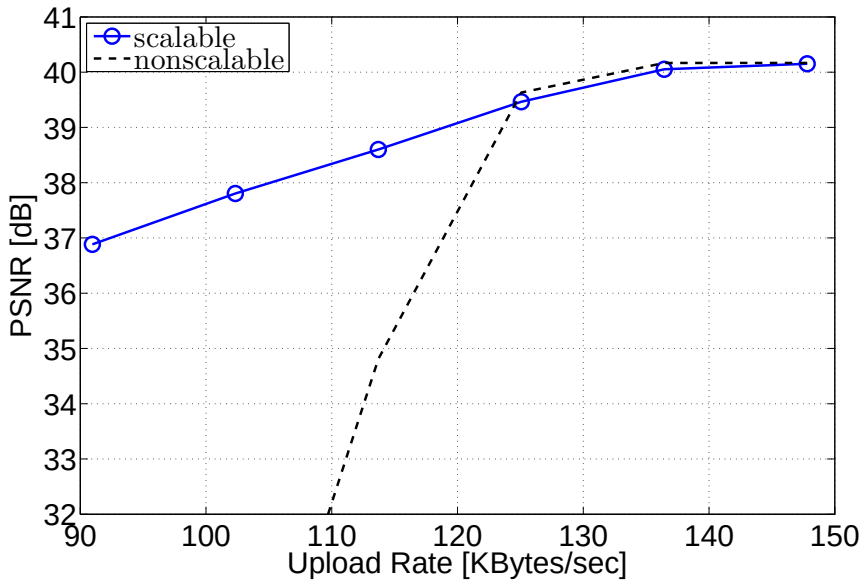

Fig. 5.17: Average PSNR vs upload rate. Network size: 50 nodes, server upload: 1 MBytes/sec, Paris CIF sequence.

PSNR. By transmitting at a rate that is not sustainable, the receivers will fail to complete the decoding of any generation at the allocated quality layer, yielding to high skipping rates. At low rates in fact, a looser estimation gives a higher PSNR, and also yields to a lower skipping rate. This means that a looser estimation of the goodput, and consequently of the overhead, is more accurate to be around 17.5% at lower rates in the 20 nodes case. Conveniently, this estimate is acceptably close also at higher rates, although not as tight as the 90% factor. Fig. 5.18 shows how the PSNR variates with the overhead, showing that, at an upload rate of 80% of the full rate, the highest quality is delivered with $\alpha_{tp} = 87.5\%$. Similar curves have been found for the 50 nodes network, where the peak is only slightly pushed towards higher values of α_{tp}.

Playback skip rate Shown in Fig. 5.19, 5.20, and 5.21 are also the skipping rate and delivery ratios experienced in the 20 nodes network experiment with server upload rate equal to 700 KBytes/sec. In the first one we can see again how the playback skips in a non-scalable system become unsustainable when the video rate is not enough, and how the prioritised system keeps instead the playback skips below 1%. Shown in Fig. 5.20 is the trend of the skips at two different upload rates, confirming that, with limited bandwidth, a relatively loose estimation yields to lower playback skips than the tighter systems (85%, 95%). A tight estimation only succeeds with plenty of bandwidth, and diverges otherwise.

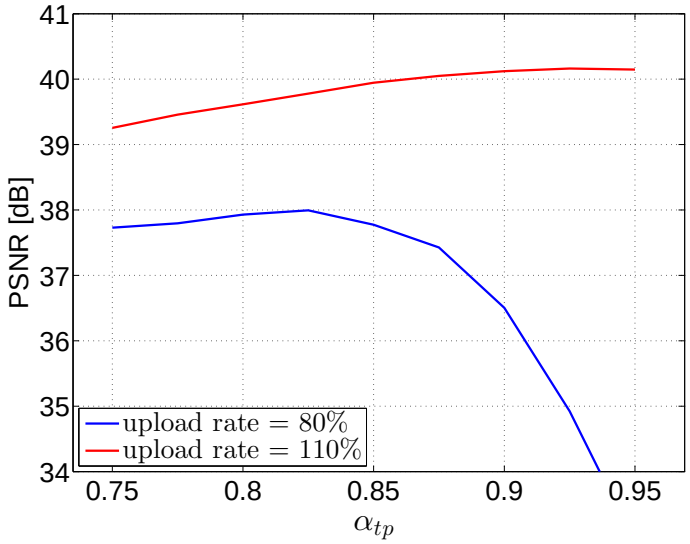

Fig. 5.18: Average PSNR vs α_{tp} at two different node upload rates. Server upload: 700 KBytes/sec. Network size: 20 nodes, Paris CIF sequence.

Finally in Fig. 5.21, we can see the delivery rate of the three quality layers, i. e. the fraction of decoded generations at a specific layer. A neater separation is obtained with $\alpha_{tp} = 75\%$ between base rate and full rate, as well as fewer resources allocated for the first enhancement layer with reduced bandwidth, meaning in average a better allocation of resources towards the basic stream. In the right figure the first enhancement layer is to be included more often by the rate allocation.

Transmission efficiency Finally, Fig. 5.22 shows the transmission efficiency achieved by the push strategy. The left graphs show the efficiency when the nodes upload below the full-rate video, the right graph when they upload above the video rate. Although the coding efficiency is in general higher when the rate estimation is tight, such efficiency does not always translate all into useful video rate, or goodput. One can at this point note a particular trend: When the estimation exceeds the sustainable goodput, i. e., the overhead is underestimated, the nodes will try to upload at a rate that exceeds the sustainable goodput. Let's consider for example the point where where the coding efficiency appears to be at his highest (left graph, $\alpha > 0.9$). At this exact point, the goodput efficiency, shown in the right graph, is instead rather low, due the channel not being able to reliably sustain the delivery of the entire generations. With these results we can conclude that an estimation of 12.5% overhead on the small network and 10% in the medium one achieve the best efficiency at low rates, as well as a satisfactory closeness to the full-rate when the channel allows.

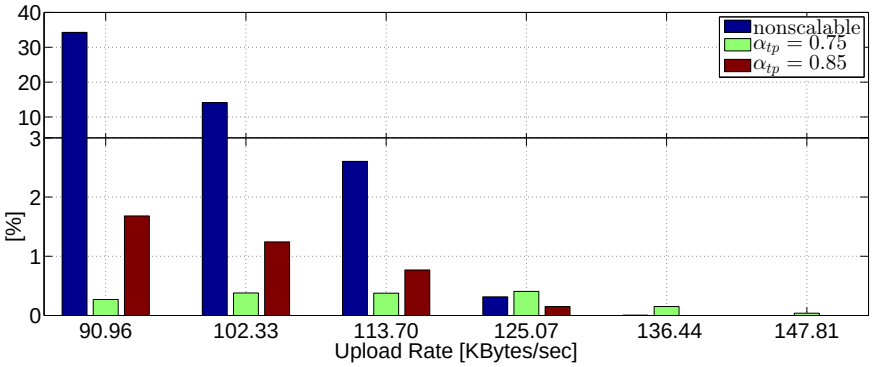

Fig. 5.19: Playback skips vs nodes upload rate. Server upload: 700 KBytes/sec. Network size: 20 nodes, Paris CIF sequence.

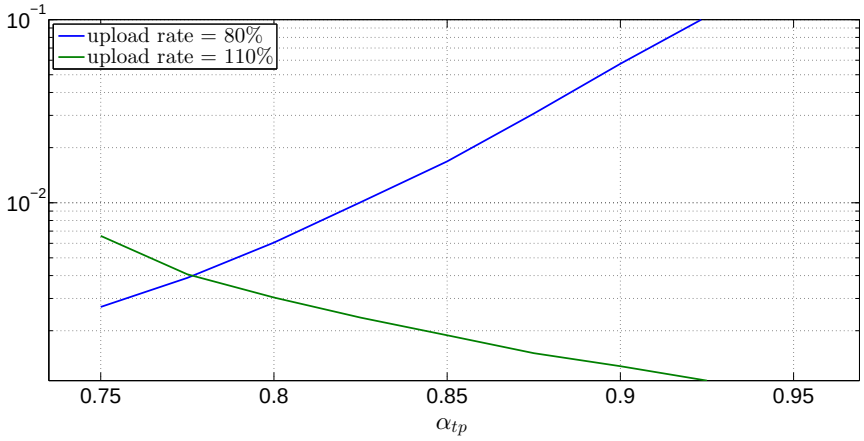

Fig. 5.20: Playback skips vs α_{tp} at two different node upload rates. Server upload: 700 KBytes/sec. Network size: 20 nodes, Paris CIF sequence.

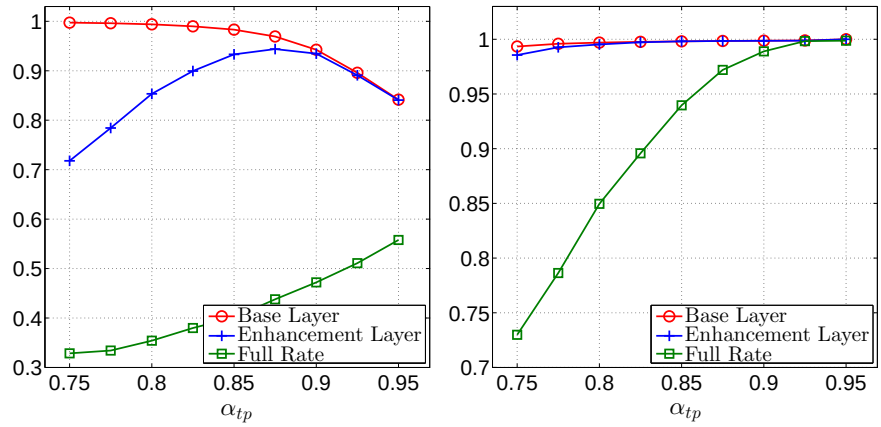

Fig. 5.21: Fraction of decoded segments vs α_{tp} at two different node upload rates: left 80%, and right 110% of the full rate video. Server upload: 700 KBytes/sec. Network size: 20 nodes, Paris CIF sequence.

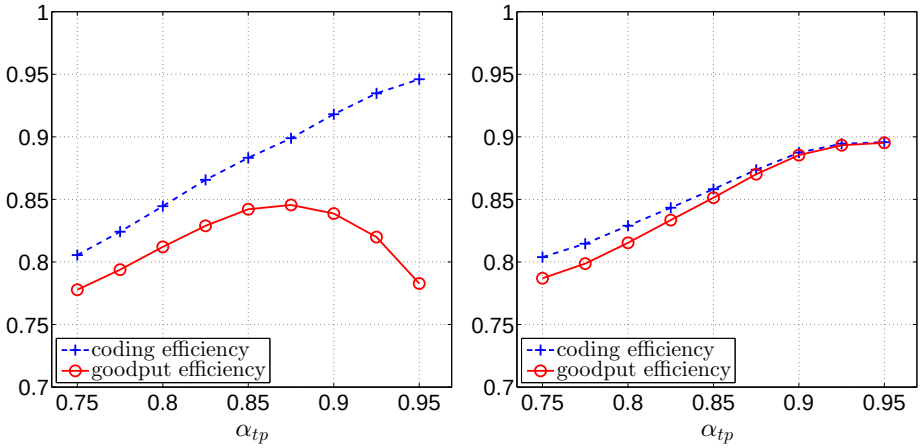

Fig. 5.22: Coding and goodput efficiency vs α_{tp} at two different node upload rates: left 80%, and right 110% of the full rate video. Server upload: 700 KBytes/sec. Network size: 20 nodes, Paris CIF sequence.

Discussion The reason why the tight estimation fails at limited throughput regimes is that, although the estimation of the pure throughput is accurate, a $5-10\%$ overhead is an underestimation of the overhead that the data allocated for such rate brings into the game at that same channel rate, yielding to an overall unreliable stream.

The overhead comes from three mechanisms: one is the statistical dependency of random linear combinations, the second one is the probability of delivering dependent packets from several sources. Breaking overhead is the third source of overhead. With regards to the last two, faster channels are more probable to deliver overhead due to braking. Conversely, slower channels suffer less from breaking overhead, but at the same time the data that can be reliably transmitted on these channels is smaller, and yields to higher probability of being non-innovative for the first two reasons.

The goodput estimation appears to be a double-loop process: One is the rate estimation, and the other is the estimation of overhead of data of some size, at that channel rate. Each time a target upload rate is chosen, this will bring to the table factors due to statistical diversity of generations with given size, and that depends with the speed of the channel, both interdependent and contributing to the channel overhead. Rather than perfectly estimating such overhead mathematically, which variates a lot depending on the scenario, one might think of applying all means to reduce it, and then dynamically learn to estimate more precisely from the previous transmission history.

Maintenance overhead An additional overhead is to be considered for establishing and maintaining our peer-assisted delivery system. Our system entails the constant exchange of

buffer maps to keep up-to-date buffering information across all peers. This involves preparing a minimum of one message every time the priority window slides forward containing the information about the buffered GOPs, to be sent to every peer in the cluster. In other words, the buffer map flows can be written as:

$$R_{buffer\_maps} = [(L+1)\gamma_{int} * P_g + \Gamma_{header}] \frac{frame\_rate}{GOP\_period}$$

where γ_{int} is the size of an integer in Bytes (4 Bytes in our example), the factor $L+1$ takes into account the number of blocks received per layer as well as the number of non innovative packets received for each generation, and Γ_{header} is the header size. For 8 frames GOPs of a 30 frames per second video, every node has to send a minimum of 480 Bytes to every other node, as well as receiving from every other node. In our examples of 20 and 50 nodes, this accounts for 9.6 Kbytes and 24 KBytes per second respectively. However, this can be easily reduced by:

- Using shorter integer representation and compressing the raw values.
- Including only the ACK and not-yet ACK as well as the subscription information, and omitting the number of received packets.
- Using a longer GOP period.
- Using the tracking server as a message forwarder, which would reduce the outgoing message from each node to only one for every buffer map update, regardless the number of peers.
- Limiting the number of peers to update by forming clusters or group of peers that see and exchange data with each other. This does not exclude that groups can overlap, to enhance the distribution of data.

Establishing and terminating the connections also brings a one off overhead from sending the updated maps of peers. An efficient way of doing is to update the users with the list of only newly joined peers, or alternatively, implicitly signalling the join of a new peer by forwarding its buffer map. While this could raise security concerns in the distributed maps exchange, when forwarding the messages at the tracking server this issues is easily circumvented. Finally, in order to signal when peers depart, nodes can forward only to nodes for which they have received a recent buffer map information, whereas if no buffer map is received, it is implied that the node has left the network. With this strategy, the network can maintained effectively and with a sustainable overhead traffic.

Delivery with lossy links In Chapter 6 we discuss the rateless coding data exchange mechanism in terms of complexity and resilience against packet losses. We argue in this contest that a linear invariance against packet losses and delays is achieved, allowing to apply the results demonstrated in this chapter in other situations of channel adversity. In fact, the only parameter driving both packet streaming and its stop condition, as well as the rate allocation, is the real actual goodput experienced by each node, regardless the channel conditions. Thanks to the possibility of encoding and streaming packets until the segments are decoded, the throughput will be higher and more protected, with more resources allocated towards lower layers, in presence of packet losses. We expect a transparent adaptation to the channel conditions, and a shift of PSNR and playback skip profiles only towards higher upload rates, but overall equivalent.

5.4 Conclusions

We've implemented a scalable video streaming system with peer-assisted delivery, over a P2P communication protocol. We've used the practical network coding framework, with a push-based distributed coding mechanism and added rate estimation and allocation. We've tackled the problem of allocating resources for scalable video and shown that the waterfilling algorithm achieves the task with limited and controllable overhead. This architecture can be also considered a cornerstone for testing and studying network coding form other perspectives, such as a channel with end-to-end characteristic as a linear operator, as in Chapter 4. On the other hand, the rateless coding can be studied as a point-to-point error control technique, to analyse the resilience against packet losses and random delays. This is discussed in Chapter 6.

The research that lead to the development of this framework was presented at an industry workshop, two peer reviewed conference papers, and an IEEE journal paper.

Chapter 6

Random Linear Fountain over $GF(q)$: Implementation and Performance

With the P2P content delivery system we have introduced a rateless coding and streaming characteristic coming from the random push of packets towards other peers. Fountain codes are a well know Forward Error Correction (FEC) coding technique, which has shown excellent erasure protection performance with very low complexity, compared to traditional block codes. The key concept of rateless coding resides in being able to deliver an arbitrarily high stream of information, geared towards overcoming by means of pure throughput the loss of packets, and decoding from any set of independent received packets.

In this chapter we discuss rateless coding as a functionality of network coding in our content delivery system. Although already studied by some [64], our system does not make use of traditional fountain codes, but introduces instead a dense linear fountain over $GF(q)$ as the means for performing network coding and being as a side effect an inter-packed loss protection coding. While fountain codes are, in their state of the art formulation, the best form of erasure protection on a point-to-point transmission, network coding relies on multiple and intersected paths to deliver the data. Additionally, batch decoding from $\omega + \varepsilon$ packets is the only way to exploit raptor codes' statistical characteristics at their best, whereas progressive decoding of network coded packets is essential to identify non-innovative packets and perform scalable decoding. For these reasons, random linear coding over $GF(q)$ as a tool for error and erasure protection, and at the same time as the key for practical implementation of network coding systems, is the focus of discussion of this Chapter.

Since in the last Chapter we presented the overall data distribution characteristic in an ideal situation, we now analyse the behaviour of point-to-point rateless transmission employing random linear coding. We analyse the computational delay introduced by our cod-

ing system on ideal conditions, and the robustness against packet losses and delays. One of our goals is to tackle one of the main weaknesses of network coding, which is the computational complexity. We make use of chunked coding to reduce the linear systems dimension, as well as an implementation of gaussian reduction that takes advantage of advanced CPU instructions. We show how our implementation is efficient, has low end-to-end delays and can be effectively implemented in live collaborative networking. We also argue the invariance of the system performance to channels with stationary packet loss rate.

6.1 A Collaborative Coding for Packet Loss Protection with Controlled Complexity

Fountain codes are now the most advanced form of forward error correction codes [43, 65]. Raptor10 [66], and the latest RaptorQ® [67] are now standards from the Internet Engineering Task Force (IETF) [66, 68] and are already part of products specifications from the 3rd Generation Partnership Project (3GPP) and the Digital Video Broadcasting (DVB) project for satellite, terrestrial and handheld broadcasting systems.

Practical network coding frameworks employing push-based policies for packet forwarding have an intrinsic rateless-ness coming from:

a. Random coding and thus memoryless production of packets of identical contribution of information.
b. Streaming at a non-specified throughput and coding rate. This can be at the same time lower than the data rate and/or directed to multiple receivers.
c. Successful decoding from receiving enough independent packets, regardless which ones and especially from where.

The idea of collaborative rateless coding is shown in Fig. 6.1. As discussed in the previous chapters, for such situation to hold, receivers and forwarders have to share a criterion to label the generations of packets. As far as the allocation of throughput share of the forwarders with regards to the information rate and the receiver's download rate is concerned, several techniques can be used. Among those, random push with capped rate has demonstrated to be a satisfactory solution, as discussed in Chapter 5.

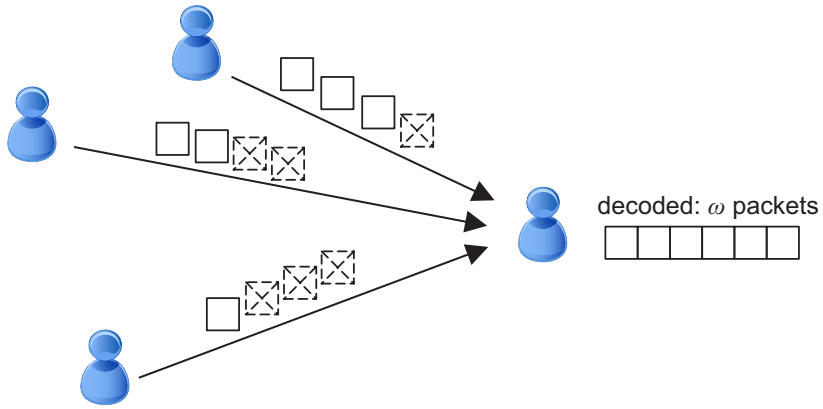

Fig. 6.1: An example of collaborative rateless delivery.

6.1.1 Rateless Coding over $GF(q)$

Collaborative streaming via network coding relies on similar assumption to those of fountain coding, one of them being the ability to decode the data from any set of received packets. Generation-based coding as well as in-band signalling of encoding coefficients allows decoding regardless the originating node, the arrival times, or the order of the received packets. Packet loss protection in collaborative coding is achieved by collecting as many packets as needed from any path converging to the receiver. Packet losses are compensated if the original information rate is carried in the packets remaining in the stream. The idea is explained in Fig. 6.1 and 6.2.

Our code, as well as fountain codes, is rateless, as the amount of data generated is not fixed. Some advantages over traditional FEC codes are:

- Flexibility of information rate allocation: The throughput of the encoded stream can be increased for extra protection, in case of increased packet loss in the network.
- Flexibility in prioritising data: We can selectively encode different portions of the data (via windowed coding, see Chapter 3) into packets of differentiated priority classes, producing a stream with unequal loss protection.
- Achievement of network coding: We can re-encode the packets that have been partially decoded, performing multi-hop and multipath delivery.

These characteristics are achieved thanks to the fact that the receivers can decode progressively upon every packet arrival, starting when the first packet is buffered and finishing when the last innovative packet is received. By doing so, the computational load is well spread over time. Degree-distribution-based fountain codes, on the other hand, perform best when

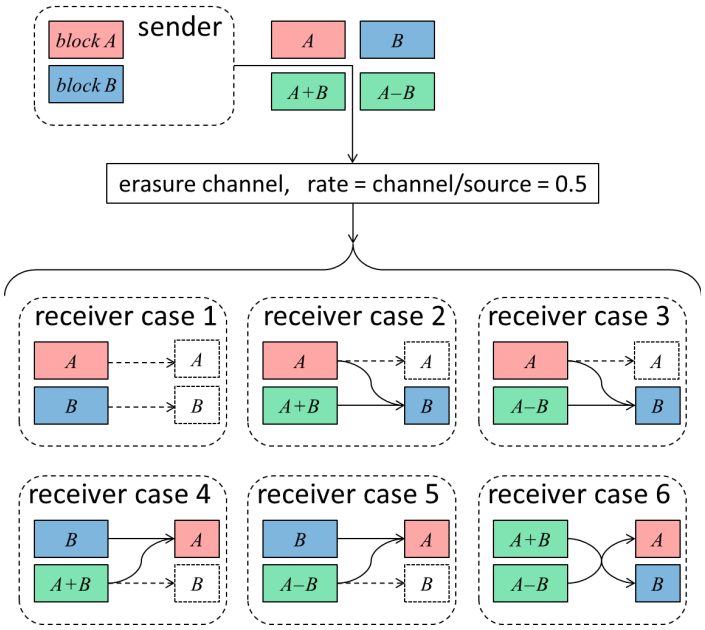

Fig. 6.2: Example of decoding from linearly independent subsets of packets.

decoding over the whole batch of packets, due to the Belief Propagation algorithm, and with a fixed overhead, i. e., a predefined number of additional packets to add to the minimal batch, to increase the chances of successful decoding. Flexibility constitutes one of the main advantages of dense codes compared to degree-distribution based fountain codes.

Our fountain over $GF(q)$ performs operations over symbols of the Galois Field of size $q = 2^m$, where m is the number of bits per symbol, and it has dense coding coefficients as the main difference with degree-distribution based fountain codes. This involves not having zero coefficients, and using a $GF(q)$ coefficient for a whole packet (of length w). Collaborative coding benefits from dense and randomised coefficients over a large field size better than sparse binary coefficients. Decoding is performed via inverting one $\omega \times \omega$ kernel for each generation of packets. However this happens progressively, and the length of the block at decoding is not dependent on the reception of a fixed overhead of packets as in raptor codes. When N_{out} packets are produced out of ω source packets the packet loss protection is close to a number of losses up to $N_{out} - \omega$, unless other overheads due to non-innovative packets. The exact probability that $N_{in} \geq \omega$ randomly encoded packets are independent was given in [24], and gives an idea of the overhead arising from random coding. The success probability was formulated as:

$$P_{\Omega,N}(\omega, N_{in}) = \prod_{n=0}^{\omega} \left(1 - \frac{1}{q^{N_{in}-n}}\right). \tag{6.1}$$

Table 6.1: Example of High Definition video coding bit-rates: MPEG 4 Visual, Advanced Simple Profile (ASP). JSVM 9.18.1 for H.264/MPEG-4 AVC High Profile (HP). High Efficiency Video Codec (HEVC - Draft 8) Main Profile (MP) [69].

1280x720p 60Hz 43 dB			
	MPEG 4 Visual	H.264/AVC HP	HEVC MP
Rate	3000 kbps	1600 kbps	1000 kbps
GOP length	GOP size		
8 frames	50 KBytes	26.67 KBytes	16.67 KBytes
16 frames	100 KBytes	53.33 KBytes	33.33 KBytes
32 frames	200 KBytes	106.67 KBytes	66.67 KBytes
64 frames	400 KBytes	213.33 KBytes	133.33 KBytes

1920x1080p 24Hz 40 dB			
	MPEG 4 Visual	H.264/AVC HP	HEVC MP
Rate	5300 kbps	2800 kbps	1900 kbps
GOP Length	N. packets per GOP		
8 frames	220.83 KBytes	116.67 KBytes	79.16 KBytes
16 frames	441.67 KBytes	233.33 KBytes	158.33 KBytes
32 frames	883.33 KBytes	466.67 KBytes	316.67 KBytes

While traditional Reed Solomon generation, e. g. with Vandermonde matrices, is a valid alternative to ensure minimal possible overhead in point-to-point transmission, randomised coding is the key to avoid duplication of encoded blocks in collaborative delivery; these are redundant data and would happen if the same linear combinations were deterministically encoded and delivered by different peers.

6.1.2 Chunked Implementation

With the introduction of cheap LCD screens and digital cinema, high resolution and high speed cameras, the video broadcasting market is moving towards providing High Definition (HD) as standard level of service and is looking forward to provide even higher resolutions to the home user. Table 6.1 shows typical compressed video rates using the most recent reference implementations of H.264/AVC and H.265/HEVC, which are regarded as the most efficient video coding standards available. Decoding with matrix inversion and reduction via the Gauss-Jordan elimination algorithm is known for having a complexity of $\mathcal{O}(\omega^3)$. With these sizes and with such growth of computational complexity, dense coding might limit the capability to stream live HD video content.

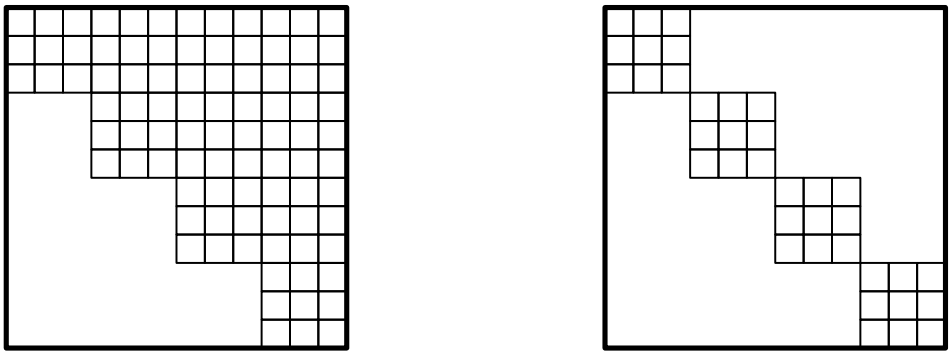

Fig. 6.3: Data partitioning for progressive coding (left) and chunked (stacked) codes (right).

We've discussed in Chapter 5 how larger linear systems yield to better coding diversity. However, large generation sizes could become a computational issue, especially for the live streaming constraints. Chunked coding aims at reducing the computational complexity by limiting ω and allowing encoding separate independent generations from the same data frame, in our case a GOP.

Let's now analyse the implications of prioritised coding and the chunked approach. The scalable and prioritised encoding was implemented as a progressive code over the generation. Generation segmentation realises instead a stacked coding of the GOP. The difference is shown in Fig. 6.3. In progressive coding, encoding is done cumulatively starting from the first segment of the generation, allowing a segment i to be encoded from all segments of index $j \leq i$. The resulting coding kernel for the generation is shown in Fig. 6.3, left. This is justified by the fact that, with scalable data, intermediate segments are useless if the preceding segments are not decoded. The stacked, or chunked code [70] approach preserves the independence of the chunks as shown in Fig. 6.3, right. In some cases the chunks might not be useful individually, but this method allows to recover the GOP by decoding several small linear systems, one for each generation. While decoding complexity decreases, packet loss protection is affected. Specifically, while a generation spanning a whole GOP is more robust against losses occurring anywhere in the stream, in chunked coding if the losses localised to one generation are more than the remaining information rate for that generation, this becomes undecodable and such data unrecoverable. It is however possible to recover the same amount of losses should the losses in each generation be such that the minimum amount of packets is recovered. Additionally, if one generation is non-decodable, the remainder of the GOP can be still used for decoding part of the video.

Appropriate sizing of the individual segments helps reaching an acceptable multipath

diversity. However, when the GOP reaches a remarkable size, stacked coding aims at providing a flexible balance between generation girth and coding diversity. Chunked coding of GOPs was implicitly introduced in Chapter 5 as a segmentation of GOPs in several generation. The coding scheme was also discussed. At the end of this chapter we will measure the benefits of chunked coding.

We now recall the coding scheme to discuss the decoding optimisation and the notation introduced in the previous Chapter. An outgoing packet $\mathbf{d}^{(g)}$ is generated by linearly combining the ω source blocks $\mathbf{b}_1^{(g)},\ldots,\mathbf{b}_\omega^{(g)}$ of generation g with random coefficients $c_1,c_2,\ldots,c_\omega$. The elements of the outgoing packet $\mathbf{d}^{(g)}=[y_1,\ldots,y_w]$ are calculated as:

$$y_j=\sum_{i=1}^{\omega} c_i x_{i,j}^{(g)},\quad j=1,2,\ldots,w \tag{6.2}$$

which is the element-wise equivalent of Eq.(5.2), where $x_{i,j}^{(g)}$ is the j-th element in the i-th block $\mathbf{b}_i^{(g)}$ in generation g. Encoding and decoding operations are performed in an algebra over a Galois Field (GF) of size 2^8, and the packets are w symbols long.

As enough packets are collected, the receiver builds a matrix of coefficients M, where $[M]_{i,j}=c_{i,j}$ corresponds to the global linear system of equations whose variables are $\mathbf{b}_1,\ldots,\mathbf{b}_\omega$:

$$\mathbf{Y}=M^T\mathbf{X}, \tag{6.3}$$

or equivalently

$$\begin{bmatrix}\mathbf{d}_1\\ \mathbf{d}_2\\ \vdots\\ \mathbf{d}_h\end{bmatrix}=\begin{bmatrix}c_{1,1} & c_{1,2} & \ldots & c_{1,\omega}\\ c_{2,1} & c_{2,2} & \ldots & c_{2,\omega}\\ \vdots & \vdots & \ddots & \vdots\\ c_{h,1} & c_{h,2} & \ldots & c_{h,\omega}\end{bmatrix}\begin{bmatrix}\mathbf{b}_1\\ \mathbf{b}_2\\ \vdots\\ \mathbf{b}_\omega\end{bmatrix}, \tag{6.4}$$

where h is the network flow or the allocated packet rate. In practice, the receiver doesn't wait to receive h packets and decode ω independent out of them, but decodes progressively as soon as every packet is received, checking the linear independency, dropping the non innovative packets, and spreading the computational cost over time.

This contrasts the use of connection-based protocols like TCP for reliable delivery. Such a protocol not only is not necessary in a distributed coding scenario, but it has also difficult application with network coding. Additionally, the flow control mechanism might yield to excessive end-to-end delays in presence of packet losses and random link latency, as explained in the Results section.

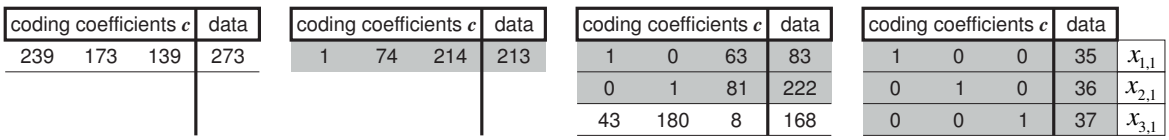

Fig. 6.4: Example of gaussian elimination.

6.1.3 Partial Decoding and Matrix Reduction

Received packets are decoded via Gauss-Jordan elimination which brings the linear system coefficients matrix to reduced row-echelon form, and transforms accordingly the buffered data as well. The row-echelon form is the key to allow partial decoding of the video as seen in Chapter 2. Non-innovative packets can be recognised and discarded.

We can now analyse the elimination algorithm and the fast implementation via accelerated CPU algebra instructions. Let's assume that at any time the buffered data has a corresponding matrix M where the coding coefficients are stored and decoded. Upon arrival of a new data packet, the decoding of the new line of coefficients goes as follows:

$$M_{4,\omega}\begin{bmatrix} 1 & 0 & 0 & c_{1,4} & \dots & c_{1,\omega} \\ 0 & 1 & 0 & c_{2,4} & \dots & c_{2,\omega} \\ 0 & 0 & 1 & c_{3,4} & \dots & c_{3,\omega} \\ c_{4,1} & c_{4,2} & c_{4,3} & c_{4,4} & \dots & c_{4,\omega} \end{bmatrix} \Rightarrow M'_{4,\omega}\begin{bmatrix} 1 & 0 & 0 & 0 & \dots & c_{1,\omega} \\ 0 & 1 & 0 & 0 & \dots & c_{2,\omega} \\ 0 & 0 & 1 & 0 & \dots & c_{3,\omega} \\ 0 & 0 & 0 & 1 & \dots & c_{4,\omega} \end{bmatrix} \quad (6.5)$$

where the $c_{i,j}$ have been transformed, but since the data in Y is decoded accordingly, Eq. (6.3) and (6.4) will hold. The Gauss-Jordan elimination procedure is outlined in Alg. 6.1 in order to introduce the advanced operations set. We can observe that the operations identified with † are performed on adjacent regions of memory. The portion of data and the performed operation are completely independent. Consecutive operations along the data blocks account for one order in exponent the polynomial complexity growth. Based on this assumption, parallelising the computation could drastically reduce the computational load.

Solutions based on General-Purpose computing on Graphic Processing Unit (GPGPU) have been proposed [71]. Modern GPU architectures enable high-performance parallel computing using simple programming languages, such as OpenCL (Open Computing Language), the Compute Unified Device Architecture (CUDA) model provided by nVIDIA, or the RenderScript language from Google. Parallel computing allows distributing the finite-field algebra operations on multiple GPU cores, reducing the decoding times and additionally leaving the CPU free to perform other operations.

Other approaches rely on instruction sets for Single Instruction Multiple Data (SIMD)

on the CPU, like the Streaming SIMD Extension (SSE) by Intel. While GPU computing could divide the job over a higher number of parallel jobs, his deployment depends on the availability of medium high-end GPUs and on the number of cores. SIMD on the other hand are nowadays common instruction sets on processing units, including smartphones, e. g. with the NEON instruction set available on ARM chipsets. It's worth notice that as soon as GPU becomes an ordinary component of hand-held devices, GPU computation might turn out to be the most efficient solution.

Algorithm 6.1 Gauss-Jordan reduction. $M_{p,\omega}$ is the matrix of partially decoded coefficients, $Y_{p,w}$ is the matrix of data, as in Eq. (6.5), left.

1: **procedure** REDUCE TO ROW-ECHELON FORM($M_{p,\omega}, Y_{p,w}$)
2: **for all** $i = 1, \ldots, p-1$ **do**
3: $c = M(p,i)/M(i,i)$
4: $M(p,:) = M(p,:) + M(i,:) * c$ † ▷ $\omega - i$ mult. and $\omega - i$ add.
5: $Y(p,:) = Y(p,:) + Y(i,:) * c$ † ▷ w mult. and w add.
6: **end for**
7: **if** $M_p(p,p) == 0$ **then**
8: find $j > p | M(p,j) \neq 0$
9: swap j-th column $M(:,j)$ with p-th column $M(:,p)$
10: **end if**
11: $M(p,:) = M(p,:)/M(p,p)$ † ▷ $\omega - p$ mult.
12: $Y(p,:) = Y(p,:)/M(p,p)$ † ▷ w mult.
13: **for all** $i = 1, \ldots, p-1$ **do**
14: $M(i,:) = M(i,:) + M(p,:) * M(j,p)$ † ▷ $\omega - i$ mult. and $\omega - i$ add.
15: $Y(i,:) = Y(i,:) + Y(p,:) * M(j,p)$ † ▷ w mult. and w add.
16: **end for**
17: **end procedure**

6.1.4 GF Arithmetic with SSE

The Streaming SIMD Extensions (SSE) instructions set allows manipulating elements spanning 128-bit registers on the CPU with a single instruction. Operations on $GF(2^8)$ elements can be performed for example on 16 elements at a time, allowing reducing the number of instructions to the CPU. We make use of an open source library implementation of GF arithmetic that uses the SSE instruction set version 4 [72] which is available at [73]. Multiplication and sum of contiguous GF elements is implemented via the *region_multiplication*

function which multiplies a 128-bits region **a** by a constant *c*. We briefly explain here the region multiplication in $GF(2^8)$, whereas implementations for other field size can be found in [72]. The routine follows the following steps:

0. Every element a_i of **a** is partitioned into $a_{i,left}$ containing the first 4 most important bits and $a_{i,right}$ containing the last 4 bits. Two tables are needed: *table1* containing all 16 multiplications of *c* with all 4-bit elements (includes all results of multiplication with 8-bit elements whose first 4 bits are zeros), and *table2* contains all the products with 8-bits elements whose last 4 bits equal to zero.
1. **a** is bitwise AND-ed with a 128-bit mask containing 16 times 11110000 (4 ones, and 4 zeros).
2. The result is multiplied by *c* by doing 16 simultaneous lookups on *table1*.
3. **a** is bitwise AND-ed with a 128-bit mask containing 16 times 00001111 (4 zeros, and 4 ones).
4. The result is right-shifted by 4 bits.
5. 16 simultaneous lookups on *table2* are made to multiply by *c*.
6. The final result is obtained by bitwise XOR of the results of operation 2 and 5 on the 128 bit word.

Precomputed tables are necessary for step 0, but since the allowed values are 256 and each table has 16 elements, these can be stored. This routine was thoroughly explained and documented for RAID-6 systems in [74] and for generic GF algebra in [72] This routine allows us to use one *region_multiplication* every 16 bytes. This means $\lceil \omega/16 \rceil$ or $\lceil w/16 \rceil$ instruction calls, the latter translating into 64 calls for data blocks of size 1024 bytes.

6.2 Experimental Validation

Our experimental validation tests aims at showing the validity of random linear fountain coding as a sustainable and reliable delivery system with resilience against packet losses. We show the performance on a point-to-point link in terms of end-to-end delay, comparing it with traditional TCP which tackles packet losses via retransmission. We also demonstrate the practicality of implementing a real live streaming scenario via our GF-SSE system by measuring absolute decoding times. Rather than relying on the sliding window model, we analyse the actual decoding and end-to-end transmission delays of GOPs. We argue that decoding can be kept at enough small values to support live High Definition (HD) video streaming.

6.2.1 Setup

Our validation setup is composed by the back-end streaming components of the P2P application. We produce data frames that are segmented into multiple generations as shown in Fig. 5.5 and considered as independent coding sets as in Fig. 6.3, right. The generations are divided into data blocks and randomly encoded with a target bit-rate on the channel calculated as $R_c = h * (w + \omega) * N_{gen}/F_l$, where ω is the length of the coefficients vector prepended to the payload, N_{gen} is the number of generation in the frame and F_l is the data frame duration in seconds.

An initial set of tests was made to measure the decoding times at the receiving end. Decoding times are taken as the interval of time occurring between the instant the first packet has reached the receiver, and the time it takes for the last innovative packet of the last generation of the GOP to be completely decoded. We monitor decoding times against the frame size, ranging from sizes typical of standard resolution video to GOP sizes of the order of 500 KBytes. We compute decoding times on two types of machines: One being a typical mid-high range desktop machine with Intel Core i5 CPU, and one a budget laptop machine with Intel Core i3 Mobile processor. We compare our GOP segmentation, entailing the partition of the GOPs in multiple generations, to single-generation GOP decoding. We will notice the reduction of decoding times and the trend linearisation against the data size.

Secondly, we carry out a real point-to-point transmission of a stream of data in a lab environment, between two machine connected on a local network. The goal is to emulate the effort made by peers in delivering data to each other, and compute the delay introduced over the transport and network protocols. We emulate the effect of adverse channels by making use of an emulation tool — the Network Emulator for Windows Toolkit (NEWT). The emulator operates on the physical network interface of the machine and allows us to manually tune channel characteristics such as, among the others, channel rate, latency, and packet loss rate. We measure the end-to-end delay between video encoding in one site and decoding on the other site from acquisition of data to delivering of the decoded frame to the video decoder. We rely on a synchronised virtual clock service with a skew of few milliseconds to validate our measures.

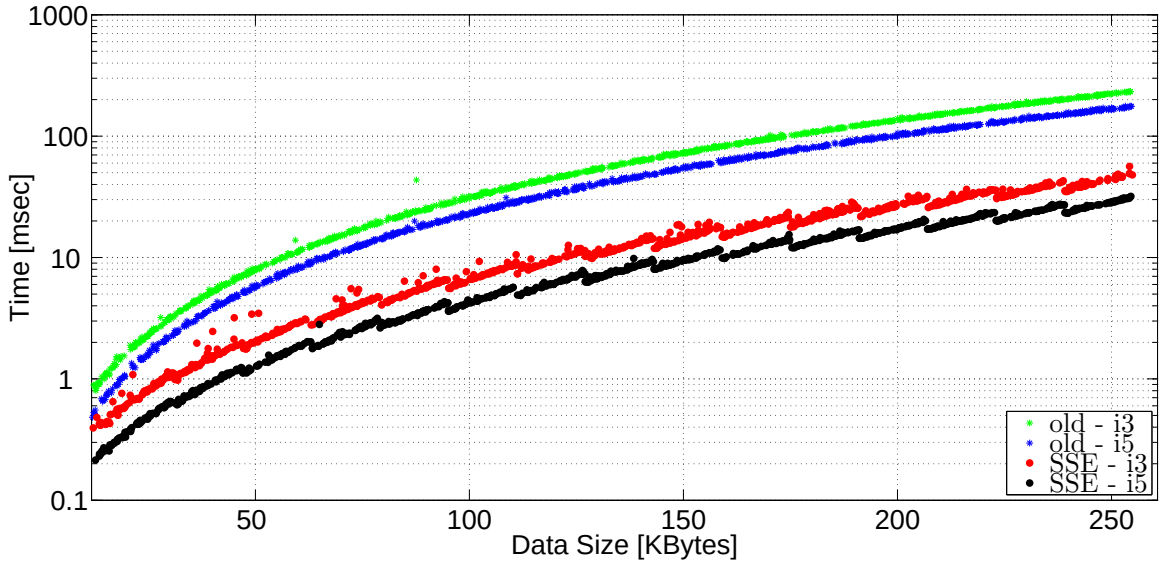

Fig. 6.5: GOP decoding delay with single-generation GOPs on the two CPU architectures Intel i3 and i5.

6.2.2 Results

The results of decoding time measurements are presented first, followed by the end-to-end delay measurements.

6.2.2.1 Decoding time

Measurements of decoding time are shown in Fig. 6.5. For these results we generated data with size from 10 KBytes to 255 KBytes, segmented into 1 KByte blocks, and encoded via a single generation for the whole data segment. Therefore the coding kernel size ranges from 10 to 255. Decoding has been performed on both the i3 and the i5 machines. The comparison with the old implementation of Gaussian elimination for decoding shows decoding times roughly 5 to 6 times faster, on both CPUs. At the highest rate ($w = 255$ blocks per generation), we can decode up to 9 MBytes per second on the i5, and 5.5 Mbytes per second on the i3, which are more than enough to decode the HD video with rates from Table 6.1.

These measurements do not take into account additional CPU time for decoding the video. Also, these decoding times are for batch decoding of a linear system. Progressive decoding over an extended period of time (depending on arrival times of packets) results in an overall smaller transmission time, whereas degree-distribution-based fountain codes best perform when decoding over the whole batch.

Fig. 6.6 shows the decoding delays with GOP partitioning. The three figures show a

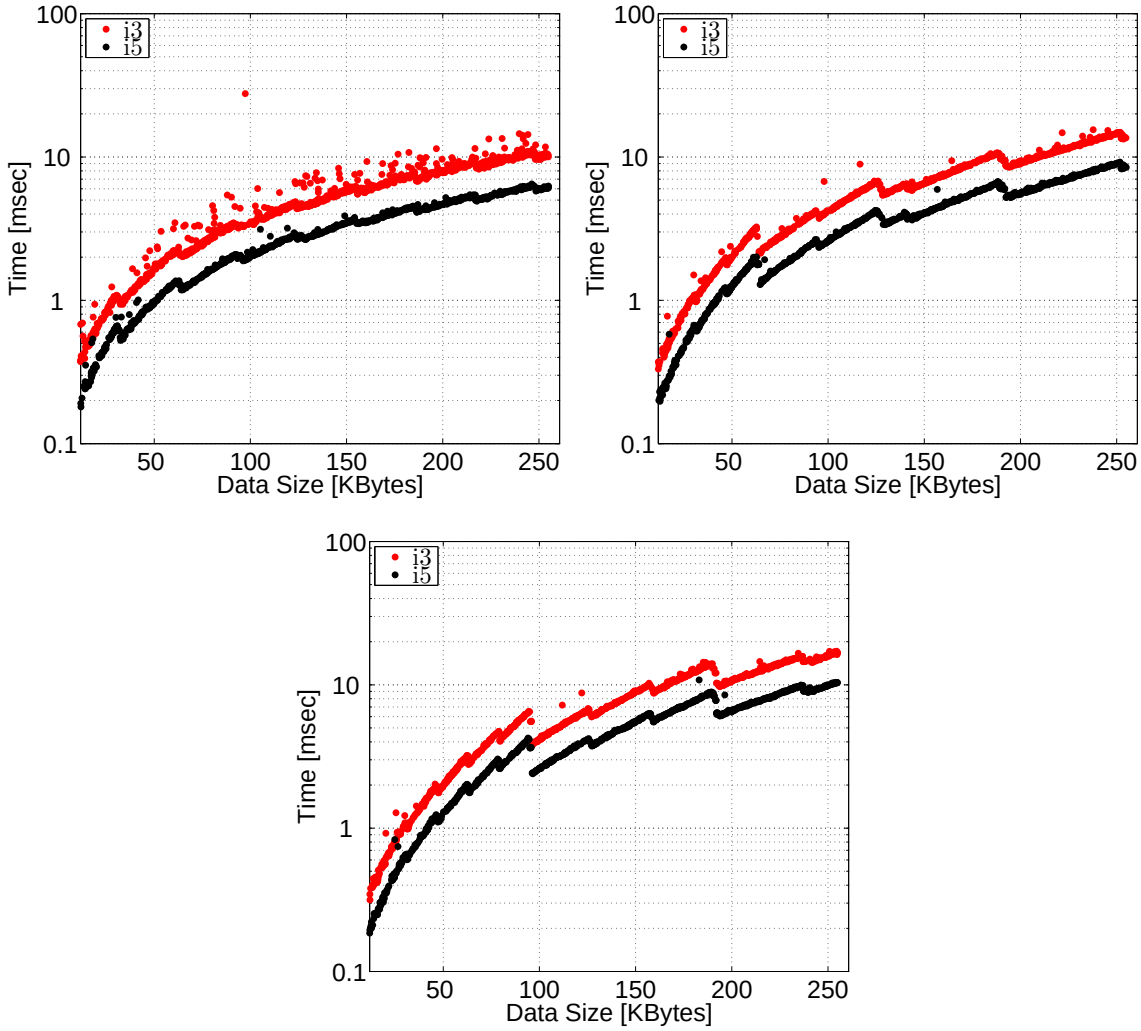

Fig. 6.6: GOP decoding delay on the two CPU architectures Intel i3 and i5, with different generation sizes: 32 Kbytes (top-left), 64 Kbytes (top-right), and 96 Kbytes (bottom).

maximum kernel size of 32, 64 and 96, respectively. Decoding rates achievable with these systems are shown in Table 6.2. The reduction in decoding time of the whole 255 KBytes GOP, when partitioned in 32 KB chunks is of another factor of 5. Decoding times on the faster machine are shown for the three systems in Fig 6.7.

Table 6.2: Decoding speeds reduction with chunked coding and SSE elimination decoding.

	single-generation	$\omega = 96$	$\omega = 64$	$\omega = 32$
i3	5.5 MB/sec	13 MB/sec	16 MB/sec	24 MB/sec
i5	9 MB/sec	24 MB/sec	28 MB/sec	40 MB/sec

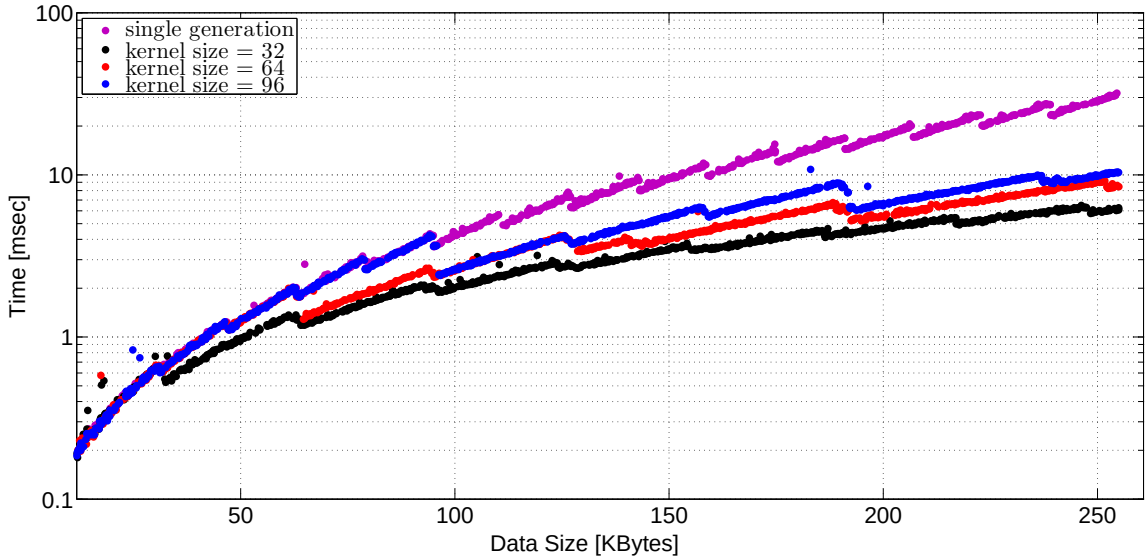

Fig. 6.7: Decoding delay on the Intel i5 CPU with different generation sizes and single-generation GOP.

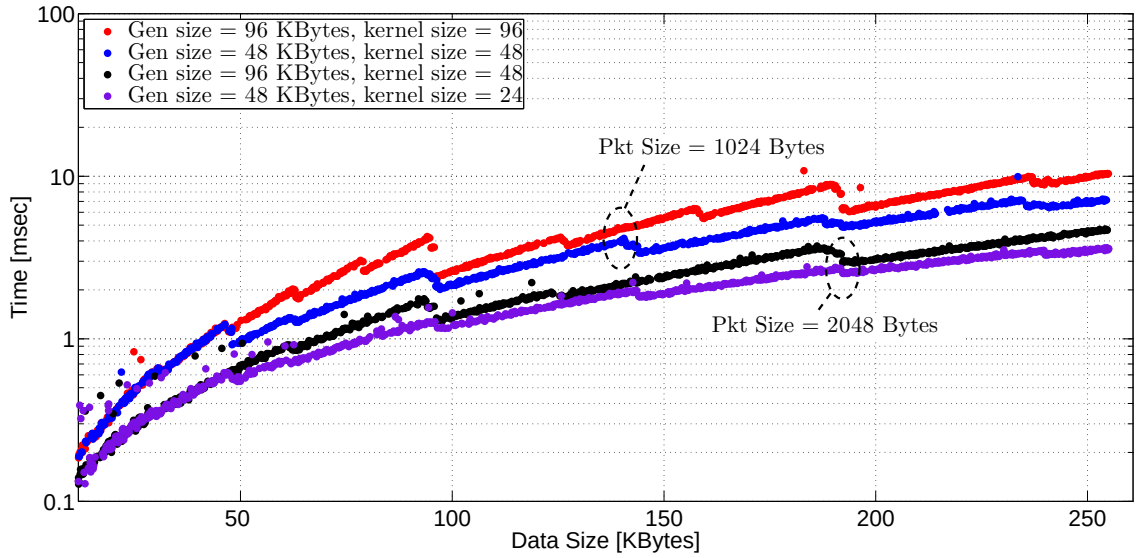

Fig. 6.8: Decoding delay on the Intel i5 CPU with different generation and packet sizes.

For the sake of comparison, if one assumes to transmits a video with GOP size of maximum 96 Kbytes, we note the following: If the GOPs consist of 16 frames, the video rate would be of 180 KBytes per second, which is compatible and higher than the value given in Table 6.1 for AVC HP and HEVC video in HD/60Hz at 43 dB. Alternatively, if the video is encoded with 8 frames per GOP, it would have a rate of 360 Kbytes per second, which also approaches the rates of HD/60Hz and full HD/24Hz video in Table 6.1, for both AVC

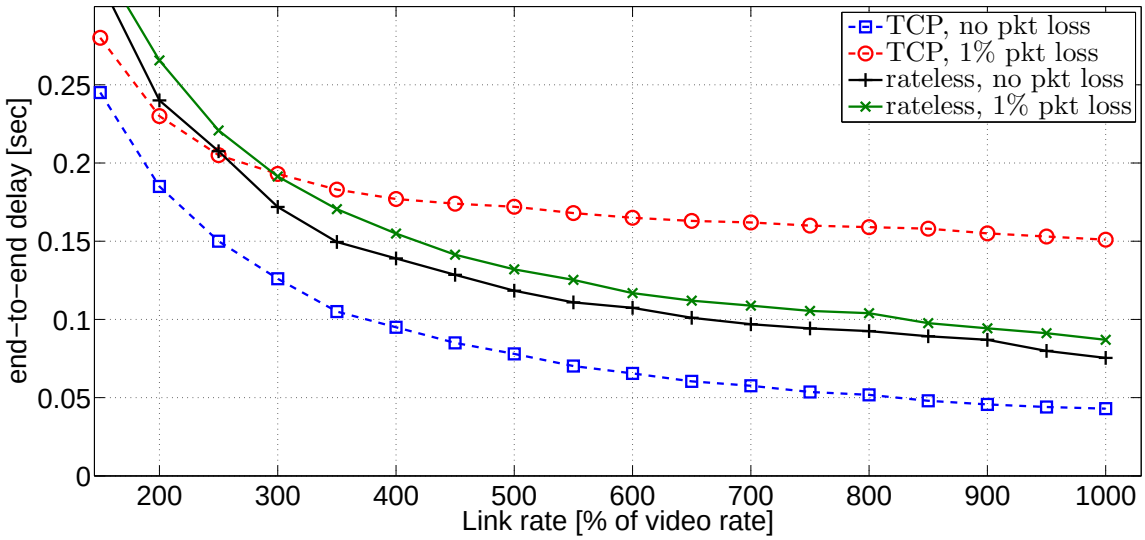

Fig. 6.9: Transmission delay of TCP and rateless coding varying depending on the available channel rate.

High Profile and HEVC. This does not exclude though that, a single-generation approach could sustain higher video rates and GOP length, give the decoding speeds commented so far, before turning to partitioning into multiple generations.

The main reason for achieving higher processing rates is the reduction of the system kernel (ω). Other than partitioning in multiple generations, or choosing a shorter GOP length, another way of reducing the kernel size is to increase the packet size, for which the decoding time become even smaller, although making the stream weaker against packet erasures. A comparison of decoding times with different packet sizes is shown in Fig. 6.8.

6.2.2.2 End-to-end delay

We measured the end-to-end delay at the application in a point-to-point transmission between two machines: one generating and encoding data, and one receiving from the network and decoding the the segments of data. The network emulator reproduces a variety of network conditions in terms of delay, bandwidth and packet loss rate at the receiver's interface. We compare the rateless transmission over UDP with a reliable TCP connection. Fig. 6.9 shows the delay with increasing throughput available from the network. We can see how the rateless coding works best with plenty of available bandwidth. One can observe also how, channel impairment leave a gap on TCP performance, due to recovery routines, whereas the rateless coding is less affected by packet losses in the global latency, being most of the delay due to the packet processing. It's important to note that the effect provoked here by the

emulator limiting the throughput is that of actually slowing down the reception of packets, thus the impact on decoding bursts of data (the GOPs) is that of longer latency.

The little variation of delay, both with network latency and packet loss rate, can allow us to conclude that the performance of the P2P system increases linearly in adverse channel condition. With packet losses, similar performance with the loss-free case are expected, provided an equivalent incoming packet rate to the peers. This is obtained by increasing the senders throughput, implying a shifting of the performance indicators shown in the last Chapter towards higher values of uplink bandwidth.

6.3 Conclusions

This Chapter analysed differences and similarities between state-of-the art fountain codes and the rateless codes over $GF(q)$ used in practical network coding systems. Fountain codes have proven to be the most efficient form of erasure protection. As we analyse possible extensions of this model to multi-source coding, network coding, and prioritised coding, a solution which is optimal on all fronts is difficult to identify. On the other hand we argue that rateless fountain over $GF(q)$ is a viable and efficient solution to most aspects.

Tests on generated data show values of the end-to-end low enough under limitedly adverse channel conditions to conclude that the content delivery system proposed in Chapter 5 is practical and even with adverse channel conditions we would expect it to perform equivalently. Implementation and real-life validation of this coding system on a platform for live teleimmersive conferencing has been presented in two position papers and a peer reviewed conference, as well as having earned an invitation to the IEEE transactions on Multimedia.

Chapter 7

Conclusions

As multimedia communication gains territory in nowadays and near future internet applications, the network infrastructure and the tools for content delivery need to be designed more around the user and around the service. The effort in formalising and standardising Information Centric Network (ICN) architectures aim at providing an efficient approach to content caching and retrieval based on objects identification rather than location. Since network coding has arisen as a new tool for improving data distribution on packet networks, coding and source-aware data manipulation have become as a key factor in the design of video streaming systems. SVC has emerged as a key technology that allows seamless diffusion of video services to different platforms in universal video systems.

After presenting fundamental theory of network coding in Chapter 2 (Part of which, together with the last two appendices has been published on the *Hindawi, International Journal of Digital Multimedia Broadcasting* in May 2011), and having briefly surveyed scalable data coding and state-of-the art channel coding in Chapter 3 we analysed various embodiments of data-chunks coding for scalable data delivery. We've assumed a randomised coding process arising from widely used practical network coding models, and proposed a Joint Source-Channel-Network Coding mechanism for error protection for this channel (Chapter 4). We then described an application-layer node architecture providing the means to perform packet-based network coding and multicast on an end-system overlay. We demonstrated the potential of the end-system multicast with a P2P protocol handling live scalable video via prioritised coding and dynamic resource allocation (Chapter 5). We've finally presented and documented an optimised implementation of rateless coding for live streaming applications and with direct application in the network coding infrastructure.

After presenting and analysing the different components, we can draw some general conclusions.

7.1 Outcome and Future Work

Our work has been presented in three main contributions analysing different aspects of the overall architecture.

Joint Source-Channel-Network Coding is based on an oblivious or non-coherent transmission model. We've presented a channel-network coding allocation model and demonstrated the performance on a matrix channel, modelled and dimensioned over simple graphs. As the flow path model of network coding has been studied with several theoretical insights on such small networks, we've also limited our study to a choice of small codes. Such choice is the best fit for demonstrating the role of the code in transmitting over multiple paths, with the algebraic transmission model helping to best express the use of network error correction coding for errors on independent links. Additionally, small codes which are fit for a sample flow-path network model don't limit the applicability to bigger networks employing the asynchronous transport model. We've demonstrated a clear trend of error impact at application layer, which at some regimes is controllable, and thanks to erasure/error decoding can ensure error free transmission at some low link error rates with very little overhead. This system has been presented at the *7th International ICST Mobile Multimedia Communications Conference* (MOBIMEDIA 2011) in September 2011 and published in the *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering* in February 2012. As far as future developments are concerned, as the underlying network dynamics move towards the asynchronous buffered model, the scheme could be applied at every receiver as an inter-packet coding method, where the redundant packets received by every node build up to providing the means for the error detection. A packet-wise protection or CRC could be also implemented and analysed as an intermediate detection step and the impact on the rate-distortion curve could be studied.

with the P2P system we've proved the validity of the synchronised buffering mechanism and the rate allocation based on goodput estimation. We've demonstrated the adaptability to reduced reception conditions, with reduction of the video stream to lower quality while keeping a continuous service, as well as the close performance with the non-scalable case (where the braking overhead is minimal) when the rate allows. We've analysed an heuristic adaptation to the overhead, finding that loose estimation is a safer option. We've also had the chance to speculate over the origin of the overhead, including braking phenomenon, size of data, size of network, and factor influencing the chunk/peer selection, while keeping this overhead low enough in our simulated experiments. An early implementation of this system was presented at the *9th ST-Day Workshop*, in October 2012, and at the *4th IEEE Latin-*

American Conference on Communications 2012 (LatinCom 2012) in November 2012, later published as a journal paper in the *IEEE Latin America Transactions*, in March 2013. The rate allocation was presented at the *20th International Packet Video Workshop (PV), 2013*, and included in the conference proceedings. As Content Delivery Networks are still studied and employed for geographic replication of the service across continents, it would be interesting to see the application of such a collaborative system with rateless network-coded datagram-based distribution across the nodes, and the impact on the playback delay of the connected users, as well as the use for multiple live streams. As this model is applicable to several scenarios another next step would be the application of the streaming system to optimised overlay topologies for content delivery. Information-Centric Networking would also benefit from an encoded approach to object caching. As multicast Over-The-Top Video becomes a high demand service, encoded architectures for caching and delivery might become an interesting approach to meet the requirements of ICNs.

Finally, we have presented an analysis of rateless streaming on a per-link basis. Assuming the dense fountain over $GF(q)$ as the most generic situation for a rateless delivery with network coding, we've implemented a decoding architecture which can process packets at a fast rate to support HD video. This work was part of a small demonstrator for live streaming of really large data and demonstrated very low end-to-end delays, in this case below the real-time interactivity threshold. It was presented as part of the system at the *4th ACM Conference on Multimedia Systems* (MMSys 2013), 2013, where it earned an invitation for an extended submission at the *IEEE Transactions on Multimedia*. Since raptor codes and LDPC codes have affirmed their superiority in terms of complexity, making a point for sparseness and randomness, and we've made our point towards distributed-ness of the code, study of raptor codes in network coding is already making its course. At the current state raptor network coding suffers from decreased efficiency with small blocks, and need to stick to specific decoding schemes. Efficiency of raptor network coding is close to that registered in our experiments, i. e. up to 90% for the chosen block sizes [75]. Although beyond the scope of this thesis, the attractiveness of state of the art fountain codes remains, where one could study, among the other options, the possibility to extend decoding beyond a fixed overhead to increase the chances of success (extended raptor decoding), and the limits and advantages of balancing partial decoding with collaborative re-encoding for retransmission. Finally the introduction of scalable data in the picture is a good reason for further exploration of the efficiency of such codes with partial decoding as well as pushing the efficiency with limited block sizes as in our system.

7.2 List of Publications

Journals and book chapters

[1] M. Sanna and E. Izquierdo, "A Survey of Linear Network Coding and Network Error Correction Code Constructions and Algorithms", *International Journal of Digital Multimedia Broadcasting*, vol. 2011, Article ID 857847, 12 pages, 2011.

[2] M. Sanna and E. Izquierdo, "A method for Detection/Deletion via Network Coding for Unequal Error Protection of Scalable Video over Error-Prone Networks," *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, 2012, Volume 79, Part 3, 105-120.

[3] M. Sanna, E. Izquierdo, "Live scalable video streaming on peer-to-peer overlays with network coding", *Latin America Transactions*, IEEE (Revista IEEE America Latina), vol.11, no.2, pp.690,697, March 2013

[4] R. Mekuria, M. Sanna, E. Izquierdo, D. C. A. Bulterman, and P. Cesar, "Real-Time Streaming of Live Reconstructed Mesh Geometry for 3D Tele-Immersion" (invited paper) submitted to *IEEE Transactions on Multimedia*, November 2013.

Conferences Papers (peer-reviewed)

[5] M. Sanna and E. Izquierdo, "Multirate Delivery of Scalable Video with Progressive Network Codes", in *Proceedings of the 19th European Signal Processing Conference* (EUSIPCO 2011), Aug 29- Sept. 2, 2011, Barcelona (Spain).

[6] M. Sanna and E. Izquierdo, "A method for Detection/Deletion via Network Coding for Unequal Error Protection of Scalable Video over Error-Prone Networks," *7th International ICST Conference on Mobile Multimedia Communications*, MOBIMEDIA 2011, Cagliari, Italy, September 5-7, 2011.

[7] M. Sanna, E. Izquierdo, "Live scalable video streaming on peer-to-peer overlays with network coding", *IEEE Latin-America Conference on Communications* (LATINCOM), 2012, vol., no., pp.1,6, 7-9 Nov. 2012.

[8] R. Mekuria, M. Sanna, S. Asioli, E. Izquierdo, D. C. A. Bulterman, and P. Cesar, "A 3D Tele-Immersion System Based on Live Captured Mesh Geometry", in *Proceedings of the 4th ACM Conference on Multimedia Systems* (MMSys 2013), Oslo, Norway, February 27- March 1, 2013.

[9] P. Fechteler, A. Hilsmann, P. Eisert, S. V. Broeck, C. Stevens, J. Wall, M. Sanna, D. A. Mauro, F. Kuijk, R. Mekuria, P. Cesar, D. Monaghan, N. E. O'Connor, P. Daras,

D. Alexiadis, and T. Zahariadis. 2013. "A framework for realistic 3D tele-immersion". In *Proceedings of the 6th International Conference on Computer Vision / Computer Graphics Collaboration Techniques and Applications* (MIRAGE ’13), 6-7 June 2013, Berlin, Germany.

[10] D. A. Mauro, R. Mekuria and M. Sanna, "Binaural Spatialization for 3D immersive audio communication in a virtual world", In *Proceedings of the 8th Audio Mostly Conference: A Conference on Interaction with Sound* (AM ’13). September 18-20, 2013, Piteå, Sweden, 8 pages.

[11] M. Sanna, E. Izquierdo, "Proactive Prioritized Mixing of Scalable Video Packets in Push-Based Network Coding Overlays," *Packet Video Workshop (PV), 2013 20th International*, vol., no., pp.1,7, 12-13 Dec. 2013.

Position Papers

[12] M. Sanna, N. Ramzan and E. Izquierdo, "Streaming of Scalable Video in Network-Coding-capable Networks: Progressive Code Design for Multirate Streaming", in 8th ST-Day, Streaming Day Workshop, 2011, 30 September 2011, Turin, Italy

[13] M. Sanna, N. Ramzan and E. Izquierdo, "Scalable streaming with network coding in push-based Peer-to-Peer networks" in 9th ST-Day Workshop, 2012, 26-26 October 2012, Milan Italy

[14] R. Mekuria, M. Sanna, P. Cesar, "Media Synchronization in 3D tele-Immersion Applications: an Architecture". International Workshop on Media Synchronization, in Conjunction with ICIN 2012, Berlin, Germany.

References

[1] N. Sprljan, *A Flexible Scalable Video Coding Framework with Adaptive Spatio-Temporal Decompositions*. PhD thesis, Queen Mary, University of London, London, August 2006.

[2] M. Mrak, *Motion Scalability for Video Coding with Flexible Spatio-Temporal Decompositions*. PhD thesis, Queen Mary, University of London, London, January 2007.

[3] P. C. Yunnan, P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Allerton Conference in Communication, Control and Computing, Monticello, IL*, oct. 2003.

[4] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h.264/avc standard," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 17, no. 9, pp. 1103–1120, 2007.

[5] J. Gaedtke, "Keynote speech: Optimizing QoE in large-scale video networks," Dec 2013.

[6] "Bittorrent live." http://live.bittorrent.com/.

[7] J. A. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. R. van Steen, and H. J. Sips, "Tribler: A social-based peer-to-peer system: Research articles," *Concurr. Comput. : Pract. Exper.*, vol. 20, pp. 127–138, Feb. 2008.

[8] N. Zeilemaker and J. Pouwelse, "Open source column: Tribler: P2p search, share and stream," *SIGMultimedia Rec.*, vol. 4, pp. 20–24, Mar. 2012.

[9] "Streamroot: Smart P2P video streaming. demo available." `http://www.streamroot.io/`.

[10] "P2P-Next FP7." `http://www.p2p-next.org/`.

[11] Y. hua Chu, S. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *Selected Areas in Communications, IEEE Journal on*, vol. 20, pp. 1456–1471, oct 2002.

[12] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, "Network information flow," *Information Theory, IEEE Transactions on*, vol. 46, pp. 1204–1216, jul 2000.

[13] D. Lun, N. Ratnakar, M. Medard, R. Koetter, D. Karger, T. Ho, E. Ahmed, and F. Zhao, "Minimum-cost multicast over coded packet networks," *Information Theory, IEEE Transactions on*, vol. 52, pp. 2608–2623, june 2006.

[14] Y. Cui, Y. Xue, and K. Nahrstedt, "Optimal distributed multicast routing using network coding," in *Communications, 2007. ICC '07. IEEE International Conference on*, pp. 3610–3615, 2007.

[15] R. Dougherty, C. Freiling, and K. Zeger, "Insufficiency of linear coding in network information flow," *Information Theory, IEEE Transactions on*, vol. 51, pp. 2745–2759, aug. 2005.

[16] S.-Y. Li, R. Yeung, and N. Cai, "Linear network coding," *Information Theory, IEEE Transactions on*, vol. 49, pp. 371–381, feb. 2003.

[17] R. Koetter and M. Medard, "An algebraic approach to network coding," *Networking, IEEE/ACM Transactions on*, vol. 11, pp. 782–795, oct. 2003.

[18] R. W. Yeung and N. Cai, "Network error correction. I. Basic concepts and upper bounds," *Commun. Inf. Syst.*, vol. 6, no. 1, pp. 19–35, 2006.

[19] N. Cai and R. W. Yeung, "Network error correction. II. Lower bounds," *Commun. Inf. Syst.*, vol. 6, no. 1, pp. 37–54, 2006.

[20] P. A. Chou and Y. Wu, "Network Coding for the Internet and Wireless Networks," tech. rep., Microsoft Research, June 2007.

[21] P. Chou and Y. Wu, "Network coding for the internet and wireless networks," *Signal Processing Magazine, IEEE*, vol. 24, pp. 77–85, sept. 2007.

[22] T. Ho, M. Medard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *Information Theory, IEEE Transactions on*, vol. 52, pp. 4413–4430, oct. 2006.

[23] H. Balli, X. Yan, and Z. Zhang, "On randomized linear network codes and their error correction capabilities," *Information Theory, IEEE Transactions on*, vol. 55, pp. 3148–3160, july 2009.

[24] X. Zhao, "Notes on "exact decoding probability under random linear network coding"," *Communications Letters, IEEE*, vol. 16, no. 5, pp. 720–721, 2012.

[25] O. Trullols-Cruces, J. Barcelo-Ordinas, and M. Fiore, "Exact decoding probability under random linear network coding," *Communications Letters, IEEE*, vol. 15, pp. 67–69, January 2011.

[26] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 4, pp. 2235–2245 vol. 4, 2005.

[27] K. Jain, M. Mahdian, and M. R. Salavatipour, "Packing steiner trees," in *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '03, (Philadelphia, PA, USA), pp. 266–274, Society for Industrial and Applied Mathematics, 2003.

[28] J. Guo, Y. Zhu, and B. Li, "Codedstream: Live media streaming with overlay coded multicast," in *In Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking (MMCN 2004*, pp. 28–39, 2003.

[29] Y. Zhu, B. Li, and J. Guo, "Multicast with network coding in application-layer overlay networks," *Selected Areas in Communications, IEEE Journal on*, vol. 22, pp. 107–120, jan. 2004.

[30] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, (New York, NY, USA), pp. 1–12, ACM, 2009.

[31] M.-J. Montpetit, C. Westphal, and D. Trossen, "Network coding meets information-centric networking: An architectural case for information dispersion through native network coding," in *Proceedings of the 1st ACM Workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications*, NoM '12, (New York, NY, USA), pp. 31–36, ACM, 2012.

[32] M. Mrak, N. Sprljan, T. Zgaljic, N. Ramzan, S. Wan, and E. Izquierdo, "Performance evidence of software proposal for wavelet video coding exploration group," tech. rep., Tech. Rep. M13146, 76th MPEG Meeting, ISO/IEC JTC1/SC29/WG11/MPEG2005, Apr. 2006, Montreux, Switzerland, April 2006.

[33] N. Adami, A. Signoroni, and R. Leonardi, "State-of-the-art and trends in scalable video compression with wavelet-based approaches," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 17, pp. 1238–1255, sept. 2007.

[34] C. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, July, October 1948.

[35] F. Hekland, "A review of joint source-channel coding." available at `http://www.iet.ntnu.no/projects/beats/Documents/JSCC_review.pdf`, 2004.

[36] N. Ramzan, S. Wan, and E. Izquierdo, "Joint source-channel coding for wavelet-based scalable video transmission using an adaptive turbo code," *J. Image Video Process.*, vol. 2007, January 2007.

[37] M. van der Schaar and H. Radha, "Unequal packet loss resilience for fine-granular-scalability video," *Multimedia, IEEE Transactions on*, vol. 3, pp. 381–394, dec 2001.

[38] I. Boyarinov and G. Katsman, "Linear unequal error protection codes," *Information Theory, IEEE Transactions on*, vol. 27, pp. 168–175, mar 1981.

[39] S. Arslan, P. Cosman, and L. Milstein, "Concatenated block codes for unequal error protection of embedded bit streams," *Image Processing, IEEE Transactions on*, vol. 21, no. 3, pp. 1111–1122, 2012.

[40] Y. Xu and T. Zhang, "Variable shortened-and-punctured reed-solomon codes for packet loss protection," *Broadcasting, IEEE Transactions on*, vol. 48, no. 3, pp. 237–245, 2002.

[41] D. MacKay, "Fountain codes," *Communications, IEE Proceedings-*, vol. 152, pp. 1062 – 1068, dec. 2005.

[42] M. Luby, "LT codes," in *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pp. 271–280, 2002.

[43] A. Shokrollahi, "Raptor codes," *Information Theory, IEEE Transactions on*, vol. 52, pp. 2551–2567, june 2006.

[44] S. Puducheri, J. Kliewer, and T. Fuja, "The design and performance of distributed LT codes," *Information Theory, IEEE Transactions on*, vol. 53, no. 10, pp. 3740–3754, 2007.

[45] N. Rahnavard, B. Vellambi, and F. Fekri, "Rateless codes with unequal error protection property," *Information Theory, IEEE Transactions on*, vol. 53, pp. 1521–1532, april 2007.

[46] D. Vukobratovic and V. Stankovic, "Unequal error protection random linear coding for multimedia communications," in *Multimedia Signal Processing (MMSP), 2010 IEEE International Workshop on*, pp. 280–285, Oct.

[47] D. Vukobratovic, V. Stankovic, D. Sejdinovic, L. Stankovic, and Z. Xiong, "Scalable video multicast using expanding window fountain codes," *Multimedia, IEEE Transactions on*, vol. 11, no. 6, pp. 1094–1104, 2009.

[48] V. Goyal, "Multiple description coding: compression meets the network," *Signal Processing Magazine, IEEE*, vol. 18, pp. 74–93, sep 2001.

[49] R. Puri, K. Ramchandran, K. Lee, and V. Bharghavan, "Forward error correction (FEC) codes based multiple description coding for internet video streaming and multicast," *Signal Processing: Image Communication*, vol. 16, no. 8, pp. 745–762, 2001. Packet Video Communications.

[50] Y. Sanchez, T. Schierl, C. Hellge, T. Wiegand, D. Hong, D. D. Vleeschauwer, W. V. Leekwijck, and Y. L. Louédec, "Efficient HTTP-based streaming using scalable video coding," *Signal Processing: Image Communication*, vol. 27, no. 4, pp. 329 – 342, 2012. Modern Media Transport — Dynamic Adaptive Streaming over HTTP (DASH).

[51] H. Bahramgiri and F. Lahouti, "Robust network coding against path failures," *Communications, IET*, vol. 4, pp. 272–284, 12 2010.

[52] A. B. Carlson, P. B. Crilly, and J. C. Rutledge, *Communication Systems*. Mc Graw Hill, 2001.

[53] R. Dougherty, C. Freiling, and K. Zeger, "Linear network codes and systems of polynomial equations," *Information Theory, IEEE Transactions on*, vol. 54, pp. 2303–2316, may 2008.

[54] Z. Zhang, "Theory and applications of network error correction coding," *Proceedings of the IEEE*, vol. 99, pp. 406–420, March 2011.

[55] D. Silva, F. Kschischang, and R. Koetter, "A rank-metric approach to error control in random network coding," *Information Theory, IEEE Transactions on*, vol. 54, pp. 3951–3967, sept. 2008.

[56] M. Wang and B. Li, "R2: Random push with random network coding in live peer-to-peer streaming," *Selected Areas in Communications, IEEE Journal on*, vol. 25, pp. 1655–1666, december 2007.

[57] Y. Lin, B. Li, and B. Liang, "Differentiated data persistence with priority random linear codes," in *Distributed Computing Systems, 2007. ICDCS '07. 27th International Conference on*, p. 47, june 2007.

[58] W. Jing and H. Jiaqing, "Zeta: A novel network coding based P2P downloading protocol." `http://code.google.com/p/zetasim/`.

[59] Q. He, M. Ammar, G. Riley, H. Raj, and R. Fujimoto, "Mapping peer behavior to packet-level details: a framework for packet-level simulation of peer-to-peer systems," in *Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003. 11th IEEE/ACM International Symposium on*, pp. 71–78, 2003.

[60] S. McCanne and S. Floyd, "ns network simulator." `http://www.isi.edu/nsnam/ns/`.

[61] B. Waxman, "Routing of multipoint connections," *Selected Areas in Communications, IEEE Journal on*, vol. 6, no. 9, pp. 1617–1622, 1988.

[62] "Waxman random network topology generator." `http://www2.math.uu.se/research/telecom/software/stgraphs.html`.

[63] ITU-T Joint Video Team 16, "JSVM (Joint Source Video Model) reference software." `http://www.hhi.fraunhofer.de/de/kompetenzfelder/image-processing/research-groups/image-video-coding/svc-extension-of-h264avc/jsvm-reference-software.html`.

[64] N. Thomos and P. Frossard, "Network coding of rateless video in streaming overlays," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 20, pp. 1834–1847, dec. 2010.

[65] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," *SIGCOMM Comput. Commun. Rev.*, vol. 28, pp. 56–67, oct 1998.

[66] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer, "Raptor forward error correction scheme for object delivery." RFC 5053, October 2007.

[67] A. Shokrollahi and M. Luby, "Raptor codes," *Found. Trends Commun. Inf. Theory*, vol. 6, pp. 213–322, mar 2009.

[68] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder, "Raptorq forward error correction scheme for object delivery." RFC 6330, August 2011.

[69] J. Ohm, G. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards—including high efficiency video coding (hevc)," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1669–1684, 2012.

[70] P. Maymounkov and N. J. A. Harvey, "Methods for efficient network coding," in *In Allerton*, 2006.

[71] H. Shojania and B. Li, "Pushing the envelope: Extreme network coding on the gpu," in *Distributed Computing Systems, 2009. ICDCS '09. 29th IEEE International Conference on*, pp. 490–499, 2009.

[72] J. S. Plank, K. M. Greenan, and E. L. Miller, "Screaming fast Galois Field arithmetic using Intel SIMD instructions," in *FAST-2013: 11th Usenix Conference on File and Storage Technologies*, (San Jose), February 2013.

[73] J. S. Plank, E. L. Miller, and W. B. Houston, "GF-Complete: A comprehensive open source library for Galois Field arithmetic," Tech. Rep. UT-CS-13-703, University of Tennessee, January 2013.

[74] H. P. Arvin, "The mathematics of RAID-6." `https://www.kernel.org/pub/linux/kernel/people/hpa/raid6.pdf`, 2011.

[75] N. Thomos and P. Frossard, "Raptor network video coding," in *Proceedings of the International Workshop on Workshop on Mobile Video*, MV '07, (New York, NY, USA), pp. 19–24, ACM, 2007.

[76] H. A. Q. Maarif, T. S. Gunawan, A. U. Priantoro, H. Al, Q. Maarif, T. S. Gunawan, and A. U. Priantoro, "Complexity evaluation in scalable video coding," *Advances in Multimedia - An International Journal (AMIJ)*, vol. 1, pp. 1–25, May 2010.

[77] Z. Ma and Y. Wang, "Complexity modeling of scalable video decoding," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pp. 1125–1128, March 2008.

[78] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h.264/avc video coding standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, pp. 560–576, July 2003.

[79] "'ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG-4 AVC)", advanced video coding for generic audiovisual services."

[80] S. Yang, *Network Coding and Error Correction*. PhD thesis, The Chinese University of Hong Kong, 2008.

[81] S. Yang and R. W. Yeung, "Characterizations of network error correction/detection and erasure correction," in *in Proc. NetCod*, 2007.

[82] S. Yang, R. Yeung, and Z. Zhang, "Weight properties of network codes," *European Transactions on Telecommunications*, vol. 19, pp. 371–383, 2008.

[83] S. Yang, R. Yeung, and Z. Zhang, "Characterization of error correction and detection in a general transmission system," in *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, pp. 812–816, july 2008.

[84] S. Yang, R. Yeung, and C. K. Ngai, "Refined coding bounds and code constructions for coherent network error correction," *Information Theory, IEEE Transactions on*, vol. 57, pp. 1409–1424, march 2011.

[85] Y. Cui, Y. Xue, and K. Nahrstedt, "Optimal distributed multicast routing using network coding," in *Communications, 2007. ICC '07. IEEE International Conference on*, pp. 3610–3615, June 2007.

[86] Y. Xi and E. Yeh, "Distributed algorithms for minimum cost multicast with network coding," *Networking, IEEE/ACM Transactions on*, vol. 18, no. 2, pp. 379–392, 2010.

[87] J. Zou, H. Xiong, C. Li, L. Song, Z. He, and T. Chen, "Prioritized flow optimization with multi-path and network coding based routing for scalable multirate multicasting," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, pp. 259–273, March 2011.

[88] Y. Lin, B. Li, and B. Liang, "Stochastic analysis of network coding in epidemic routing," *Selected Areas in Communications, IEEE Journal on*, vol. 26, pp. 794–808, June 2008.

[89] M. Langberg, A. Sprintson, and J. Bruck, "Network coding: A computational perspective," *Information Theory, IEEE Transactions on*, vol. 55, pp. 147–157, jan. 2009.

[90] R. W. Yeung, S.-Y. R. Li, N. Cai, and Z. Zhang, "Network coding theory - part I: Single source," *Foundations and Trends in Communications and Information Theory*, vol. 2, no. 4, 2005.

[91] S. Jaggi, P. Sanders, P. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *Information Theory, IEEE Transactions on*, vol. 51, pp. 1973–1982, june 2005.

[92] N. J. A. Harvey, D. R. Karger, and K. Murota, "Deterministic network coding by matrix completion," in *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '05, (Philadelphia, PA, USA), pp. 489–498, Society for Industrial and Applied Mathematics, 2005.

[93] X. Guang, F.-W. Fu, and Z. Zhang, "Construction of network error correction codes in packet networks," in *Network Coding (NetCod), 2011 International Symposium on*, pp. 1–6, july 2011.

[94] R. Koetter and F. Kschischang, "Coding for errors and erasures in random network coding," *Information Theory, IEEE Transactions on*, vol. 54, pp. 3579–3591, aug. 2008.

[95] D. Silva and F. Kschischang, "On metrics for error correction in network coding," *Information Theory, IEEE Transactions on*, vol. 55, pp. 5479–5490, dec. 2009.

[96] R. Matsumoto, "Construction algorithm for network error-correcting codes attaining the singleton bound," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E90-A, pp. 1729–1735, September 2007.

Appendix A

Scalable Video Coding Architecture

This appendix briefly describes the H.264/SVC, the video codec used for our experiments. This architecture has been designed by the joint consortium of ITU-T and ISO/IEC JTC1, the first one using the name H.264, and the second one releasing the codec under the MPEG-4 suite, namely calling it MPEG-4 Part 10. SVC is the Annex G of the H.264/AVC codec, being thus an extension of the standard in use. Although alternatives exist, this remains the reference coding architecture for SVC in all research-related fields. Although the SVC acronym usually refers to the standard H.264/SVC, we've often used this word to the scalable coding paradigm applied to video data.

The SVC Extension of the H.264/AVC Standard

Fig. A.1 shows the structure of the H.264/SVC. Most of the coding tools of the architecture of the non scalable version, the H.264/AVC, are found in the scalable extension. This allows the SVC to produce a base layer which is compatible with AVC, and that can be decoded by legacy AVC decoders.

H.264 takes a block-based approach for spatial coding, with a two-stage integer transform — Discrete Cosine Transform (DCT) and Hadamard applied to the base DCT coefficients. The legacy I-, P- and B-slice distinction is also kept, where I-slices use only *intra-picture* spatial prediction, P-slices add *inter-picture* prediction from previous frames, and B-slides allow forward and backward inter frame prediction. Two entropy coding modalities are allowed: Context-based Adaptive Variable Length Coding (CAVLC) uses variable length codes and its adaptivity is restricted to transform coefficients coding. Context-based Adaptive Binary Arithmetic Coding (CABAC) uses more sophisticated adaptivity and arithmetic coding and reaches lower bitrates than CAVLC [4].

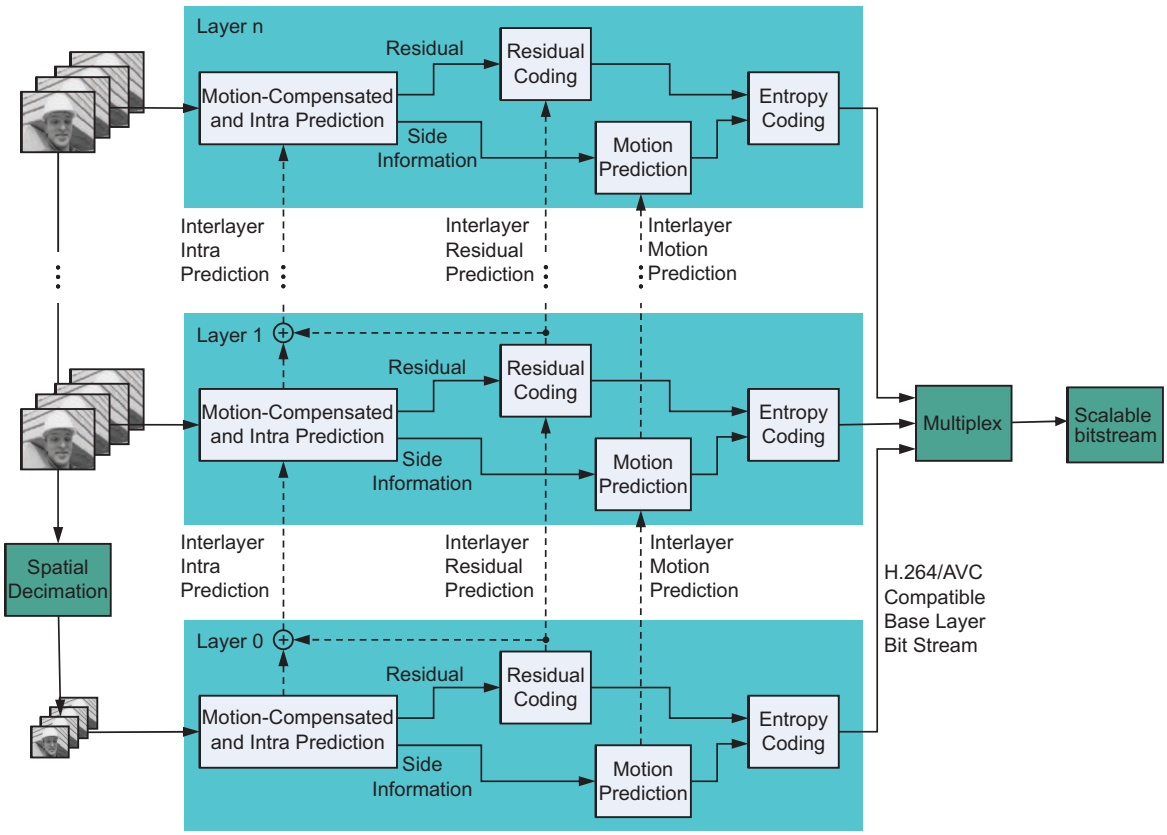

Fig. A.1: Block Diagram of H.264/SVC [4]

The legacy of AVC is present in SVC to allow retro-compatibility for the base layer but extends coding functionalities towards full-scalability, i.e., temporal, spatial and quality scalability. Temporal scalability comes directly from the hierarchical prediction structure of AVC. An example is shown in Fig A.2. The frame at the end of each GOP (both Predictive and Intra) may constitute the base temporal layer, providing a low frame rate, while the other Predictive (P) and Bi-predictive (B) frames may be part of enhancement layers for increasing the frame rate. Spatial and quality scalability is supported by a multilayer coding similar to the one used in H.263 where motion-compensated prediction and intra-prediction are employed in each layer. In addition, SVC provides so-called inter-layer prediction methods which allow an exploitation of the statistical dependencies between different layers. It is possible to vary the quantisation parameter across the layers in order to encode with better fidelity the lower layers.

Three important concepts have been implemented in SVC: key pictures, prediction of macroblock modes and associated motion parameters, and prediction of the residual signal. Improving over the prediction structures introduced for H.262, H.263, and H.264, the key

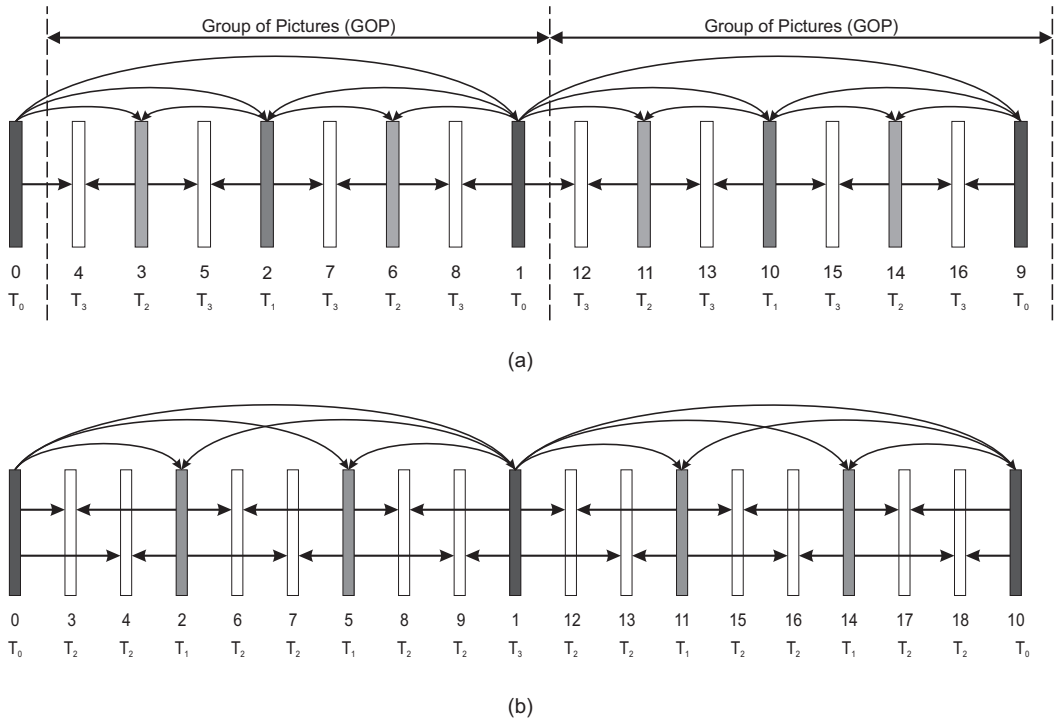

Fig. A.2: Hierarchical structure enabling temporal scalability. (a) coding with hierarchical B-frames. (b) Nondyadic structure with 2 subsequences at $1/3$rd and $1/9$th of the full rate. Number under pictures indicates the coding order indicated and T_k specifies the temporal layer.

pictures flag identifies those frames whose motion compensated prediction is performed based on the quality layer of the reference frame. This imposes that the motion parameters don't change between base and enhanced reconstruction, limiting the quality drifting of frames whose reference is the key frame, in case the enhancement layer of the reference frame for the key picture is not available. In inter-layer prediction the macroblock prediction signal is completely inferred from colocated blocks in the reference layer without transmitting any additional side information. When the colocated reference layer blocks are intra-coded, the prediction signal is built by the up-sampled reconstructed intra signal of the reference layer — a prediction method also referred to as inter-layer intra prediction. In inter-layer macroblock mode and motion prediction, the enhancement layer macroblock is inter-picture predicted as in single-layer coding, but the macroblock partitioning — specifying the decomposition into smaller blocks with different motion parameters — and the associated motion parameters are completely derived from the co-located blocks in the reference layer. This concept is also referred to as inter-layer motion prediction. Inter-layer residual prediction targets a reduction of the bit rate required for transmitting the residual

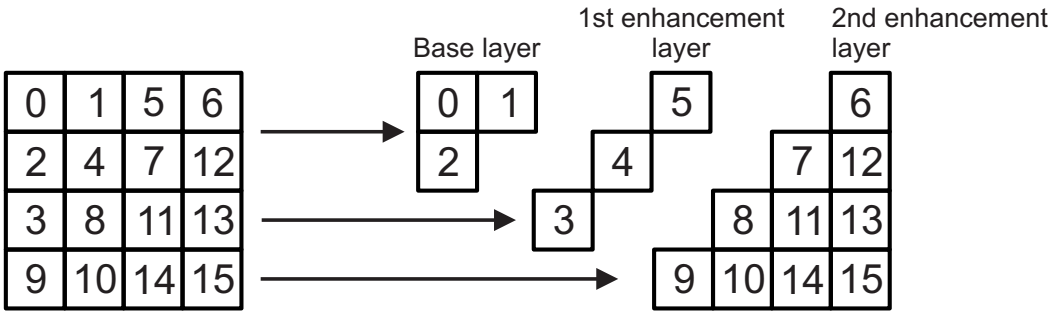

Fig. A.3: DCT coefficient partitioning for quality and spatial scalability.

signal of inter-coded macroblocks. With the usage of residual prediction the up-sampled residual of the colocated reference layer blocks is subtracted from the enhancement layer residual (difference between the original and the inter-picture prediction signal) and only the resulting difference is encoded using transform coding.

Quality scalability is treated as a special case of spatial scalability with identical picture sizes for base and enhancement layer. This case, which is also referred to as coarse-grain quality scalable coding (CGS), is supported by the general concept for spatial scalable coding as described above. Medium-grain quality scalability (MGS) uses a modified high-level signalling, which allows switching between different MGS layers in any access unit. Another mode, fine-grain quality scalable (FGS), performs motion compensation only using the base layer reconstruction as reference, and thus any loss or modification of a quality refinement packet does not have any impact on the motion compensation loop.

The H.264/AVC was designed to work on packet networks; the Network Abstraction Layer (NAL) provides flexible adaptation of the stream packet transmission via NAL units, containing coded data portions that can be packetised. NAL units that contribute to decoding one picture are considered to belong to the same coding unit.

When compared to the non-scalable standard, SVC adds little syntax repetition to the bit-stream. It does however yield to loss in coding efficiency and additional computational complexity. While coding of spatial layers is the main source of redundant entropy information, DCT segmentation for quality scalability (See Fig. A.3) and temporal scalability (See Fig. A.2) are a product of standard AVC tools. Reduced efficiency of entropy coding over sub-streams also accounts for overhead. Such overhead has been measured to be variable, but often reducible via rate optimisation to around 10% with respect to a single rate encoder [4]. Additional computational complexity comes however from the prediction structures envisioned for cross-layer dependency exploitation as well as drift control for loss resilience [76]. At the terminal side, the complexity increases because of the compensation

and prediction structure, and can be resumed as an exponential increase of complexity with the number of temporal and spatial layers, whereas it has a linear growth with the quality layers. For more details about the decoder complexity refer to [77]. It has to be noted that thanks to the fine hierarchical structure, even if some packets are lost most of the data can be always decoded at the receiver, increasing quality or resolution partially and allowing little waste of bit-stream.

Overview papers detailing the SVC and AVC standards can be found at [4] and [78], respectively. Full specification of the standards can be found at [79].

Appendix B

Network Coding Weights and Bounds

This appendix, together with the next one, aims at giving some extra definitions in the scope of network coding theory. We will limit ourselves to simple definitions, applicable to situations that we have encountered, namely linear coding with algebraic formulation, and single source multicasts.

Network error correction (NEC) was proposed by Cai and Yeung, to drive network coding mechanisms to recover erroneous symbols as well as lost packets with network error-correcting codes [18, 19]. Definitions given for traditional coding theory, such as coding distance, weight measures and coding bounds have been revisited with the premise of a coding operator defined by the network itself.

Let's consider a directed acyclic graph, where s is a unique source of information and T is a set of sink nodes. Let's also consider a source alphabet $\mathscr{C}$ whose codewords are in a coding space of dimension ω in Galois Field of size q: $\mathscr{F}_q$. The algebraic transmission model presented in Chapter 2 considers the input of errors as random alterations of the symbols on the edges and erasures as symbol cancellations. Error is introduced by means of an additive $1 \times |E|$ error vector $\mathbf{z}$ as:

$$\mathbf{y}_t^T = (\mathbf{x}^T A + \mathbf{z})(I-K)^{-1} B_t. \tag{B.1}$$

The error pattern $\rho_{\mathbf{z}}$ is defined as the cut set defined by $\mathbf{z}$, i. e. its non-zero components. Let's also define $F_t = (I-K)^{-1}B_t$, and $M_t = A * F_t$ as it is useful to define the network code weights in the next section. We consider a Linear Code Multicast (LCM) for receiver t, and l_t the maximum overall number of errors that can occur in the network by which the receiver t is still able to decode the source information. We additionally define δ_t as the redundancy

at the sink node t, given as:

$$\delta_t = \text{mincut}(t) - \dim(\mathscr{C}) = h_t - \omega. \tag{B.2}$$

In the following we present bounds, weights and coding distance used in NEC theory.

Weights measures of Linear Network Codes

Yang *et al.* introduced two classes of coding weights for the variables involved in the algebraic model of network channel: received vector $\mathbf{y}$, error vector $\mathbf{z}$ and message vector $\mathbf{x}$ [80–82].

A first class of measures are defined as follows. A weight measure for the *min-cut* of the error pattern is formulated as:

$$w_c^t(\mathbf{z}) = \text{mincut}_t(\rho_\mathbf{z}), \tag{B.3}$$

which entails the minimum cut-set of error pattern $\rho_\mathbf{z}$. For this it holds that $w_c^t(\mathbf{z}) \leq w_H(\mathbf{z})$, which is the Hamming weight in classic coding theory. For linear codes, the *rank* of an error pattern is:

$$w_r^t(\mathbf{z}) = \text{rank}_t(\rho_\mathbf{z}), \tag{B.4}$$

which is calculated as the rank of the matrix formed by the rows in F_t corresponding to the edges in $\rho_\mathbf{z}$. For this measure it holds that $w_r^t(\mathbf{z}) \leq w_c^t(\mathbf{z})$. Finally, given the classic definition of Hamming weight $w_H(\mathbf{z})$ as the number of nonzero elements in the vector $\mathbf{z}$, the *network Hamming weight* can be expressed as:

$$w_n^t(\mathbf{z}) = \min\{w_H(\mathbf{z}' : \mathbf{z}'F_t = \mathbf{z}F_t\}, \tag{B.5}$$

where the minimum is searched among all the error vectors $\mathbf{z}'$ that result in receiving the same word as for $\mathbf{z}$ when the transmitted word is $\mathbf{x} = \mathbf{0}$. For the network hamming weight it holds that $w_n^t(\mathbf{z}) \leq w_r^t(\mathbf{z})$.

Another class of weights has been defined in [83]: Network Hamming weight of the received vector $\mathbf{y}$:

$$W_t^{rec}(\mathbf{y}) = \min w_H(\mathbf{z}), \tag{B.6}$$

where the minimum is searched among all the error vectors $\mathbf{z}$ that result in receiving the word $\mathbf{y} = \mathbf{z}F_t$ at the receiver, which is identical to the definition given before. Network

Hamming weight of the error vector **z**:

$$W_t^{err}(\mathbf{z}) = W_t^{rec}(\mathbf{z}F_t). \tag{B.7}$$

Network Hamming weight of the message vector **x**:

$$W_t^{msg}(\mathbf{x}) = W_t^{rec}(\mathbf{x}M_t). \tag{B.8}$$

According to these definitions one can define the Hamming distance between received vectors, and between source vectors as:

$$D_t^{rec}(\mathbf{y}_1, \mathbf{y}_2) = W_t^{rec}(\mathbf{y}_1 - \mathbf{y}_2), \tag{B.9}$$

and

$$D_t^{msg}(\mathbf{x}_1, \mathbf{x}_2) = W_t^{msg}(\mathbf{x}_1 - \mathbf{x}_2), \tag{B.10}$$

respectively. It follows that the minimum distance can be defined as:

$$d_{\min,t} = min\left\{D_t^{msg}(\mathbf{x}_1, \mathbf{x}_2) : \mathbf{x}_1, \mathbf{x}_2 \in \mathscr{C}, \mathbf{x}_1 \neq \mathbf{x}_2\right\}. \tag{B.11}$$

We define a code as l error correcting if $d_{\min,t} \leq 2l+1$ and l error detecting if $d_{\min,t} \leq l+1$. In classic coding theory, erasure correction is performed when missing elements reduce the length of the codeword to be decoded. In coherent network coding, one could subsume erasures as a special case of errors, where symbols at the edges where erasures are happening are set to zero. In this case error correction theory applies. Alternatively, it is necessary to assume knowledge of the location of the erasures, in which case a whole new set of definitions and conclusions can be drawn, for which we refer to [80–82].

Linear Network Codes Bounds

Hamming bound, Singleton bound, and Gilbert-Varshamov bound define the relation between codebook size, field size, min-cut capacity and existence of a code with specific property [18, 19]. We give here some definitions as proposed in [84]. Definitions of these bounds were also given in [18, 19].

Consider a source codebook $\mathscr{C}$, with size $|\mathscr{C}| = q^{\omega}$. The Singleton Bound, for an LCM with minimum distance equal to $d_{\min,t}$, bounds the codebook size according to the following

definition:

$$|\mathscr{C}| \leq q^{h_t - d_{\min,t}+1}, \tag{B.12}$$

for all sink nodes t. This can me also formulated as which can also be formulated as:

$$d_{\min,t} \leq \delta_t + 1, \tag{B.13}$$

A Minimum Distance Separable (MDS) code can be defined as a LCM which verifies both Eq. B.12 and Eq. B.13 with the equality.

The Hamming Bound constitutes also an upper bound on the size of the source codebook, and was formulated in [84] as follows:

$$|\mathscr{C}| \leq \min_{t \in T} \frac{q_t^h}{\sum_{i=0}^{\tau_t} \binom{h_t}{i} (q-i)^i} \tag{B.14}$$

where $\tau_t = \lfloor \frac{d_{\min,t}-1}{2} \rfloor$.

These definitions, together with many that we don't mention here, can regard the NEC with the same characteristics of classic linear channel codes in terms of correction and detection of error and correction of erasures. We recommend to refer to the bibliography for detailed treatment of the topic.

Appendix C

Linear Network Code Construction and Algorithms

This appendix concludes a review of network coding theory and tools, initiated in Chapter 2 and continued in Appendix B, with a survey of network codes construction and algorithms.

Let's make a first distinction of transmission and construction approaches. These can be classified into *coherent* or *non-coherent*. Coherent transmission assumes that the sink nodes have knowledge of network topology and coding functions. By contrast when decoding takes place without this knowledge we refer to it as non-coherent transmission. Deterministic construction is usually performed for coherent transmission, and non-coherent transmission follows a randomised construction. However, it is possible to transmit and decode non-coherently with deterministically constructed codes, and use coherent transmission with randomly constructed codes, as long as the previous definitions of transmission hold. While deterministic construction is usually a centralised process, randomised approaches could be implemented both in a centralised and distributed way. However among these approaches, distributed randomised codes are more practical, allowing each node to autonomously and randomly chose the coding functions. We also differentiate between symbol-based transmission and packet-based transmission, the first one being feasible only with coherent transmission, whereas non-coherent transmission needs to communicate in-band the coding characteristic to the receiver.

We present now some algorithms for code construction, both coherent and non-coherent. We will consider the network as a directed and acyclic graph. Multicast routing with network coding has gained a lot of attention due to the important result that network coding can reduce the intractable problem of optimal multicast routing optimisation, to a tractable optimisation procedure. We refer to the following bibliography for further reading: [85–88].

Table C.1: Comparison of complexity of deterministic algorithms for network code construction. $|E|$ is the total number of edges in the network, $|V|$ the number of nodes and $|T|$ the number of sinks. $\mathscr{M}(h)$ is the time required to perform an $h \times h$ matrix multiplication. Langberg suggest that the correct running times of the algorithms of Jaggi *et al.* are different when the number of edges is small [89].

	Field size (q)	Time complexity
Yeung [90], Algorithm 2.19	$\binom{\|E\|+h-1}{h-1}$	$\mathscr{O}(\|E\|)$
Jaggi *et al.* [91] RLIF	$2\|T\|$	$\mathscr{O}(\|E\|\|T\|h^2)$
Jaggi *et al.* [91] DLIF	$\|T\|$	$\mathscr{O}(\|E\|\|T\|h(h+\|T\|))$
Jaggi *et al.* [91] fast	$2\|E\|\|T\|$	$\mathscr{O}(\|E\|\|T\|h+\|T\|\mathscr{M}(h))$
Langberg *et al.* [89]	$\|T\|$	$\mathscr{O}(\|E\|\|T\|^2h+h^4\|T\|^3(\|T\|+h))$
Harvey *et al.* [92]	$\|T\|$	$\mathscr{O}(\|T\|h^3\log h)$

Coherent Construction Approaches

Deterministic coherent construction can be performed via the *flow-path* approach, firstly proposed for generic codes by Li *et al.* and later improved for multicast by Jaggi *et al.* in their preservative design [91] or via *matrix-completion* proposed by Harvey *et al.* [92] within the scope of matroids theory applied to network codes. A comparison of these algorithms by required field size and complexity is presented in Table C.1.

The approach by Jaggi *et al.* [91] constructs a single-source LCM with linear independence of the coding vectors for a specific set of nodes T. A cut set of h paths from source to sink is considered, then the global encoding vectors are assigned to form a base for the coding space. The algorithm proceeds then to considers all the edges one at a time in topological order for the calculation of the local encoding vectors and ensures the preservation of linear independence until the edges that feed into the receiver are considered. When choosing a base for each cut-set of edges, there are two options: a semi-randomised procedure assigns the kernels randomly first and test the linear independence. This approach is known as the Randomised Linear Information Flow (RLIF). Another approach calculates the local encoding kernels with an algebraic operation, as it is know as Deterministic LIF (DLIF). A fast approach was also proposed where the coefficients of all edges can be chosen randomly, and the rank of the resulting transfer matrix is calculated [91]. The execution times of these algorithms are compared in Table C.1.

Another deterministic approach, which belongs to the field of matroid theory, consider the equivalence between the network graph and a matroid, and then apply independent matching of matroids to transfer matrices [92]. This algorithm can be also applied to multiple-source multicast problems as opposed to the aforementioned algorithms which work for single-source networks. The complexity of the algorithm in the single source multicast version is compared with the other algorithms in Table C.2.

Coherent Network Error Correction

Network Error Correction code construction can be performed with flow path approaches. The approaches discussed in this section are compared in Table C.2, again in terms of required field size and computational time. Consider that the formulae in table C.2 have been adapted to fit a simplified notation with equal parameters for all receivers. E.g., the formula for Guang *et al.* 's algorithm [93] considers the complexity of constructing the code for up to a number of failures, whereas we reformulate for a maximum of $d_{min} - 1 = h - \omega$ errors.

Yang *et al.* proposed two completely deterministic algorithms achieving the Refined Singleton bound, including unequal flows to the sinks [84]. The algorithms design both network code and codebook to achieve at each receiver the minimum distance $d_{min,t} = \text{mincut}(t) - \omega + 1$. The first algorithm determines the network coding kernels first and then constructs an appropriate codebook. The encoding kernels spanning the coding space are chosen, then the codebook is chosen using the basis for the network code that keep the minimum coding distance [84]. The second algorithm determines a codebook by means of traditional block-code generation and then constructs the network code iteratively, one edge at a time in up-stream to down-stream topological order. Failure patterns are considered, and the kernels are chosen among those that allow the flow to sustain the source rate. This technique is a generalisation of the linear independence of the preservative design. The processing load and the required field size of these two algorithms are compared in Table C.2.

Matsumoto proposed another centralised and deterministic algorithm, which builds an extended network with imaginary nodes feeding the regular edges and expanding the number of edge disjoint paths to the sinks. To let the new paths be independent from the regular ones, the Jaggi's preservative algorithm mentioned in the previous section is run on the extended network. This algorithm possesses similar requirements to Yang's algorithms, as showed in Table C.2.

Bahramgiri and Lahouti also proposed some variants of construction algorithms in line with the design by Jaggi *et al.* [51]. Three version are proposed: A first scheme called

Robust Network Coding 1 (RNC1) chooses the local kernels with a criterion of partial linear independence. All ω-subsets of the paths from source to sink have to be independent. A random subroutine is considered in this case, and the linear independence is verified. When $\text{mincut}(t) - \omega$ paths fail the receiver can always recover ω information symbols, but cannot cope with alteration of paths intended to other sinks, thus the scheme is successful only in case of erasures in a unicast scenario. A second scheme (RNC2) considers a subset of paths which have any edge in common to any other path to other sinks. The scheme increase the complexity by a factor equal to the number of paths with joint edges, but copes with all path failures in case of erasures. A third scheme (RNC3) considers the use of backup paths. In all construction algorithms $\text{mincut}(t)$ edge-disjoint paths are considered from source to sink, not considering edges in other possible paths. Backup paths superpose with the edge-disjoint and use inactive edges as a backup in case any of the other paths fail. At each step of the algorithm, the partial independence is verified among ω-subsets of all possible paths. This scheme increases the complexity by a factor equal to the number of possible ω-subsets of paths. The three algorithms are compared in Table C.2 and discussed in detail in [51].

Another greedy algorithm was proposed by Guang *et al.* [93] where the linear independence of local encoding kernels is guaranteed for all error patterns, either by testing with linear independence or with a deterministic implementation, similarly to Jaggi *et al.* 's RLIF and DLIF criteria.

Noncoherent Network Coding

In situations that require a distributed code construction, randomised coding provides an efficient solution. Nodes select the coding coefficients for the outgoing packets randomly and independently. The most common approach to produce a decodable stream is to attach unitary vectors to packets at the source as a pre-payload. When encoded together with the actual payload, receivers will be able to deduce the coding kernels and decode the messages. This approach was thoroughly discussed in Chapter 2.

We analysed the case in which a randomised choice of the local encoding kernel produces a random matrix channel. The probability of having a random code with desired characteristics depends on many factors. A sufficiently large base field is usually enough to ensure the existence of the code and the possibility of successfully decoding the transmission [22]. Additionally, if the codebook has a degree of redundancy $\delta_t = \text{mincut}(t) - \omega, \forall t \in T$, where $\omega = \dim(\mathscr{C}) \leq h$, decoding can be succesful in presence of link failures or errors with increasing probability as the GF size increases [23].

The probabilities of successful construction with or without degradation, and the required field size for the existence of such code are resumed in Table C.3.

Another solution came with the introduction of *subspace coding*, where the receivers perform an algebraic analysis of the space spanned by the received packets and deduce information about the encoded source information, or better, the vector space injected into the network at the encoder [94]. A class of codes that conform to the subspace design have been studied using rank distance metrics and exploiting partial knowledge of network degradation in terms of erasures (knowledge of error location but not the value) and deviations (knowledge of error value but not the location) [55]. Their approach with rank-metric is based on Gabidulin codes which is analogous to the subspace metric approach with Reed-Solomon codes. Also in case of adversarial error injections, an *injection* metric has been proposed for a better design of non-constant-dimension codes than the subspace metric [95].

To conclude this appendix: Decoding by matrix inversion has an advantage over subspace coding against random channel errors. On the other hand the performance for erasure correction is the same for both approaches. Moreover, rank-metric codes can achieve higher performance against attacks from malicious nodes since there's no need to record the encoding kernels in the header. Finally, centralised and deterministic approaches to code construction are seemingly impractical because such complete knowledge of the network may not be available at any instant.

Table C.2: Comparison of centralised algorithms for single source multicast network error correction code construction. $|E|$ is the total number of edges in the network, $|T|$ the number of sinks, d is the minimum distance at all sinks, h is the multicast rate, n_s the number of outgoing channels from the source. In Bahramgiri and Lahouti's algorithms: $R^{(d)}$ is the set of failure patterns with at most $d-1$ failures of edges in common with other paths, Ω includes all possible sets of paths [51].

	Field size (q)	Time complexity
Yang *et al.* [84], 1	$\|T\|\binom{\|E\|}{d-1}$	$\mathscr{O}\left(n_s\|T\|\omega\binom{\|E\|}{d-1}\left(n_s^2+\|T\|\binom{\|E\|}{d-1}\right)\right)$
Yang *et al.* [84], 2	$\|T\|\binom{h+\|E\|-2}{d-1}$	$\mathscr{O}\left(h\|T\|^2\|E\|\binom{h+\|E\|-2}{d-1}\left(h^2\|T\|^2+\|T\|\binom{h+\|E\|-2}{d-1}\right)+h^3d\|E\|\|T\|\binom{\|E\|}{d-1}\right)$
Matsumoto [96]	$\|T\|\binom{\|E\|}{d-1}$	$\mathscr{O}\left(h\|E\|\|T\|\binom{\|E\|}{d-1}\left(\|T\|\binom{\|E\|}{d-1}+h+d\right)\right)$
Bahramgiri and Lahouti [51], 1	$\|T\|\binom{h-1}{h-d}$	$\mathscr{O}\left(\|E\|\|T\|(n-d+1)^3\binom{h-1}{h-d}\left(1-\binom{h-1}{h-d}\frac{\|T\|}{q}\right)^{-1}\right)$
Bahramgiri and Lahouti [51], 2	$\|T\|\|R^{(d)}\|\binom{h-1}{h-d}$	$\mathscr{O}\left(\|E\|\|T\|\|R^{(d)}\|(n-d+1)^3\binom{h-1}{h-d}\left(1-\binom{h-1}{h-d}\frac{\|T\|\|R^{(d)}\|}{q}\right)^{-1}\right)$
Bahramgiri and Lahouti [51], 3	$\|T\|\prod_T\|\Omega\|$	$\mathscr{O}\left(\|E\|\|T\|(n-d+1)^3\prod_T\|\Omega\|\left(1-\frac{\|T\|\prod_T\|\Omega\|}{q}\right)^{-1}\right)$
Guang *et al.* [93], 1	$\|T\|\binom{\|E\|}{d-1}$	$\mathscr{O}\left(\|E\|(\omega+d-1)\|T\|\binom{\|E\|}{d-1}\left(1+\omega+\frac{\|E\|+1}{2}\right)\right)$
Guang *et al.* [93], 2	$\|T\|\binom{\|E\|}{d-1}$	$\mathscr{O}\left(\|E\|\|T\|\binom{\|E\|}{d-1}\left(\|T\|\binom{\|E\|}{d-1}(\omega+\frac{\|E\|+1}{2})+\omega+d-1\right)\right)$

Table C.3: Lower bounds for success probability of randomized coding and the required field size. Upper table shows the success probability, i.e., the probability of having successful transmission with coding redundancy ($\delta_t = mincut(t) - dim(\mathscr{C}), \forall t \in T$) with and without degradation in the network up to d [23]. The lower table shows the success probability of generating a Minimum Distance Separable (MDS) code with distance $D_{min,t} = mincut(t) - dim(\mathscr{C}) + 1$, with and without codebook redundancy [93]. J is the set of internal nodes.

	without degradation ($d = 0$)	with degradation ($d > 0$)	
$Pr(D_{min,t} \leq \delta_t + 1 - d, \forall t \in T)$	$1 - \sum_{t \in T} \frac{\binom{\delta_t + \lvert J \rvert + 1}{\lvert J \rvert}}{(q-1)^{\delta_t + 1}}$	$1 - \sum_{t \in T} \frac{\binom{\lvert E \rvert}{\delta_t - d}\binom{d + \lvert J \rvert + 1}{\lvert J \rvert}}{(q-1)^{d+1}}$	[23]
Field size q	$\sum_{t \in T} \binom{\lvert E \rvert}{\delta_t}$	$1 - \left(\sum_{t \in T} \binom{\lvert E \rvert}{\delta_t - d}\binom{d + \lvert J \rvert + 1}{\lvert J \rvert} \right)^{\frac{1}{d+1}}$	

	without redundancy ($\delta_t = 0$)	with redundancy ($\delta_t > 0$)	
MDS, $Pr(D_{min,t} \leq \delta_t + 1, \forall t \in T)$	$\left(1 - \frac{\lvert T \rvert}{q-1}\right)^{\lvert J \rvert + 1}$	$\left(1 - \sum_{t \in T} \frac{\binom{\lvert E \rvert}{\delta_t}}{q-1}\right)^{\lvert J \rvert + 1}$	[93]
Field size q	$\lvert T \rvert \lvert E \rvert$	$\sum_{t \in T} \binom{\lvert E \rvert}{\delta_t}$	