NETWORK CODING
FOR
TRANSPORT PROTOCOLS

Steluta Gheorghiu

Advisor: Alberto Lopez Toledo, PhD
Co-Advisor: Pablo Rodriguez, PhD
Telefonica Research
Barcelona, Spain

Tutor: Prof. Jorge Garcia Vidal, PhD
Technical University of Catalonia
Department of Computer Architecture

A Thesis presented to the Technical University of Catalonia
in partial fulfillment of the requirements for the degree of
Doctor in Computer Science

July 2011

ACTA DE QUALIFICACIÓ DE LA TESI DOCTORAL

Reunit el tribunal integrat pels sota signants per jutjar la tesi doctoral:
Títol de la tesi: **Network Coding for Transport Protocols**
Autor de la tesi: **Steluta Gheorghiu**

Acorda atorgar la qualificació de:

- ☐ No apte
- ☐ Aprovat
- ☐ Notable
- ☐ Excellent
- ☐ Excellent Cum Laude

Barcelona, ... de/d'..... de .....

El President	El Secretari	
.....	.....	
(nom i cognoms)	(nom i cognoms)	
El vocal	**El vocal**	**El vocal**
.....	.....	.....
(nom i cognoms)	(nom i cognoms)	(nom i cognoms)

Abstract

With the proliferation of smart devices that require Internet connectivity anytime, anywhere, and the recent technological advances that make it possible, current networked systems will have to provide a various range of services, such as content distribution, in a wide range of settings, including wireless environments. Wireless links may experience *temporary losses*, however, TCP, the de facto protocol for robust unicast communications, reacts by *reducing the congestion window drastically* and *injecting less traffic in the network*. Consequently the wireless *links are underutilized* and the overall performance of the TCP protocol in wireless environments is poor. As content delivery (i.e. multicasting) services, such as BBC iPlayer, become popular, the network needs to support the *reliable transport of the data* at *high rates*, and *with specific delay constraints*. A typical approach to deliver content in a scalable way is to rely on peer-to-peer technology (used by BitTorrent, Spotify and PPLive), where users share their resources, including bandwidth, storage space, and processing power. Still, these systems suffer from the *lack of incentives for resource sharing and cooperation*, and this problem is exacerbated in the presence of *heterogenous users*, where a tit-for-tat scheme is difficult to implement.

Due to the issues highlighted above, current network architectures need to be changed in order to accommodate the users' demands for reliable and quality communications. In other words, the emergent need for advanced modes of information transport requires revisiting and improving network components at various levels of the network stack.

The innovative paradigm of network coding has been shown as a promising technique to change the design of networked systems, by providing a shift from how data flows traditionally move through the network. This shift implies that data flows are no longer kept separate, according to the "store-and-forward" model, but they are also processed and mixed in the network. By appropriately combining data by means of network coding, it is expected to obtain significant benefits in several areas of network design and architecture.

In this thesis, we set out to show the benefits of including network coding into three communication paradigms, namely *point-to-point* communications (e.g. unicast), *point-to-multipoint* communications (e.g. multicast), and *multipoint-to-multipoint* communications (e.g. peer-to-peer networks). For the first direction, we propose a network coding-based multipath scheme and show that TCP unicast sessions are feasible in highly volatile wireless environments. For *point-to-multipoint* communications, we give an algorithm to optimally achieve all the rate pairs

from the rate region in the case of degraded multicast over the combination network. We also propose a system for live streaming that ensures reliability and quality of service to heterogenous users, even if data transmissions occur over lossy wireless links. Finally, for *multipoint-to-multipoint* communications, we design a system to provide incentives for live streaming in a peer-to-peer setting, where users have subscribed to different levels of quality.

Our work shows that network coding enables a reliable transport of data, even in highly volatile environments, or in delay sensitive scenarios such as live streaming, and facilitates the implementation of an efficient incentive system, even in the presence of heterogenous users. Thus, network coding can solve the challenges faced by next generation networks in order to support advanced information transport.

Resumen

Con la proliferación de dispositivos inteligentes que requieren conectividad a Internet en cualquier momento y en cualquier lugar, y los recientes avances tecnológicos que hacen que sea posible, los sistemas de red actuales tendrán que proporcionar una gama variada de servicios, tales como distribución de contenido, en una gama amplia de entornos, incluyendo entornos inalámbricos. Los enlaces inalámbricos pueden sufrir *pérdidas temporales*, sin embargo, TCP, el protocolo de facto para las comunicaciónes unicast robustas, reacciona con *la reducción drástica de la ventana de congestión* y *la inyección de menos tráfico en la red*. En consecuencia, *los enlaces inalámbricos son subutilizados* y el rendimiento global del protocolo TCP en estos entornos es bajo. Los servicios de distribución de contenidos (es decir, multicast), como BBC iPlayer, son cada vez mas populares, y la red tiene que soportar el *transporte fiable de los datos a altas velocidades, y con requerimientos específicos de retraso*. Un enfoque típico para entregar el contenido de una manera escalable es confiar en la tecnología peer-to-peer (utilizada por BitTorrent, Spotify y PPLive), donde los usuarios comparten sus recursos, incluyendo el ancho de banda, espacio de almacenamiento y potencia de procesamiento. Sin embargo, estos sistemas peer-to-peer sufren de la *falta de incentivos para el intercambio de recursos y la cooperación*, y este problema se agrava en presencia de los *usuarios heterogéneos*, donde un esquema de "tit-for-tat" es difícil de aplicar.

Debido a los problemas señalados anteriormente, las arquitecturas actuales de red necesitan ser cambiadas para adaptarse a las demandas de los usuarios en materia de comunicaciones fiables y de calidad. Es decir, la necesidad emergente de modos avanzados de transportar la información requiere revisar y mejorar los componentes de red en distintos niveles de la pila de protocolos de red.

El paradigma innovador de network coding se ha demostrado como una técnica prometedora para cambiar el diseño de sistemas en red, proporcionando un cambio en el modo en que los flujos de datos tradicionalmente se mueven a través de la red. Este cambio implica que los flujos de datos ya no están separados, siguendo el modelo "store-and-forward", sino que también se procesan y se combinan en la red. Si la combinación de datos a través de network coding se aplica de una manera adecuada, se espera obtener beneficios significativos en varias áreas de diseño y arquitectura de red.

En esta tesis, nos propusimos mostrar los beneficios de la inclusión del network coding en tres paradigmas de comunicacion, en concreto, comunicaciones *punto-a-punto* (unicast), comunicaciones *punto-a-multipunto*

(multicast), y la comunicacion *multipunto-a-multipunto* (redes peer-to-peer). Para la primera direccion, proponemos un esquema basado en network coding que utiliza caminos múltiples, y que hace las sesiones TCP unicast viables en entornos inalámbricos altamente volátiles. Para las comunicaciones *punto-a-multipunto*, presentamos un algoritmo para alcanzar de manera óptima todos los pares de velocidades de transmisión de la región de velocidades posibles, en el caso de multicast sobre la red de combinación. También proponemos un sistema de transmisión de video en vivo que garantiza la fiabilidad y la calidad de servicio a los usuarios heterogéneos, incluso si se producen transmisiones de datos sobre enlaces inalámbricos con pérdidas. Por último, para las comunicaciones *multipunto-a-multipunto*, introducimos un sistema para incentivar la transmisión de video en vivo en una red peer-to-peer, donde los usuarios se han suscrito a diferentes niveles de calidad.

Nuestro trabajo muestra que network coding permite un transporte fiable de datos, incluso en entornos de alta volatilidad, o en escenarios sensibles a retrasos como la transmisión en vivo, y facilita la implementación de un sistema eficaz de incentivos, incluso en presencia de los usuarios heterogéneos. Por lo tanto, network coding puede resolver los desafíos a los que las redes de próxima generación se enfrentan con el fin de apoyar el transporte avanzado de información.

Contents

List of Algorithms

List of Figures

List of Tables

Acknowledgements

Pursuing a PhD degree is a challenging and exciting journey, and like in all journeys, the people that you travel with, make the experience much more enriching and enjoyable. I would like to acknowledge here the people who accompanied me during this journey.

First of all, I would like to express my gratitude to my advisors, Pablo Rodriguez and Alberto Lopez Toledo, for their support and guidance through this process. Alberto taught me many small lessons about research, and encouraged me whenever he felt appropriate. Pablo taught me that focusing on the bigger picture, while gaining a deep understanding of a problem, are key for a successful researcher.

I would like to thank Jorge Garcia, my tutor from UPC, for his patience and his help to solve the administrative problems. Pablo and Jorge were the ones who made possible for me to do my PhD in the lab at Telefonica Research – and I will always be indebted to them for this opportunity. I am most grateful to Jose M. Barcelo for his generous and altruistic help; although I have been working with Jose M. only for a short period of time during my first months at UPC, he continued to give me advice whenever I asked him.

I have been fortunate to work with Christina Fragouli, and I thank her for the chance she gave me to do an internship with the ARNI group at EPFL. My stay at EPFL was a great experience and one person whom I would like to thank for this is Shirin Saeedi, my office mate during that period of time. Each of the collaborations that I had during my studies, helped me improve my research skills, and I would like to acknowledge the people with whom I worked and I have not mentioned previously: Luísa Lima, João Barros, Muriel Médard, Christos Gkantsidis, Wenjun Hu, Bozidar Radunovic, Peter Key, David Fusté Vilella, Johann Lopez.

From my pre-PhD journey, I have to mention two persons. Carmen Delcea played an important role during the last year of high school, and had a major

influence on my decision to study Computer Science. Prof. Theodor Borangiu, my advisor for the diploma project at University "Politehnica" of Bucharest, helped me in a key moment and supported my decision to pursue a PhD degree.

Big organizations would not function without the hard work and dedication of people like Anna Jordana (Telefonica Research), Trinidad Carneros (UPC), and Françoise Behn (EPFL).

I would like to thank the people that I met during my stay with Telefonica Research lab: Rade, for his friendship; Georgos, Nikos and Josep, for the postcards they brought me from their travels, for my collection; Meeyoung, Xiao, Javi and Narseo with whom I shared many special moments during my first months in the lab; Tomasz, Karen, Domenico, Parminder, Michalis, Xavier (Anguera), Xavier (Amatriain), Rodrigo, Joachim, Pere, for being great colleagues. "Moltes gràcies" to Eduard for his prompt help on many issues.

I would like to thank my friends from UPC – Liliana, Ruken and Lori for being available for coffee breaks during my short visits to UPC. I am grateful to my close friends from home, to Cristina, for sharing with me the good and the bad moments, to Adina and Gabriel, for saying a little prayer for me every now and then.

I feel blessed for having a big family. Special thanks to Mihaela, for always being optimistic, for always being an amazing friend. I am most grateful to mom, dad, my brother and his family, for their love and continuous support.

Finally, I would like to thank Costin – I started this journey because of him.

CHAPTER 1

Introduction

With the proliferation of smart devices that require Internet connectivity anytime, anywhere, and the recent technological advances that make it possible, current networked systems will have to provide a various range of services, such as content distribution, in a wide range of settings, including wireless environments.

TCP is the de facto protocol for reliable communications in unicast sessions, and it has been optimized for wired networks. TCP offers robustness through its mechanism to control congestion – which is the typical cause of packet loss in wired networks. However, in wireless environments the link quality varies in time due to sporadic effects such as fading or shadowing, thus causing *temporary losses*. Consequently, *TCP erroneously reduces the window size* (therefore injecting less traffic into the network), following with the congestion avoidance mechanism, finally leading to an *underutilization of the wireless links*.

With the increasing popularity of content delivery (i.e. multicasting) services, such as BBC iPlayer [iPl] and Netflix [Net], the network needs to *support the reliable transport* of the data at *high rates*, and to provide *tight delay guarantees*. A typical approach to distribute content in a scalable way is to rely on peer-to-peer technology, introduced by Napster [Nap] in 1999. In peer-to-peer networks, users share their resources, including bandwidth, storage space, and processing power to efficiently deliver the content of interest, without the need for central coordination. This technology has been used by numerous systems, out of which we mention BitTorrent [Bit], Spotify [Spo], and PPLive [PPL]. One of the problems of peer-to-peer is that of free-riding – users that take ad-

vantage of the system's resources, but do not contribute their own resources for the common good. Implementing a strict tit-for-tat policy as BitTorrent has a limited effect, especially for live content or in the presence of heterogenous users. Therefore, an efficient peer-to-peer system should account for *incentives for resource sharing and cooperation*.

Due to the issues highlighted above, current network architectures need to be changed in order to accommodate the users demands for reliable and quality communications. In other words, the emergent need for advanced modes of information transport requires revisiting and improving network components at various levels of the network stack.

The concept of network coding was introduced in the seminal paper [ALY00], where the authors showed that by allowing the nodes not only to forward packets, but also to mix them by means of algebraic operations, the total throughput delivered in a multicast setting is optimal; in addition, the optimal throughput can only be achieved by means of network coding. The result has drawn the attention of the research community and several other work soon followed, opening exciting directions for the use of network coding to provide significant benefits at various stages of network design.

Motivated by these observations, in this thesis, we focus on three main communication paradigms:

- *point-to-point communications*, where the data are transported between a source node and a receiver node, e.g. unicast sessions,
- *point-to-multipoint communications*, where a source node transmits data to multiple destination nodes, e.g. multicast sessions,
- *multipoint-to-multipoint communications*, where any node transmits data to any other node in the network, as it is the case of peer-to-peer networks,

and embark on to investigate what are the gains and advantages that network coding could bring for data transport in each case. For a quick glance at the research directions pursued in this thesis, we refer the reader to *Figure 1.1*. In particular, we set out to answer the following research questions:

- How can network coding be used to increase throughput and reliability of TCP communications in wireless settings, when data is transfered from a source node to a destination node (i.e. point-to-point transfer)? What is the impact of implementing such a network coding scheme into practice?

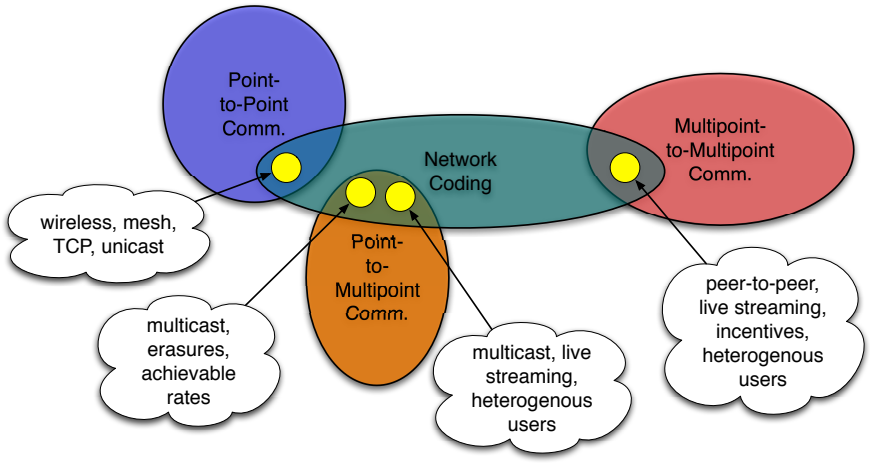


Figure 1.1: Thesis overview – a quick glance.

- Having knowledge about the network topology, how to assign the available resources to optimally achieve all the rate pairs in the rate region in a degraded multicast setting (i.e. when transport occurs from a sender to multiple receivers)? How well can we perform if the information about the underlying topology is not available?

- How to provide differentiated service to distinct users in a multicast wireless setting (i.e. where a source node streams video to several destinations)? What are the gains from using network coding in this context?

- How can network coding be used to efficiently provide incentives in peer-to-peer networks (i.e. multipoint-to-multipoint transport) where participating users have heterogenous requirements, while ensuring quality of service?

In this chapter, we present an outline of the thesis in *Section 1.1*, followed by a summary of the main contributions in *Section 1.2*.

1.1 An overview of the thesis

We start by giving a brief summary of the network coding concepts used throughout this thesis and review the relevant literature in *Chapter 2*. Next, we focus on the three areas identified above, as follows.

Point-to-point communications

In this area, we are interested in the benefits that network coding could bring for *wireless transmissions*. There has been extensive work that aims to improve the forwarding performance in wireless multihop networks. Techniques such as multipath, local retransmissions and network coding have been successfully proposed to address performance issues, increasing throughput and reducing losses. However, while minimizing losses and increasing throughput suffices for UDP traffic, these techniques often introduce other side effects such as packet reordering and delay that heavily affect TCP traffic. In particular, TCP simply does not work in the case of high losses or volatility.

In order to address this problem, in *Chapter 3* we introduce *CoMP*, a network coding multipath forwarding scheme that improves reliability and performance of TCP sessions in wireless mesh networks. Our protocol leverages the intrinsic characteristics of network coding and exploits the wireless mesh path diversity. *CoMP* performs congestion control and uses a credit-based method to control the rate at which linear combinations are generated in the network. Our scheme uses a simple algorithm to estimate losses and to send redundant linear combinations in order to maintain the decoding delay at a minimum and to prevent TCP timeouts and retransmissions. Furthermore, the network coding block of our proposal accounts for two situations: i) that when the sender has feedback on the rate at which the destination receives innovative linear combinations – which implies small changes at the TCP layer and therefore is not easy to implement, and ii) that when the sender has feedback on the rate at which the destination can actually decode linear combinations – which does not need any change to the TCP layer and can be readily implemented into practice. We evaluate *CoMP* through extensive simulations for a realistic topology and compare it to state-of-the-art protocols. We show that *CoMP* not only achieves a higher throughput, but also is more efficient than existing protocols, making TCP sessions feasible for wireless mesh networks even under heavy losses.

Point-to-multipoint communications

In this area, we analyze the case of *degraded multicasting* – that is, the case where different users require different subsets of the source content. In *Chapter 4*, we give a characterization of the rate region for the degraded two message set problem, applied to a combination network with erasure channels. We also provide an algorithm that uses topological information in order to deliver the

two messages to the receivers, and we show that our algorithm is optimal, in the sense that it achieves any rate pair in the region. We compare our algorithm analytically with a naive approach oblivious to the network structure, and we give an insight on what benefits should be expected for different classes of networks.

In *Chapter 5* we take a step further and take a closer look to an application of *degraded multicasting* – that of live streaming to users with diverse requirements. Scalable video can be used to provide video streaming reliably to an heterogeneous set of receivers with different subscription levels. However, the performance of such schemes can be highly affected by scheduling constraints and unreliable feedback. Network coding, on the other hand, has been shown to reduce scheduling and prioritization problems and to perform well in wireless scenarios with perfect feedback. Motivated by this observation, we implement and analyze a system architecture for network coding-based multiresolution video streaming in a wireless environment. In contrast to existing work, we take into account realistic feedback, where the control packets are sent over the same unreliable channel as data packets, and compare it to the case of perfect feedback, where the server has perfect knowledge of the state of the buffer at every receiver. We provide an evaluation of the system via simulation and show that even in highly volatile environments, a network coding-based scheme with limited and unreliable feedback can achieve a good performance.

Multipoint-to-multipoint communications

In this case, we focus on providing incentives for live streaming scenarios to heterogenous users. To this end, in *Chapter 6* we consider the design of an efficient streaming system for live video over peer-to-peer networks. Such a system must accommodate for large populations of heterogeneous users, behave robustly irrespective of user dynamics and ensure prescribed levels of quality of experience. Seeking a solution capable of addressing also the lack of incentives for peer-to-peer collaboration and the scarcity of certain video segments, we present a network coding based scheme for layered peer-to-peer live streaming. The key idea is to use layered coding as a tool to generate incentives and combine it with a specific network coding scheme to achieve efficient scheduling and delay minimization. The proposed solution is shown to achieve a short decoding delay, which further enables a smooth playback, while ensuring that users' decoding rate is proportional to their contribution to the system.

We conclude in *Chapter 7* with some final remarks and open directions for research.

1.2 Main contributions

The main contributions of this thesis are as follows:

- **a network coding multipath forwarding scheme, *CoMP*, to improve the performance of point-to-point transport with TCP for wireless environments**, in *Chapter 3*. *CoMP* performs error control in order to adjust online the rate of sending coded packets, uses a hybrid credit-based approach to balance the load on the available paths, and relies on a backpressure mechanism to control congestion. In addition, our scheme accounts for two situations: i) the sender has knowledge on the *rate of receiving linear combinations* at destination – which requires changes to the TCP layer and cannot readily be implemented, and ii) the sender has knowledge on the *rate of decoding linear combinations* at destination – which lends to a practical implementation. We show that *CoMP* achieves a higher throughput and is more efficient than existing state-of-the-art. Part of this work has been published in [GLR10] and part of this work has been accepted for publication [GLR11], as detailed below:
  - **Multipath TCP with Network Coding for Wireless Mesh Networks**, ***Steluta Gheorghiu***, Alberto Lopez Toledo, Pablo Rodriguez. IEEE International Conference on Communications (ICC) 2010, WNS
  - **A Network Coding Scheme for Seamless Interaction with TCP**, ***Steluta Gheorghiu***, Alberto Lopez Toledo, Pablo Rodriguez. IEEE International Symposium on Network Coding (NetCod) 2011 (poster)
- **an algorithm that uses topological information to optimally achieve the rate region** for the case of point-to-multipoint transport over the combination network with erasure links in the special case of degraded multicasting, in *Chapter 4*. Through an analytical comparison with a simple approach oblivious to the network structure, we show that

without complete knowledge about the underlying topology, it is not possible to achieve the highest rate pairs from the rate region. This work has been accepted for publication [GSFL11]:

  - **Degraded Multicasting with Network Coding over the Combination Network**, ***Steluta Gheorghiu***, Shirin Saeedi Bidokhti, Christina Fragouli, Alberto Lopez Toledo. IEEE International Symposium on Network Coding (NetCod) 2011; also as Technical Report EPFL-REPORT-152016, EPFL, September 2010

- **a system architecture for network coding-based video streaming to users with different subscription levels, in a wireless multicast setting**, in *Chapter 5*. By carefully matching layered video with a specific network coding technique, our system achieves gains in terms of buffering delay, and variability of the quality played at the sinks, even if the feedback received by the sender is limited and unreliable. This work has been published in [GLL+10] and [LGB+10], see below:

  - **On the Performance of Network Coding in Multi-Resolution Wireless Video Streaming**, ***Steluta Gheorghiu***, Luisa Lima, Alberto Lopez Toledo, Joao Barros, Muriel Medard. IEEE International Symposium on Network Coding (NetCod) 2010
  - **Secure Network Coding for Multi-Resolution Wireless Video Streaming**, Luisa Lima, ***Steluta Gheorghiu***, Joao Barros, Muriel Medard, Alberto Lopez Toledo. IEEE Journal on Selected Areas in Communications, Wireless Video Transmission, 2010, vol. 28

- **a system architecture to efficiently provide an incentive scheme to foster cooperation among users for live video streaming in a peer-to-peer setting**, in *Chapter 6*. Our system accounts for peers with heterogenous requirements through the use of layered video, and solves the complex issues of scheduling in order to prioritize the base layer through the use of network coding. We further show that our system achieves a short decoding delay, thus enabling a smooth playback at the users, while also ensuring that a user's contribution to the system is proportional to the decoded rate. This work has been accepted for publication [GLLB11]:

  - **A Layered Network Coding Solution for Incentives in Peer-to-Peer Live Streaming**, ***Steluta Gheorghiu***, Luisa Lima, Al-

berto Lopez Toledo, Joao Barros. IEEE International Symposium on Network Coding (NetCod) 2011

For a complete list of publications of the author, we refer the reader to *Appendix* A.

CHAPTER 2

Background

After its introduction in [ALY00], the innovative paradigm of network coding received an increasing attention from the research community and several theoretical results soon followed. In [LY03], the authors prove that linear network coding suffices to achieve the optimal throughput in a multicast setting. [LL04] addresses the case of undirected networks and shows that optimal throughput achievement, which is NP-hard with routing, is possible in fact by means of network coding.

Let us first give an example introduced in [ALY00] to illustrate the concept of network coding. Consider a simple setting, with a source node S, and two receiver nodes R_1 and R_2, as shown in *Figure 2.1*. The source S needs to deliver two bits, x_1 and x_2 respectively, to both receivers, using each network edge only once. Therefore, it sends bit x_1 to relay A, and bit x_2 to relay B, and each of the relays further broadcasts the received bit. Consequently, destination R_1 receives x_1, destination R_2 receives x_2, while relay C obtains both bits. With *traditional* approaches, node C needs to *decide which of the two bits to forward*. If for example it picks bit x_1, and further relay D broadcasts x_1 to both of the receivers, then R_1 obtains only bit x_1 (twice), and R_2 obtains both of the source bits. With *network coding*, the nodes are allowed to *perform coding operations* on the incoming data, thus node C forwards $x_1 + x_2$ (*xor* operation), which is then broadcasted by D to both receivers. Consequently, receiver R_1 obtains x_1 from A and $x_1 + x_2$ from D, which enables it to decode bit x_2 as well. Similarly, node R_2 decodes both bits as well. Thus, by means of network coding, the source S can deliver two bits simultaneously to both

destinations, using each network edge only once.

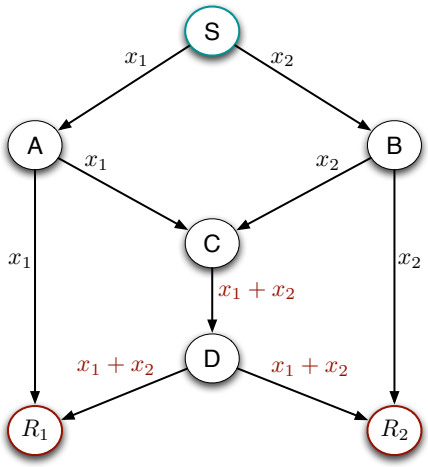


Figure 2.1: Network coding – the butterfly network.

If in the above example, a simple *xor* operation was sufficient in order to achieve the desired goal (i.e. delivery of two source bits, to two receiver nodes), in more complex scenarios, one may need more complex coding operations, such as linear network coding, where outgoing packets are linear combinations of incoming packets. For generating linear combinations, one can use fixed coefficients (deterministic algorithms), or random coefficients, drawn uniformly from a finite field (Random Linear Network Coding). For more information on deterministic algorithms, we refer the reader to [HKM05, FS06a]. In the following, we focus on the second approach, as this is the method we use for network coding throughout this thesis.

Random Linear Network Coding (RLNC) can be implemented in a distributed fashion, where nodes draw several coefficients at random from a finite field and use them to form linear combinations of incoming packets [HKM+03]. In particular, with RLNC the nodes can operate completely desynchronized and can forward a random linear combination independently of the information present at other nodes in the network. Additionally, when collecting a random combination of packets, there is a high probability of getting a linearly independent packet. With RLNC, the linear coding capacity of a network depends on the size of the chosen field [DFZ05].

The original packets, that have not undergone any coding, are also called *native packets* and for the rest of this thesis we will use the terms of *linear*

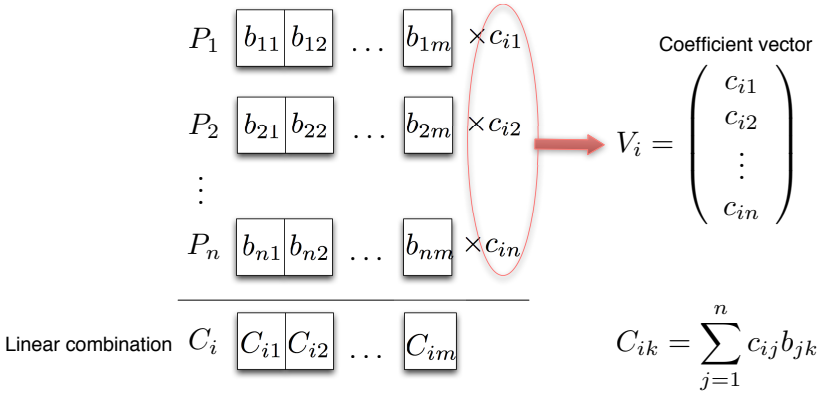


Figure 2.2: Random Linear Network Coding – generating a linear combination C_i of n packets.

combination and *coded packet* interchangeably. Consider one has to generate a linear combination of n packets. Then, the payload of each packet $j, 1 \leq j \leq n$, is split into m blocks b_{jk}, with $1 \leq k \leq m$. The number of blocks m depends on the size of the field over which the coding operations are performed and the size of the payload. Next, one chooses n random coefficients, c_{ij}, from a finite field, multiplies each block of packet j with the corresponding coefficient c_{ij}, and adds them together to form the coded packet C_i, as illustrated in *Figure 2.2*. Thus, the k-th block of the linear combination C_i is given by:

$$C_{ik} = \sum_{j=1}^{n} c_{ij} b_{jk} \tag{2.1}$$

The randomly selected coefficients c_{ij} form the coefficient vector V_i, which is included in the header of the coded packet C_i and travels together with it towards the destination. In order to obtain the native packets, a node needs to collect n independent linear combinations. The coefficient vectors V_i, $1 \leq i \leq n$ corresponding to the collected coded packets form the *coefficient matrix* – denoted by A in *Figure 2.3*. The destination next solves a linear system, by performing Gaussian elimination over the coefficient matrix, and decodes the native packets.

In the next sections, we discuss the bodies of work relevant to this thesis contributions, as follows. In *Section 2.1* we review the existing work for multipath techniques for wireless networks, with an emphasis on network coding and TCP sessions. We introduce the problem of multicasting to heterogenous

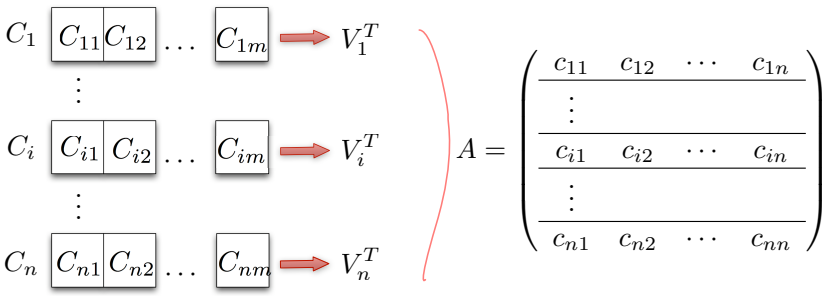


Figure 2.3: Random Linear Network Coding – coefficient matrix A corresponding to n coded packets.

users in *Section 2.2* and introduce the main results in the area. In *Section 2.3* we present an overview of the related work for media streaming in peer-to-peer networks.

2.1 Wireless networks

A typical approach to improve throughput in wireless environments is to exploit the multiple paths inherently available in such contexts and several research efforts have addressed this issue.

In [PRSR06] the authors rely on path diversity to reduce congestion effects. Their solution uses a routing protocol, *Biased Geographical Routing (BGR)* to forward packets along curves, instead of along the shortest path, towards the destination. Thus, the congestion level is decreased, but at the cost of higher delay.

ExOR [BM05] is an integrated MAC and routing protocol, that relies on path diversity in order to choose the best forwarder for each packet. With *ExOR* the packets destined to the same destination are grouped in batches, which are then broadcasted and stored only by nodes from the forwarder list included in the header. At the end of the batch, the nodes will start to transmit the received packets following the priorities from the forwarder list, if another neighbor with a higher priority has not already transmitted them. This scheduling avoids collisions and duplicates, and is shown to increase throughput, but the overhead grows proportionally with the number of nodes.

However, if simply forwarding traffic across multiple paths is sufficient to achieve higher throughput in UDP sessions, for TCP connections extra care needs to be taken to reduce the side effects, such as packet reordering and

delay, introduced through the use of multiple paths.

The authors of [LPCR02] propose small modifications to TCP to cope with out-of-order packet arrivals. On the sender side, they make use of a simple heuristic to determine the fast retransmission threshold T for the given number of available paths, N, as $T = D(1+\log_2 N)$, where D is the number of duplicate ACKs. This way, the time that receivers wait for packets to arrive on different paths is increased. On the receiver side, an ACK is not sent immediately after an out-of-order packet is received, but after two such packets. In case that a retransmitted packet arrives at the destination, then an ACK is sent right away. With these changes, the TCP performance with two paths is almost doubled as compared to what single path achieves.

Another work that explicitly addresses the problem of balancing TCP over multiple paths in wireless mesh networks is [RGGK08]. The authors introduce *Horizon*, a novel system for multipath forwarding, based on the theoretical results on backpressure, which uses a light-weight mechanism to accurately estimate path quality and calculate backpressure between nodes. It also employs an algorithm that delivers packets in-order and with a smooth rate at the destination, thus guaranteeing that TCP can take advantage of multipath routing without collapsing its congestion window. However, *Horizon* does not provide any reliability and any lost packet is recovered through TCP retransmissions.

In wireless environments, network coding provides benefits in several directions, such as robustness against losses [DEH+05] and energy gains [FS06b, SE06]. In [LMK05], the authors address the problem of minimum-energy multicast, which is NP-complete with routing, but can be solved in polynomial time with network coding. They divide the problem into two parts and solve each of them separately: i) find a subgraph of minimum cost – where cost refers to energy consumption, average latency or monetary cost, and ii) devise a coding scheme and apply it to the subgraph.

Typically, network coding approaches for wireless networks can be broadly classified as: i) *intra-flow encoding*, where the coding operations are performed over the packets that belong to the same flow of data, such as *MC2* [GHK+07], and *MORE* [CJKK07], and ii) *inter-flow encoding*, where the coding is realized over packets from different flows of data, such as *COPE* [KRH+06], and *Analog Network Coding* [KGK07].

MC2 [GHK+07] exploits the path diversity in wireless mesh networks and solves the node coordination problem by means of network coding. It uses an algorithm to control the data rate across each available path, which in addition guarantees fair allocation across several concurrent flows, and uses backpressure

to control congestion. The authors show that the proposed scheme achieves a throughput 30% higher than with single path routing, and 20% higher than other multipath routing schemes.

A similar approach is introduced in [CJKK07], *MAC-independent Opportunistic Routing & Encoding (MORE)*, which randomly mixes packets before forwarding them in the network across multiple paths. *MORE* is shown to improve throughput by 22% over *ExOR* for unicast sessions, and by up to 45% in certain topologies with spatial reuse opportunities. However, *MORE* does not perform congestion control. In addition, neither *MC2*, nor *MORE*, discusses the interaction with TCP.

With *COPE* [KRH+06], nodes store all the packets they overhear for a limited period of time and broadcast reception reports to inform their neighbors about which packets they have stored. When a node has to transmit a packet, it generates a linear combinations of the stored packets, such that to maximize the number of packets transmitted while ensuring that all the intended next hops are able to recover the packets they need. *COPE* improves the throughput by a factor of $3-4$x, when coding opportunities are high, that is when multiple flows run concurrently in the network.

Analog network coding (ANC) [KGK07] performs network coding over signals at the physical layer, by encouraging simultaneous transmissions at carefully selected source nodes. Next, the destinations use information from the network level to cancel the interference and decode the signal they are interested in. The authors show through an evaluation over a small testbed that *ANC* provides an increase in throughput of 70% over traditional wireless routing and 30% over digital network coding.

A more recent approach which enables the use of network coding for TCP sessions over single paths in wireless networks is presented in [SSM+09b], where the coding operations are performed only on the packets from the TCP's transmission window. In order to prevent TCP from collapsing its congestion window due to the delay introduced by coding, the approach uses a feedback mechanism to inform the sender about the rate at which coded packets arrive at the receiver, even if no native packet can be decoded immediately. Such a mechanism requires changes to the TCP layer, thus it does not lend itself to a practical implementation.

2.2 Multicasting to heterogenous users

Multicasting refers to the delivery of the source content to a group of receivers, while degraded multicasting implies that different users require different subsets of the source content. If the first case is by now well understood, for the second case several heuristic algorithms have been proposed, however, there is in general no exact characterization of the optimal achievable rate region [FS06b].

The problem of delivering a set of degraded messages over a general broadcast channel was first introduced in [Cov72], and several special cases have been discussed in [Ber74, Gal74]. [KM77] addresses the problem of delivering three messages to two receivers, where one message is public, and the other two are private messages – one for each receiver. In [NE09], the authors show the capacity region for a class of three-receiver broadcast channels with two-degraded message sets. [SDFP09] examines two-message broadcasting over a linear deterministic channel.

Degraded broadcasting is motivated by various scenarios, such as video streaming applications, where users are heterogeneous and have different subscription levels, thus requiring a different resolution of the content [GLL+10]. To ensure graceful degradation in the presence of packet losses and differentiated service provision to heterogenous users with distinct requirements, typical video codecs adopt a multi-resolution source coding approach to generate a scalable video stream with multiple layers. In the following, we introduce two multi-resolution techniques, namely *Multiple Description Coding* and *Layered Coding*, and discuss related state-of-the-art.

With *Multiple Description Coding (MDC)* the video signal is divided into several descriptions. If a client receives only one description, it can reconstruct the signal with some level of distortion [SOPJ00], while if more descriptions are received, then the quality of the decoded video is higher. In the case of *Layered Coding (LC)*, the video consists of a base layer and multiple enhancement layers, with the base layer providing a basic level of quality. The enhancement layers can be decoded only if the base layer has been decoded, which makes *LC* sensitive to the losses of the base layer.

The key difference between the two approaches is that while with *LC* a node needs to receive all previous layers in order to decode the current layer, with *MDC* the quality of the stream is directly proportional to the number of descriptions received [TF00]. However, *MDC* coding is more computationally complex than *LC*.

Several bodies of work have addressed the benefits of using each of these

techniques. In [SOPJ00], the authors show that *LC* is better than *MDC* for error free networks, if no feedback is available; however the performance of *LC* deteriorates rapidly in the presence of losses. In addition, *LC* outperforms *MDC* when using multiple paths and with a good allocation of packets sent over each path [NCO04].

On the other hand, *MDC* is more efficient for scenarios with strict delay constraints and no feedback [WPLM02]. For the case of delivery over one path or two paths, the authors in [CHG03] propose a radio distortion optimized scheduling (RaDiO framework), that takes into account the interdependencies between data packets when scheduling transmissions on the available paths. Using this framework, they show through an extensive experimental evaluation that *LC* is generally better than *MDC* even over error-prone networks.

The work in [CWP03] combines *MDC* and *LC* for a multicast scenario, by considering two layers for the system. The base layer is transmitted to each low-bandwidth client, while both layers are transmitted to each high-bandwidth client.

2.3 Peer-to-peer networks

Peer-to-peer (P2P) networks have been shown as a promising architecture to improve content distribution to heterogenous users, due to their intrinsic characteristics, such as decentralized operation and robustness. While the contributions in the area of P2P are numerous, in this section we do not try to present an exhaustive review of the literature, but we focus only on the research closer to our work, namely streaming in P2P networks.

In [ATW02] the authors address the problem of streaming media in a CDN. The content is located at multiple servers, and the proposed scheme relies on path diversity to ensure resilience to errors. *CoopNet* [PWCS02] combines infrastructure-based CDN and P2P content distribution to stream media, encoded with *MDC*. With *CoopNet* each substream is delivered to each client via a different peer, using application level multicast. In a later work [PWC03], the authors show that *MDC* works well with multiple distribution trees.

The work in [GHK+07] showed that network coding provides benefits in P2P networks, as well. In addition, the inherent properties of RLNC make it particularly suitable as a framework for dynamic and unstable networks, such as delay-tolerant networks [WL05] and content distribution networks (CDNs) [GR05, DGWR07].

[WL07a] introduces *Lava*, a pull-based P2P live streaming protocol, and

shows that network coding can support a wide range of streaming rates (100 KB per second - 8 MB per second). Through evaluation over a testbed, they show that network coding is more resilient to peer departures, but shows a higher percentage of playback skips, for the flash crowd scenario. However, with network coding the clients receive less redundant packets. [WL07b] further develops this work to propose a random push P2P streaming protocol which uses RLNC in order to take advantage of these and simultaneously improve playback. The strategy is to make peers proactive in sending innovative segments to downstream peers without pulling the data, thus enforcing a push scheme on a mesh topology.

In [CWCC08], the authors use network coding to reduce network traffic and server load by leveraging on end host's buffer space. The strategy is to use a tree-based system, in which the coding vectors of the parents are determined to guarantee that the clients receive linearly independent information. The work in [LPDG06] includes a scheme for peer-to-peer live media streaming with network coding, in which data is propagated using pull-based gossip. Nodes exchange their coding vectors in order to determine exactly how much innovative data they can send to a neighboring peer. Results show that network coding helps decrease the initial buffering delay and improve playback quality.

Reference [FL08] presents an analytical framework for P2P systems with network coding in a push setting. Results show that the use of network coding yields a high playback quality, short initial buffering delays, resilience to peer dynamics and lower server load. The key idea is that multiple seeds can serve the same peer simultaneously. *CodedStream* [GZL03] is a media distribution system for high-bandwidth applications which uses network coding at the bottlenecks in the overlay multicast in order to prevent stress on the links. An overlay multicast graph is built to optimize the use of network coding, and the video stream is encoded with *MDC*. [NNC07] uses a nested coding structure to combine scalable video stream with network coding in order to stream a video to a single user using multiple servers. A different approach is used in [AGG+07], where network coding is used judiciously to solve scheduling problems within segments of a video-on-demand protocol. Finally, a more general approach is proposed in [TF07], where raptor codes for video streaming are re-encoded at intermediate nodes to take advantage of path diversity in the overlay network.

2.3.1 Incentives

A typical problem in P2P networks is that of lack of incentives for collaboration among users for the common good. P2P systems work under the premise

that users cooperate with each other in sharing resources. However, these systems are typically large and too dynamic to impose generalized incentive mechanisms, with users leaving the network and ending share of information. Additionally, there can exist an asymmetry of interest, in that not all users can perform a direct trade.

Systems to enforce cooperation have several challenges to deal with, such as collusion between users, zero-cost identities (that is, systems that allow users to "reboot" their identity) and traitors [FLSC04]. There are several traditional measures to mitigate these attacks. The most common, which consist of virtual currency, payment or credit-based approaches, require a robust reputation system and a trusted entity for exchange of reputation information. Other systems include scoring, auditing and client puzzles [FLSC04, LSP+07b]. BitTorrent uses a rate-based tit-for-tat mechanism for motivating user cooperation, in which each user only uploads file segments to its 4 top seeders [PIKA08]. Even if this scheme works well in keeping the network "alive", it is unfair as it permits free-riding and strategic manipulation. In fact, an average user can obtain the same download performance as another that is contributing 100 times more [PIKA08].

Although the primary performance measure for the contribution of each peer in traditional file downloading settings is the download time, a valid approach for P2P video streaming is to use higher video quality as the performance measure for incentives [LSP+07b]. This motivates the use of *LC* video as an incentive to sharing. A similar approach is considered in [LSP+07a], however with *MDC*. The quality of video received by a user is proportional to the number of descriptions it receives, which is in turn proportional to the number of descriptions the user contributes to the network.

Other directions include trading of seeds and exchange of control data for incentives [SFC08]. *Coopnet* [PWC03] uses peer cooperation to help the server cope with peer transience scenarios (such as flash crowd). Users are willing to collaborate since they only distribute content in which they are interested as well; moreover, they contribute only as much upstream bandwidth as they consume. The scheme is push-based, however it uses redundancy in network paths through the use of multiple distribution trees and redundancy in the shared data through the use of *MDC*. A heuristic-based strategy for combining *LC* and network coding is presented in [ZYZ+06]. Although throughput and network resource consumption are improved, delay slightly increases.

Part I

Point-to-Point Communications

CHAPTER 3

Improving the Performance of Wireless Transmissions

Wireless mesh networks are emerging as the next-generation systems to connect communities [AW05]. As these networks proliferate and become ubiquitous, it is expected that the users will start to demand more versatile applications such as video streaming and VoIP. Unfortunately, TCP does not work well in wireless multihop environments and the problem of performance degradation of traditional TCP protocols when used in wireless networks is well known [BPSK97]. When losses occur in a wireless environment, TCP considers them as being caused by congestion (i.e. buffer overflow), and triggers its congestion avoidance mechanism which reduces the sending rate to adapt to the new network conditions. Still, the actual capacity of the wireless network has not changed, and hence the network becomes underutilized.

A common solution to address this problem is to exploit the broadcast advantage of the wireless medium, where transmissions are overheard at no cost by multiple nodes situated in the range of the transmitter. These nodes can be further used in parallel to relay packets, and consequently the packets can travel along multiple paths from a source to a destination[1]. These multipath schemes have been shown to increase robustness and capacity in wireless mesh networks [LPCR02], [LL06].

[1] Multipath schemes have been applied successfully to wired networks as well [Cet07, KMT06, HSH+06]

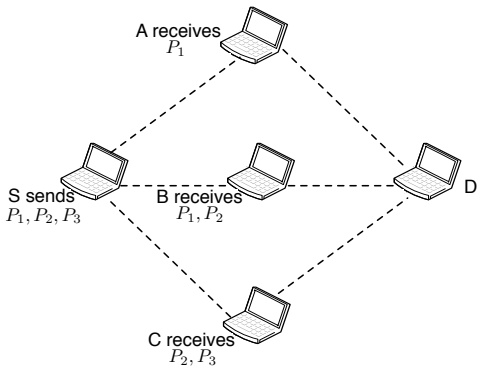


Figure 3.1: Motivating example. Source S can reach destination D through three neighbors, A, B and C.

However, multipath techniques pose further challenges to TCP because the available paths may have arbitrary delays and packets may arrive at destination out-of-order [RGGK08]. TCP receivers react to out-of-order arrivals by sending duplicate ACKs, forcing the TCP sender to unnecessarily retransmit the packet that the receiver is expecting. In addition, an efficient use of the multiple paths poses further challenges. Consider the simple scenario from *Figure 3.1*, where the source S can communicate with destination D using 3 neighbors, A, B and C. Assume that S sends three packets, P_1, P_2, P_3, and the three relays receive the packets as illustrated in the figure. In this situation, the nodes need to *coordinate* and *schedule the packets' transmissions* so that the destination receives all of them without duplicates.

Network coding [ALY00] can effectively be employed to solve the relay coordination and the packet scheduling problems. In the above example, each of the relays can use network coding to simply forward a linear combination of the received packets. Once the destination D receives the three linear combinations, it solves a linear system and recovers the original packets. Typical network coding techniques divide traffic in *batches* [CJKK07] or *generations* [GHK+07] and a coded packet is a linear combination of the packets that belong to the same group. These approaches introduce a *coding delay* at the source, i.e. the waiting time to receive the batch of packets to encode, and a *decoding delay* at the destination, i.e. the waiting time to receive enough linear combinations to solve the linear system and recover the original packets. These delays cause the TCP sender to timeout, lower its transmission window and unnecessarily retransmit packets.

In order to avoid the *coding delay*, [SSM+09b] introduces an *online coding* technique, where the sender generates linear combinations as new native packets arrive. However, due to the unreliable links of the wireless medium, not every coded packet received by the destination can immediately reveal a native packet, hence the *decoding delay* cannot be completely eliminated. Therefore, in order to prevent TCP timeouts, [SSM+09b] uses a feedback mechanism, where the destination acknowledges the rate of receiving innovative packets, instead of the decoding rate. In other words, this feedback mechanism practically informs the sender that coded packets arrive at the destination, although no native packet can be decoded right away.

With these considerations in mind, we design *CoMP*, a network coding multipath protocol for wireless mesh networks that extends current state-of-the-art and combines network coding with multipath routing to increase throughput for TCP sessions in wireless mesh networks. The main features of our protocol are as follows:

- *CoMP* is a multipath online network coding scheme that exploits the path diversity in a mesh network, and eliminates the delay problems introduced by the coding operations (*Section 3.3.1*);
- *CoMP* performs error control, by using an algorithm to estimate the loss probability in the network and to adjust online the rate of sending linear combinations (*Section 3.3.2*);
- *CoMP* uses a hybrid credit-based algorithm to balance the load on the available paths, where nodes compute distributively the rate of generating linear combinations (*Section 3.3.3*);
- *CoMP* performs congestion control using a mechanism based on back-pressure (*Section 3.3.4*).

We extensively evaluate *CoMP* using the *ns-2.33* network simulator [ns2], and compare it to state-of-the-art protocols: *Horizon* [RGGK08], *Square coding* such as [CJKK07, GHK+07], and *Online coding* [SSM+09b]. The results show that *CoMP* not only achieves a higher throughput, but it is also more efficient, due to exploiting path diversity and re-encoding packets on a hop-by-hop basis.

The chapter is organized as follows. In *Section 3.1* we show the benefits brought by the use of multiple paths. In *Section 3.2* we briefly describe two network coding schemes, relevant to our work. We present our protocol and its main functional blocks in *Section 3.3*. *Section 3.4* provides the results of

our simulations and a comparison to existing state-of-the-art. In *Section 3.5* we revisit the feedback mechanism for TCP and present an enhancement that ensures a smooth interaction with the TCP protocol, even if the source receives feedback only on the decoding rate. *Section 3.6* concludes with some final remarks and directions for future work.

Part of this work has been published in [GLR10] and part of this work has been accepted for publication [GLR11].

3.1 Multipath benefits in wireless mesh networks

In this section we evaluate the potential of using multiple paths to increase the throughput in wireless mesh networks. To this end, we define several multipath protocols that provide us with some insights into how different network components interact, insights which will be taken into account for the design of new protocols based on network coding, that also exploit path diversity.

The *non-network coding approaches* are the following:

- *plain multipath*: for each packet, the source chooses the next hop in a round-robin manner. When receiving a packet, a relay will just forward it.
- *plain overhearing*: the same as *plain multipath*, plus all nodes receive packets in promiscuous mode. Destination receives duplicate packets, because a piece of information may be overheard and forwarded by multiple relays.
- *credit assignment and reassignment*: we use a credit-based scheme in order to avoid reception of duplicate packets at destination. For each packet that it forwards, the source assigns a credit to one of the next hops and only the node receiving a credit is allowed to do the forward. Upon reception of a packet, each relay sends an acknowledgment. When losses occur and the node that was assigned a credit did not receive the packet, but one of its neighbors did, the source reassigns the credit to that neighbor.
- *hop-by-hop reliability*: the same as *credit assignment and reassignment*, plus the source retransmits packets that are lost. Usually, a node will retransmit a packet 5 times before dropping it.

We implement and test these protocols using the network simulator *ns-2.32* [ns2]. For the simulations we use a toy diamond-shaped topology shown

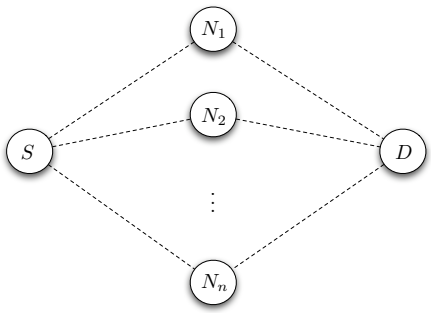
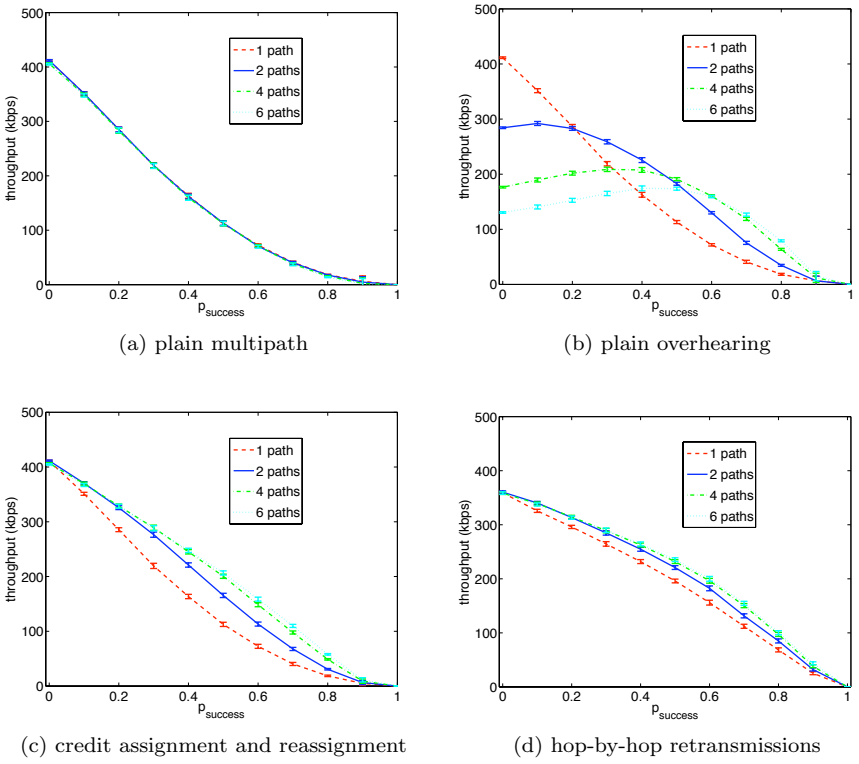


Figure 3.2: Diamond topology used for *ns-2* simulations

in *Figure 3.2*. In order to fully understand the impact of nodes working in promiscuous mode, we disable MAC layer retransmissions for all our experiments. We use UDP, with a CBR source sending at 450 kbps, and TCP, with an FTP application on top, as transport protocols. Moreover, we use *two-ray ground* as the propagation model and we vary the probability of having a successful transmission per link, $p_{success}$ and the number of relays doing the forward.

In *Figure 3.3* and *Figure 3.4* we plot the average throughput in kbps on the y-axis with the probability of having a successful transmission on each link, $p_{success}$, on the x-axis. When using *plain multipath* with UDP, the throughput is the same independent of the number of paths that are used (see *Figure 3.3a*). This is due to the fact that the traffic is basically split equally among the paths and losses or packets arriving out-of-order at destination do not influence the throughput at the sender. Unfortunately, this is not the case for TCP. As we can see from *Figure 3.4a*, there are small differences in throughput when multiple paths are used, but overall the performance is significantly affected by the losses in the network.

For *overhearing* we observe similar results for both UDP (*Figure 3.3b*) and TCP (*Figure 3.4b*), that is, as $p_{success}$ decreases, using more relays improves the total throughput. When the losses are low, multiple paths achieve a lower throughput than a single path, because intermediate relays forward all the packets that they receive, thus bandwidth is wasted with duplicates. Moreover, the maximum throughput that TCP gets is lower than that obtained with UDP, because packets arrive at destination out-of-order and for TCP, this will result in duplicate acks being sent to the source of the flow, further triggering unnecessary retransmissions.



(a) plain multipath

(b) plain overhearing

(c) credit assignment and reassignment

(d) hop-by-hop retransmissions

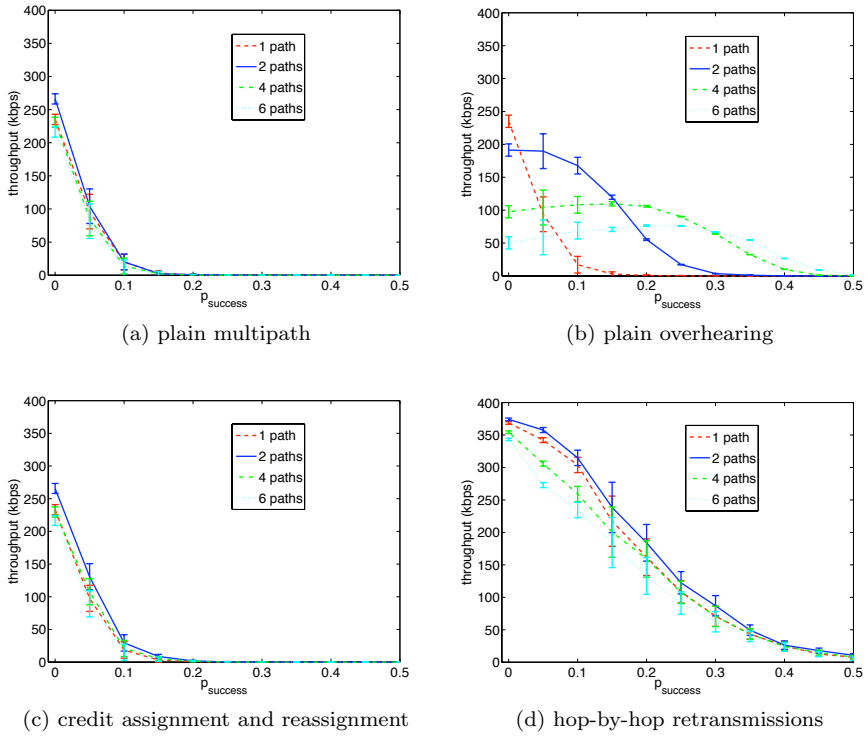
Figure 3.3: UDP without network coding - diamond topology

In the case of *credit assignment and reassignment*, notice from *Figure 3.3c* that for UDP multiple paths yield a clear improvement over single path. This is due to the fact that no more duplicates arrive at the receiver and also some of the losses are recovered when credits are reassigned. As an example, two paths is up to 67% better than one path. For TCP though, credit reassignment has a negative effect (*Figure 3.4c*), and even if more packets arrive at destination, they are out-of-order and do not help at all.

With *hop-by-hop retransmissions* we obtain the best results for both UDP and TCP because all the losses are recovered with high probability, p_r:

$$p_r = 1 - (1 - p_{success})^{n_{retx}},$$

where n_{retx} is the number of retransmissions and is equal to 5. For UDP



(a) plain multipath

(b) plain overhearing

(c) credit assignment and reassignment

(d) hop-by-hop retransmissions

Figure 3.4: TCP without network coding - diamond topology

(*Figure 3.3d*), the higher the number of paths, the better, although the benefit is not proportional to the number of paths. For example, the throughput obtained with two paths improves over that with one path by up to 44%. For TCP (*Figure 3.4d*), using two paths achieves an improvement of up to 40% over single path. If more paths are used, then the interference between nodes increases and the throughput decreases.

These preliminary results show that multiple paths used in parallel to forward data from a sender to a receiver have the potential to improve the throughput of the network in a wireless setting. There are several factors that influence the overall performance and that should be taken into account when designing efficient protocols. Among these factors, we find the topology of the network which is very close related to spatial reutilization. If the node density is high, the opportunities to do parallel transmission are reduced. Even if multiple paths are available from source to destination, they may not be spatially dis-

joint and using more paths may cause more interference, consequently reducing throughput.

Moreover, transport algorithms should be aware of the protocols from the higher layers and provide mechanisms to ensure a reliable service. For example, with UDP no special care needs to be taken, but for TCP, the interactions among itself, the MAC protocol and the Link Layer protocol are more subtle and not completely understood.

3.2 Preliminaries

We consider that network coding is implemented as a slim layer situated below the transport layer in the protocol stack, as illustrated in *Figure 3.5*. The packets sent from the TCP layer at the source are intercepted in the network coding block and stored in a *coding buffer*. Next, the sender uses *Random Linear Network Coding (RLNC)* to mix the packets from the *coding buffer* according to a network coding scheme (which we describe in the next sections) and sends the resulting coded packets into the network. On the receiver side, the coded packets are stored in a *decoding buffer*. When the destination receives enough linear combinations, it uses Gaussian elimination to solve the linear system and obtain the native packets, which are then passed up to the transport layer. We refer to the number of packets mixed in a linear combination as the *coding window*.

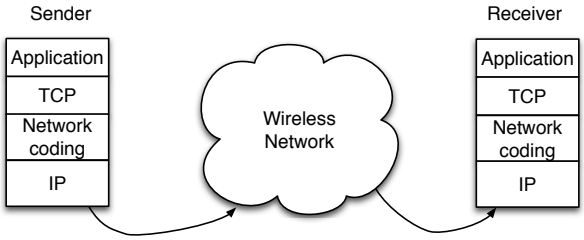


Figure 3.5: Simplified protocol stack, with network coding implemented below the transport layer.

In the following, we explain in detail two existing network coding schemes that are relevant to our work, namely *square coefficient matrix*, and *online coding* respectively, using a simple setting. We consider a scenario where a sender has to send three packets to a receiver, situated one hop away. Assume that the source produces a packet in a time unit and a packet needs a time unit as well to propagate from the sender to the receiver. For the purpose

of this example, we consider that the coding and the decoding operations are performed instantaneously. Next assume that the sender receives packets P_1, P_2 and P_3 in time slots t_1, t_2 and t_3, respectively.

3.2.1 Square coefficient matrix

Typical network coding schemes [GHK+07], [CJKK07] use a *per generation* approach proposed in [CWJ03], where traffic from a flow is divided into generations and linear combinations encode only packets from the same generation. These schemes use full square coefficient matrices: the source waits until all the source packets from a generation are received to start coding packets. Each coded packet is the result of mixing all the packets from a generation, thus the *coding window* is equal to the size of the generation. For the rest of the chapter, we use g_{size} to denote the size of the generation.

Table 3.1: Network coding using a full *square coefficient matrix* with size of the generation of 3 packets.

Time slot	**Source**			**Destination**		
	Recv	Coding Bfr.	Sent	Recv	Decoded	Sent
t_1	P_1	P_1				
t_2	P_2	$P_1,\ P_2$				
t_3	P_3	$P_1,\ P_2,\ P_3$	$C_1 = c_{11}P_1 + c_{12}P_2 + c_{13}P_3$			
t_4		$P_1,\ P_2,\ P_3$	$C_2 = c_{21}P_1 + c_{22}P_2 + c_{23}P_3$	C_1		
t_5		$P_1,\ P_2,\ P_3$	$C_3 = c_{31}P_1 + c_{32}P_2 + c_{33}P_3$	C_2		
t_6		$P_1,\ P_2,\ P_3$		C_3	$P_1,\ P_2,\ P_3$	$Ack_{1,2,3}$
t_7	$Ack_{1,2,3}$					

With a *square coefficient matrix*, in the setting from above the source starts to send coded packets in time slot t_3, only after it receives all the packets from the generation, as shown in *Table 3.1*. The destination decodes the packets in time slot t_6, only after it receives all three linear combinations. Therefore, this coding approach introduces two delays: a) the *coding delay* at the source and b) the *decoding delay* at the destination, which have a negative effect on the performance of TCP. The sender removes the packets from the buffer in time slot t_7, after it receives the ACK for all the packets from the generation, then it proceeds with the delivery of the following generation of packets.

3.2.2 Online coding

In *online coding* introduced in [SSM+09b], the source mixes the packets from the TCP transmission window. For *online coding* there is no notion of "generation" or "batch". Whenever the sender receives a packet, she stores it in the *coding buffer* and generates a linear combination, by mixing all the packets from the buffer. Note that this way the *coding delay* is eliminated, however the *decoding delay* may not be avoided. Due to routing inefficiencies or high volatility in the wireless medium, the destination may not be able to decode packets on-the-fly, as it does not receive the necessary degree of freedom. This *decoding delay* causes the TCP layer to see traffic on a bursty pattern and affects the round-trip time estimation at the TCP sender. If the decoding delay is too high, then the TCP sender may timeout, lower its transmission window and unnecessarily retransmit packets.

To prevent these effects, [SSM+09b] introduces the following feedback mechanism: every reception of a linear combination is acknowledged to the TCP sender, even if no native packet can be decoded right away. The goal of this mechanism is to inform the TCP sender that packets arrive at destination, thus preventing TCP timeouts and useless retransmissions. Effectively, what it is signalled to the TCP sender is the *rate at which information is received at the destination*, which exactly corresponds with the information that a TCP sender would expect to obtain in a fixed network and without network coding.

Native packets are removed from the *coding buffer* whenever the source receives their corresponding ACKs. Consequently, the coefficient matrix evolves in a block-diagonal shape. For this case, the *coding window* increases by one when a native packet arrives to the *coding buffer* and decreases by one when an ACK reaches the sender.

Table 3.2: *Online network coding.* The source sends a linear combination every time a native packet arrives to its *coding buffer*, allowing the destination to decode on-the-fly.

Time slot	**Source**			**Destination**		
	Recv	Coding Bfr.	Sent	Recv	Decoded	Sent
t_1	P_1	P_1	$C_1 = c_{11}P_1$			
t_2	P_2	$P_1,\ P_2$	$C_2 = c_{21}P_1 + c_{22}P_2$	C_1	P_1	Ack_1
t_3	$Ack_1,\ P_3$	$P_2,\ P_3$	$C_3 = c_{32}P_2 + c_{33}P_3$	C_2	P_2	Ack_2
t_4	Ack_2	P_3		C_3	P_3	Ack_3
t_5	Ack_3					

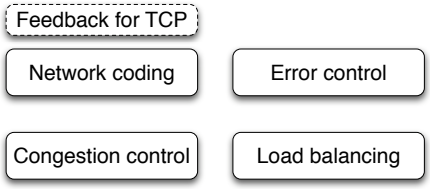


Figure 3.6: The functional blocks of our protocol.

Notice from *Table 3.2* that with this approach the source sends coded packets on the fly, thus eliminating the *coding delay*. Moreover, in this ideal setting where no losses occur, the destination can decode a native packet with every reception of a new coded packet, thus eliminating the *decoding delay* as well. For the above example, the destination can decode the three packets after 4 time units, reducing the time by 33%, as compared to the *square coding* approach.

3.3 Protocol description

In this section we introduce our protocol *CoMP*, and elaborate on its main functional blocks, illustrated in *Figure 3.6*.

3.3.1 Network coding

For our multipath network coding scheme, we use a similar approach as the *Online coding* from [SSM+09b], but with some essential differences. First, *Online coding* uses only one path to forward the packets from the sender to the receiver, while *CoMP* takes advantage of the path diversity characteristic of the wireless mesh networks and forwards packets along multiple paths. Second, in *Online coding* the relays only forward the packets received from the upstream nodes and the coding operations are performed end-to-end. In our scheme, intermediate nodes re-encode packets received from different upstream neighbors, on a hop-by-hop basis. Finally, *CoMP* uses the feedback mechanism as well, in order to alleviate the effect of the *decoding delay* and prevent TCP timeouts and retransmissions.

In the next sections we discuss in more detail the mechanisms that we use to enable online network coding for multipath schemes.

3.3.2 Error control

The *Online coding* scheme described in *Section 3.2.2* relies on the structure of the coefficient matrix – block diagonal – being preserved at the destination. This may not be possible if the coded packets arrive out of order, for example when the different paths from the multipath topology are unbalanced, or when some of the packets are lost. While multipath unbalance can easily be solved with appropriate routing schemes, the effect of losses deserves special attention.

Traditional network coding using square coefficient matrices recovers from losses sending enough linear combinations of the source packets so that the destination receives enough that are linearly independent. The typical approach is to keep sending redundant linear combinations at the end of a generation until the destination responds with a "generation decoded" message [GHK+07, CJKK07]. With block diagonal coefficient matrices, if the structure is not preserved, then the destination is not able to decode packets on-the-fly and to avoid the *decoding delay*. The question then, is to be able to detect with low granularity the rate at which redundant linear combinations must be sent, so that they can be interleaved with other packets, and enable on-the-fly decoding.

The answer to the question "At which rate do nodes need to forward linear combinations to recover from losses?" in the context of network coding is a challenge, and remains an open research question. If the rate is too low, due to the unreliability of the wireless medium, receivers may find themselves in the situation of not being able to decode. On the other hand, if the rate is too high, nodes may receive redundant linear combinations that consume network resources, but are not innovative. Moreover, since the wireless channel is variable in time, the rate of sending coded packets needs to be updated frequently enough to take into account current channel conditions.

With these considerations in mind, we design an algorithm to estimate the loss between any two nodes and to distributively compute the rate at which each node should forward linear combinations. In order to estimate the loss, each node N_k keeps the following information:

- p_i: a list of the ids of the packets sent to each of its downstream neighbors, N_i;
- r_i: the number of total packets that each of its downstream neighbors N_i reports it had received from N_k;
- s_{kl}: id of the last packet that node N_k received from each of its upstream

neighbors, N_l;

- t_{kl}: the total number of packets that N_k received from each of its upstream neighbors, N_l.

Node N_k updates the p_i list whenever it sends a packet to downstream node N_i. s_{kl} and t_{kl} are updated with each reception of a new packet from the upstream neighbor N_l. Note that the update is done regardless of the packet being innovative.

The loss estimation algorithm has two parts:

- *loss estimation at the intermediate nodes*: due to the broadcast nature of the wireless medium, a node may overhear the transmissions of its downstream neighbors. We take advantage of this feature in the following way. Whenever a node N_k has to send a packet, it piggybacks s_{kl} and t_{kl} in the header for each of its upstream nodes, N_l. Next, its neighbors can pick up the packet and find out the information they need.

- *loss estimation at the destination*: the destination does not forward any linear combination and the previous approach cannot be used. However, the destination needs to send TCP ACKs to the sender and therefore it piggybacks s_{kl} and t_{kl} on these packets.

Once a node N_k receives feedback from one of its next hops, say N_i, it updates the information for the corresponding hop and computes the loss towards that hop, according to *Algorithm 1*. If the node determines that a packet has been lost, then it sends a redundant linear combination and adjusts the sending rate taking into account the total loss that it estimates. The total loss that N_k sees is given by: $loss_k = \prod_{j=1}^{n_h} loss_{kj}$, where n_h represents the number of downstream nodes of relay N_k.

3.3.3 Load balancing

When multiple paths are used in parallel, one important decision is how much of the traffic needs to be sent through each path. Consider the example in *Figure 3.1*. If each node A, B and C would generate a linear combination every time it receives a packet from the source, it would result in an explosion of the rate, with the destination receiving several packets that are not innovative. Also, when multiple paths are used, the likelihood of flows traversing the same set of links increases, so one must use a mechanism to ensure fairness among those flows.

Algorithm 1: **Loss estimation**. Node N_k uses the feedback received from N_i to estimate the loss towards N_i, $loss_{ki}$

```
1  if node N_k receives s_ik and t_ik from downstream neighbor N_i then
2      w_s ← t_ik − r_i;
3      counter ← 0;
4      for j = 0 to size(p_i) do
5          if p_i(j) ≤ s_ik then
6              counter ← counter + 1;
7          end
8      end
9      if counter > w_s then
10         loss_ki = (counter − w_s) / counter;
11     end
12     r_i ← t_ik ;
13     while p_i(0) ≤ s_ik do
14         erase p_i(0)
15     end
16 end
```

To address this issue, [CJKK07] and [GHK+07] associate a credit with each packet sent in the network. The credits are created by source, transfered to downstream nodes and consumed at destination. For *CoMP*, we use a *hybrid credit-based scheme* which has two parts. First, credits are generated at the source and transfered to the forwarders, as the schemes mentioned above do. Second, credits can be generated by the intermediate nodes as well whenever they detect a loss, which differentiates our approach from the existing ones. This second part enables each node to compute locally and distributively the rate that it needs to send to recover the losses and ensure a smooth decoding rate at the receiver.

3.3.4 Congestion control

In order to select a forwarder for a packet, we use the *backpressure* methodology [TE92], which allows a node to send packets to a neighbor as long as the neighbor is less congested. The level of congestion is given by the size of the queue, i.e. the more packets are queueing at a node, the more congested the node is. To this end, each node N_k keeps the following information for each of the flows that it forwards:

- an output queue, $output\ buffer_i, i = 1 : n_f$, where n_f is the number of flows that node N_k forwards;

- the queue sizes of its next hops for each flow, $q_{ij}, i = 1 : n_f, j = 1 : n_{hi}$, where n_{hi} is the number of next hops that N_k has for flow i.

Whenever the node has the opportunity to transmit a packet in the wireless medium, it first selects the flow for which to forward a packet, as the one for which it queues the highest number of packets (see *Algorithm 2*). After the flow is selected, N_k chooses the forwarder to be the node with the shortest output queue, as long as the backpressure condition holds.

Algorithm 2: **Flow scheduling and forwarder selection**. When sending a packet, node N_k selects a flow f and chooses the forwarder for that flow. Next, it updates the state information and it sends the first packet from the *output buffer*$_f$.

```
1 select the flow f such that: f = argmax_i size(output buffer_i), i = 1 : n_f ;
2 select the forwarder N_h for the flow f as: N_h = argmin_j q_fj, j = 1 : n_hf ;
3 remove P_0 from output buffer_f ;
4 add id(P_0) in p_h list ;
5 add q_fk in the header of P_0 ;
6 foreach upstream neighbor N_l do
7 |   add (s_kl, t_kl) in the header of P_0 ;
8 end
9 send P_0
```

Next, N_k removes the first packet from the *output buffer*$_f$ and it adds the packet id in p_h, the list of packets sent to N_h. It adds in the header of the packet the size of its own queue, q_{fk} for backpressure computation and for each of its upstream neighbors N_l, the id of the last received packet, s_{kl}, and the total number of received packets, t_{kl}, for loss estimation. Finally, N_k sends the packet.

3.3.5 *CoMP* in a nutshell

Apart from the *output buffer*$_f$, a node keeps also a *coding buffer*$_f$ for each flow f. The *coding buffer*$_f$ contains packets that the node uses to generate linear combinations. The source stores here native, uncoded packets, while the other nodes store linear combinations. The *output buffer*$_f$ contains coded packets that will be sent in the network. Packets are removed from the *output buffer*$_f$ whenever the node has the possibility to transmit a packet. In this case, a node first runs *Algorithm 2* to select a flow f and a forwarder N_h. After that, the

node removes the first packet from the *output buffer*$_f$, it updates the statistics for the downstream node N_h and it sends the packet.

Algorithm 3: **Online coding at the source.** When receiving a data packet, the source stores it and generates a linear combination. m represents the number of packets from the *coding buffer*$_f$. Coefficients c_i are randomly chosen from a finite field. If the received packet is a TCP ACK, then the source removes the corresponding packet from the *coding buffer*$_f$.

```
1 receive packet P_k of flow f ;
2 if P_k is DATA packet then
3 |   store packet in the coding buffer_f ;
4 |   generate LC_k = sum_{i=1}^{m} c_i P_i ;
5 |   store LC_k in the output buffer_f ;
6 else
7 |   P_k is a TCP ACK for packet P_i ;
8 |   remove P_i from coding buffer_f ;
9 end
```

Sender. When a native packet of flow f arrives from the upper layer, the source stores it in the *coding buffer*$_f$. Note that the source performs *online coding*, which means that it generates a new linear combination every time it receives a new packet. To generate a coded packet, the source mixes all the packets from the *coding buffer*$_f$ and stores the resulting linear combination in the *output buffer*$_f$ (see *Algorithm 3*). When the source receives a TCP ACK for flow f, it removes the corresponding packet from the *coding buffer*$_f$, thus sliding the coding window.

Relays. When receiving a coded packet from flow f, an intermediate node N_k first updates the state for the upstream neighbor N_l that sent it. The updated information is the id of the last received packet, s_{kl}, and the total number of received packets, t_{kl}. Next, if the coded packet is innovative, N_k stores it in the *coding buffer*$_f$, otherwise N_k drops it (see *Algorithm 4*). A relay generates and forwards a linear combination when it receives a credit. Note that a linear combination of coded packets is also a linear combination of the original packets.

Receiver. When the destination N_d receives a linear combination of flow f, it updates the id of the last received packet and the total number of received packets for the sender. Next, it stores the coded packet only if it is linear independent with all the packets received previously. Since the source is coding online, then the destination decodes a native packet on the fly, if possible, and passes it up to the TCP layer, as explained in *Algorithm 5*. If no native packet

Algorithm 4: **Coding at the relays.** This algorithm is used by a relay N_k when it receives a packet of flow f from neighbor N_l. m represents the number of packets from the coding_buffer$_f$. Coefficients c_i are randomly chosen from a finite field.

```
receive coded packet LC_r of flow f ;
s_kl = id(LC_r); t_kl ← t_kl + 1 ;
if LC_r is innovative then
    store LC_r in the coding buffer_f ;
    if receive credit associated to LC_r then
        generate LC_j = Σ_{i=1}^{m} c_i LC_i ;
        store LC_j in output buffer_f ;
    end
else
    drop LC_r ;
end
```

Algorithm 5: **On-the-fly decoding at the destination**. Destination N_d decodes native packets as soon as it receives independent linear combinations.

```
receive coded packet LC_k of flow f from neighbor N_l ;
s_dl = id(LC_k); t_dl ← t_dl + 1 ;
if LC_k is innovative then
    store LC_k in the coding buffer_f ;
    if a packet can be decoded then
        deliver the decoded packet to the TCP layer
    else
        send an ACK to the sender
    end
else
    drop packet LC_k ;
end
```

can be decoded, the destination acknowledges the reception of the current packet to the source.

3.4 Performance evaluation

We evaluate our protocol through extensive simulations with the network simulator *ns-2.33* [ns2] and compare it to improved versions of *Horizon* [RGGK08], *Square coding* as in [CJKK07, GHK+07], and *Online coding* [SSM+09b]. *Horizon* estimates the TCP window and signals congestion to the TCP sender

whenever the estimated value reaches a certain threshold. In our simulations, we feed *Horizon* with the actual TCP window size, not an estimate. Also, *MORE*[CJKK07] sends probe packets periodically to estimate the loss in the network and adjusts the rate of sending linear combinations accordingly. For our implementation of *Square coding* we use our algorithm to estimate losses instead, eliminating the overhead of sending probes. For *Online coding*, only the source sends redundant linear combinations and the authors state that the redundancy factor R should be adjusted online. Since the authors do not give any indication on how to adjust R online, we use our loss estimation algorithm *Algorithm 1* to adjust the rate of sending coded packets in the network.

3.4.1 Simulation setup

To illustrate some practical issues, such as the accuracy of the loss estimation algorithm, we use the diamond topology shown in *Figure 3.1*. For this scenario, we vary the probability of having a loss between any two nodes, p_{loss}, across experiments. As the propagation model, we use the *Two-ray ground* model.

We further run the protocols over the Roofnet topology using the location information provided at [Roo]. The losses are given by the Shadowing propagation model and depend on the distances between the nodes. We choose randomly 100 pairs of source–destination nodes and compute the routing information in MATLAB using a modified version of the directed diffusion algorithm, reinforcing up to d paths [LW06]. By varying the reinforced paths, we vary the number of downstream neighbors that a node can pick as next-hops for the packets. The average path length is of 3.5 hops.

We disable RTS/CTS for all experiments. In order to simulate heavy loss conditions, we also disable MAC layer retransmissions for all the experiments. The rest of the parameters have the default *ns-2.33* values. We set the transmission power and the carrier sensing threshold such that to obtain a transmission range of 400 m and carrier sensing range of 800 m. The rest of the parameters have the default *ns-2.33* values. In *Table 3.3* you can find a short summary of the settings for each of the topologies.

The metrics that we use to evaluate our protocol are the following:

- *throughput* measured at destination (decoded rate);
- *efficiency*, defined as the ratio of the goodput at destination to the total throughput transmitted in the network;

Topology	Settings
Diamond	- Two-Ray Ground propagation model - Transmission Range = 20 m - Carrier Sensing Range = 44 m - Area = 250 m^2 - variable p_{loss} across experiments - simulation time = 100 s
Roofnet	- Shadowing propagation model - Transmission Range = 400 m - Carrier Sensing Range = 800 m - p_{loss} depends on the propagation model and the distance between nodes - Area = 3 km^2 - simulation time = 50 s

Table 3.3: Summary of settings for each of the topologies used for simulations, diamond and Roofnet.

- *relative improvement* of *CoMP* over the other protocols, defined as the ratio of the throughput obtained with *CoMP* to the throughput obtained with the other protocols;
- *aggregate throughput* only for the case when multiple flows are running in parallel in the network. The aggregate throughput is computed as the sum of the individual throughput of all flows that run simultaneously.

For all the figures, each point is the average of at least 10 runs such that the standard deviations for the different protocols do not overlap.

3.4.2 Overall results

In this section we present the results for different scenarios and discuss how the four protocols, *CoMP*, *Horizon*, *Square coding* and *Online coding* compare one with another. We start with an evaluation of the *Algorithm 1* to estimate the loss between two neighbors in *Section 3.4.2.1*, then we discuss the results for using multiple paths in *Section 3.4.2.2* and for having multiple flows running in parallel, in *Section 3.4.2.3*.

3.4.2.1 Performance of the loss estimation algorithm

We evaluate our loss estimation algorithm for the topology in *Figure 3.1*, where two relays forward packets to the destination. For the *optimal* curve, the nodes

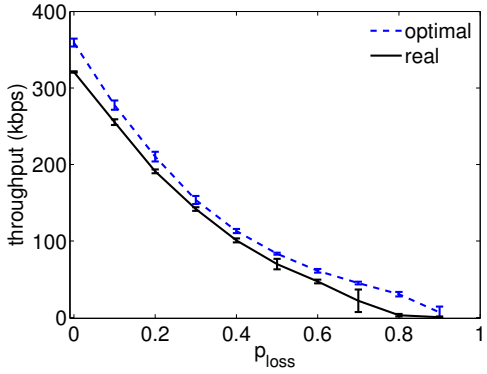


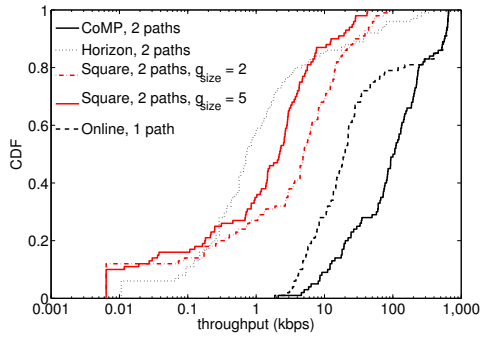
Figure 3.7: Evaluation of the loss estimation algorithm for the topology in *Figure 3.1*.

send packets at the optimal redundancy rate, which is computed based on the given loss probability. For the *real* case when the nodes use *Algorithm 1* to distributively adjust the rate of sending coded packets, the throughput is about $7\% - 15\%$ lower than optimal, when losses in the network are small. In extreme conditions, i.e. $p_{loss} > 0.6$, the throughput with the loss estimation algorithm is up to 72% less than optimal. Note that the loss estimation depends on the feedback received from neighbors. If the network conditions are harsh, then the nodes do not receive the necessary information as soon as they need it, therefore the rate is adjusted slowly.

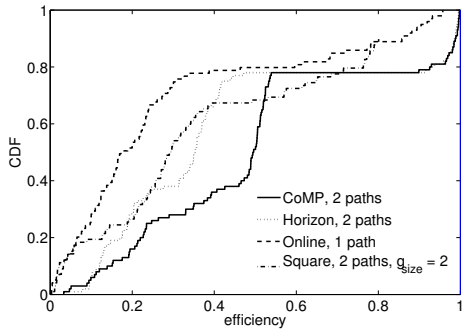
3.4.2.2 Multiple paths

In this case, there is only one flow active in the network at a time, and the packets are sent using up to 2 paths in parallel. Note from *Figure 3.8a* that *CoMP* achieves significantly higher throughput than the other protocols, and it can yield an improvement of 10x over *Online coding* and more for *Horizon* and *Square coding* (see *Figure 3.8c*). There are about 20% of flows in the network for which the source and the destination are one hop away, and for those flows *Online coding* has a similar performance to *CoMP*. If the paths are longer, then *Online coding* is more sensitive to losses. *Horizon* achieves a lower throughput because it does not perform error control and it relies on TCP retransmissions to recover every lost packet. The throughput with *Square coding* decreases significantly if the generation size increases due to the additional delay introduced by the coding operations at the sender.

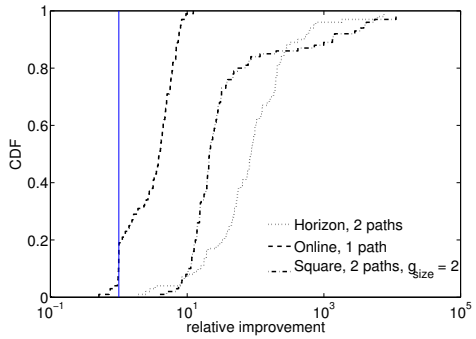
Regarding the efficiency, we compute it as the ratio of the decoded rate



(a) Throughput



(b) Efficiency



(c) Relative improvement

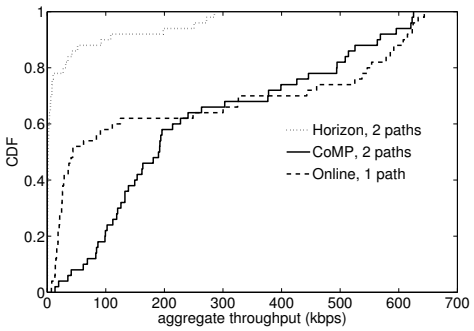
Figure 3.8: Roofnet topology with only one flow active in the network. Nodes have at most two neighbors, $d = 2$.

at destination to the total rate sent into the network, and therefore for a 2-hop long flow, the maximum efficiency can be equal to 0.5. *CoMP* generally outperforms the other protocols, except for the 20% flows that are one-hop long and *Online coding* is equally efficient (see *Figure 3.8b*).

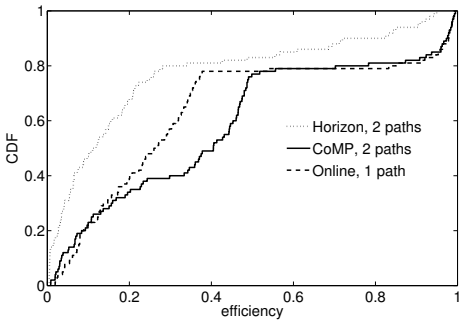
3.4.2.3 Multiple flows

In *Figure 3.9* we show the results for the case when multiple flows run in parallel in the network. Since the performance of *Square coding* degrades very quickly as the generation size increases, we do not take it into account for this case.

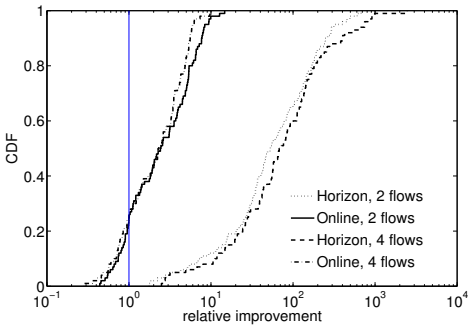
Notice from *Figure 3.9a* that for 60% of the flows, the aggregate throughput obtained when two sessions run concurrently in the network is higher with *CoMP* than with *Online coding*. This happens for performance-challenged



(a) Aggregate throughput for 2 flows



(b) Efficiency for 4 parallel flows

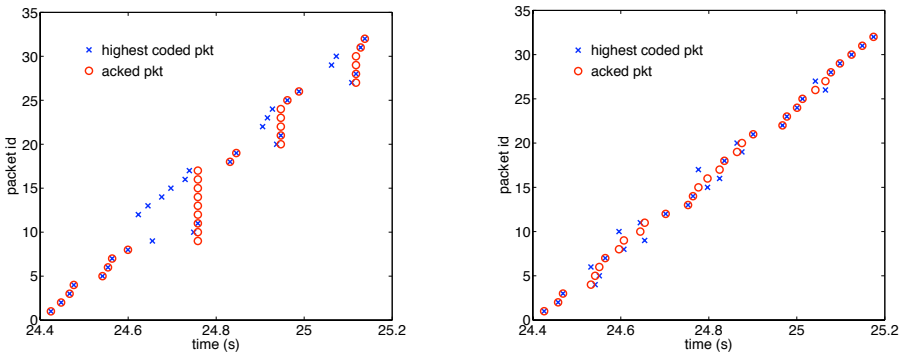


(c) Relative improvement

Figure 3.9: Roofnet topology with multiple flows running in parallel. Nodes have at most two neighbors, $d = 2$.

flows, where paths are long and experience high losses. For these situations, using one extra path together with coding hop-by-hop instead of end-to-end increase throughput. On the other hand, for the case when paths are short and possibly physically overlapping, using one more path increases the interference, thus having a negative effect on the throughput obtained with *CoMP*. The effect of the interference can also be noticed from *Figure 3.9c*, where 20% of the flows have a relative improvement less than 1, meaning the throughput is lower with *CoMP* than with *Online coding*. *Horizon* achieves a throughput close to 0 for almost 80% of the flows, due to the fact that it does not offer any reliability in an extremely volatile environment.

In terms of efficiency, *CoMP* outperforms the other protocols for 55% of the flows. For the 20% of one-hop flows, *Online coding* is equally efficient, since multiple paths cannot be exploited. There are another 25% of flows for which



(a) *Feedback mechanism on decoded rate*: packets are acknowledged when decoded, in bursts.

(b) *Feedback mechanism on innovative rate*: the destination acknowledges the reception of every innovative linear combination.

Figure 3.10: The effect of the feedback mechanism. A blue cross represents the highest *id* of the native packets encoded in the linear combination that arrived at the destination. A red circle represents the *id* of a native packet that is acknowledged by the destination. If a circle and a cross overlap, it means that the packet arrives at destination in order and it is decoded right away.

CoMP is slightly less efficient than *Online coding*, because even if with *CoMP* the destination receives a higher throughput, there are also more packets sent into the network, lowering efficiency.

3.5 The feedback mechanism for TCP revisited

As we explained in *Section 3.2.2*, *CoMP* uses a *feedback mechanism on innovative rate*, introduced in [SSM+09b], to alleviate the effect of the *decoding delay* on the behavior of the TCP sender. Through this mechanism the destination informs the sender that it receives coded packets, although it can not decode a native packet right away. In the following we take a closer look at this method, with the goal to clarify the implications of integrating it in the protocol stack.

Figure 3.10 shows the effect of using the *feedback mechanism on innovative rate* as opposed to the *feedback mechanism on decoded rate* (i.e. acknowledging a packet once it is decoded). One blue cross represents the highest *id* of the native packets mixed in a linear combination that arrives at the destination. One red circle represents the *id* of a native packet that the destination acknowledges to the sender. With the *feedback on decoded rate*, linear combinations sometimes pile up at the receiver, only to be decoded when a coded packet with the

missing degree of freedom reaches the destination. This causes the TCP layer at the receiver to see packets on a bursty pattern, and the TCP sender to receive TCP ACKs with delay, thus increasing the round-trip time. With the *feedback on innovative rate*, the destination acknowledges every reception of an innovative linear combination, regardless of whether it can decode a native packet or not. This results in a smoother rate of delivery for TCP. Note that in both cases the capacity of the network, i.e. the actual number of independent linear combinations that can be sent to the destination, is the same.

However, the *feedback mechanism on innovative rate* requires changes to the TCP protocol, to take into account the rate of arrivals rather than the rate of decoding the native packets, which does not respect the end-to-end principle of TCP. Furthermore, such changes to the TCP protocol cannot readily be implemented in practice. In order to overcome the issue of the delay introduced by the coding operations, without the need to modify the TCP layer, in this section we propose an enhancement for *CoMP* – our scheme proposed in *Section 3.3*. This enhancement consists of disabling the *feedback module (on innovative rate)* shown in dashed in *Figure 3.6* and changing the network coding block to account for generations of packets mixed by means of *triangular coefficient matrices*. In particular, our approach – further denoted by *CoMP tri* – encodes packets on the fly in order to eliminate the *coding delay* at the sender and it relies on the *feedback on the decoding rate* at the receiver. We compare *CoMP tri* with *Square coding*, *CoMP*, and *CoMP with feedback on decoded rate*, through simulations with *ns-2.33* [ns2]. The main goal is to provide insights about how network coding can be used in conjunction with TCP, without any changes to the transport layer, thus without using the feedback on the innovative rate, but on the decoding rate at the destination. Our findings can be summarized as follows:

- *CoMP tri*, that is, network coding using *triangular coefficient matrices* for generations of packets, significantly outperforms *Square coding* and *CoMP with feedback on decoded rate*.
- *CoMP* achieves higher throughput if the source and the destination are closer, in terms of hops. However, in extremely lossy environments, when the sender and the receiver communicate over multiple hops, *CoMP tri* has the best performance.
- Traditional network coding schemes based on *full square coefficient matrices* do not interact well with TCP, due to the *coding delay* at the sender.

3.5.1 Network coding with triangular coefficient matrix

For the coding operations at the sender, we propose using *lower triangular coefficient matrices*[2] for generations of packets. This approach is a combination of the *square coefficient matrix* scheme (discussed in *Section 3.2.1*) and the *online coding* scheme (discussed in *Section 3.2.2*), in the sense that the sender splits the traffic in generations, but it performs *incremental coding*. This means the sender generates and forwards a linear combination as soon as a new packet is produced. Incremental coding has the advantage that the sender can transmit packets on the fly, i.e. no arbitrary delay is introduced at the source for the coding operations. Moreover, in the ideal case when no losses occur in the network, the lower triangular structure is preserved at the receiver, thus there is no need to perform Gaussian elimination to solve the resulting linear system. In this case, the *coding window* increases by one every time a new native packet arrives to the *coding buffer*, until it reaches the size of the generation.

Table 3.4: Network coding using a *lower triangular coefficient matrix* for a generation of 3 packets.

Time slot	**Source**			**Destination**		
	Recv	Coding Bfr.	Sent	Recv	Decoded	Sent
t_1	P_1	P_1	$C_1 = c_{11}P_1$			
t_2	P_2	$P_1,\ P_2$	$C_2 = c_{21}P_1 + c_{22}P_2$	C_1	P_1	Ack_1
t_3	$Ack_1,\ P_3$	$P_1,\ P_2,\ P_3$	$C_3 = c_{31}P_1 + c_{32}P_2 + c_{33}P_3$	C_2	P_2	Ack_2
t_4	Ack_2	$P_1,\ P_2,\ P_3$		C_3	P_3	Ack_3
t_5	Ack_3					

For the scenario described in *Section 3.2*, with a sender and a receiver, the source can send linear combinations from the first time slot, thus eliminating the *coding delay*, as shown in *Table 3.4*. Moreover, the destination can decode a native packet every time it receives a linear combination, hence eliminating the *decoding delay* as well. In this case, the generation is decoded after 4 time units, which represents a decrease in the total time by 33% as compared to using a *square coefficient matrix*. The sender flushes the *coding buffer* in time slot t_5, after receiving the ACKs for all the packets from the generation.

Finally, note that for this simple and ideal case (i.e. one-hop communication without losses), *triangular coding* achieves the same performance as *online cod-*

[2]The idea of triangular coding is also discussed in [RN04], but in the context of source coding with FEC.

Figure 3.11: Chain topology, where N_0 is sending traffic to N_n, through relays $N_i, 0 < i < n$.

ing, described in *Section 3.2.2*; however, for the case of wireless transmissions over unreliable links, we will see in the following section that using *triangular coding* outperforms *online coding* in certain conditions.

3.5.2 Results

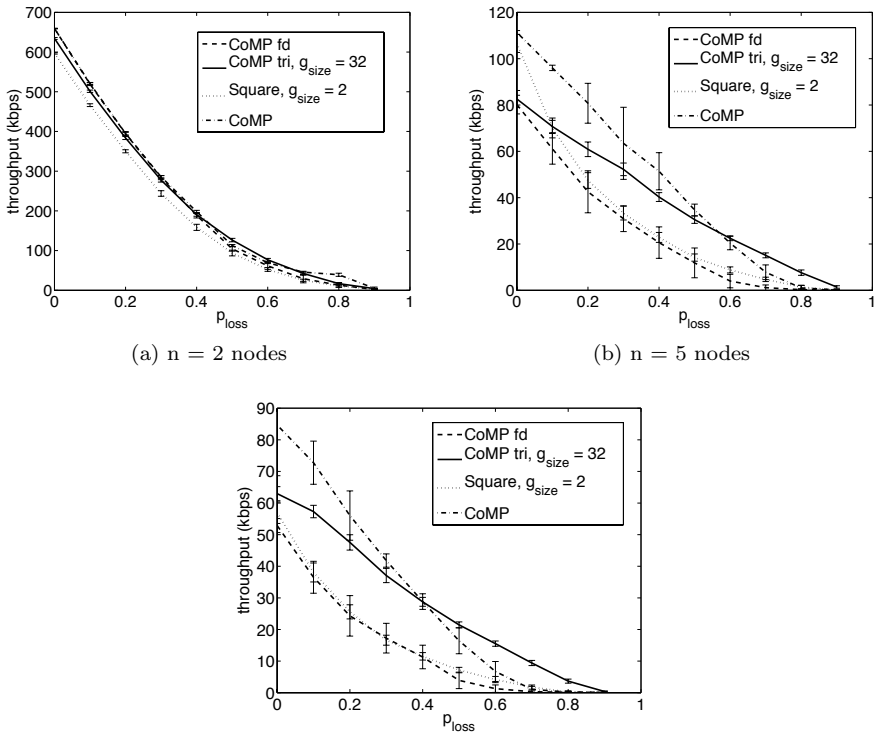
We evaluate the *CoMP tri* scheme proposed in *Section 3.5.1* by comparing it with *Square coding*, *CoMP with feedback on decoded rate* (further denoted by *CoMP fd*) and *CoMP*. As we will see in the following, the *Square coding* scheme is mainly affected by the *coding delay*, hence it cannot benefit from the use of the feedback on innovative rate.

We run extensive simulations in *ns-2.33* [ns2] over a chain topology illustrated in *Figure 3.11* and a realistic topology, the Roofnet [Roo]. In the chain topology, the source N_0 sends coded packets to destination N_n, using nodes $N_i, 0 < i < n$ as relays. We vary the probability of having a loss between any two nodes, p_{loss}, and the number of relays, n, across experiments. As the propagation model, we use the *Two-ray ground* model. The Roofnet topology was explained in *Section 3.4.1*. For all the figures, each point is the average of multiple runs, 50 runs for the chain topology and 10 runs for the Roofnet.

In *Figure 3.12* we show the average throughput with variable probability of loss in each link, for a different number of nodes in the chain topology. Note that for a topology that contains only the source and the destination, the network coding techniques under consideration achieve a similar throughput, depicted in *Figure 3.12a*. However, as the path length and the p_{loss} increase (*Figure 3.12b* and *Figure 3.12c*), the differences in throughput obtained with each scheme become obvious.

In order to understand these differences, in the following we analyze the behavior of the TCP protocol for one run of each of the coding techniques, for the case of having $n = 9$ nodes in the network and for the loss probability $p_{loss} = 0.5$. For each approach, we plot the *coding window*, the TCP congestion window, the time when TCP ACKs arrive at the sender, and the round-trip time (RTT) measured at the TCP layer.

We begin our analysis with the observation that *Square coding* works with



(a) n = 2 nodes

(b) n = 5 nodes

(c) n = 9 nodes

Figure 3.12: Average throughput with the increasing p_{loss}, for the chain topology with variable number of nodes.

TCP only if small generations are used, e.g. $g_{size} = 2$. For larger generations, the sender needs to wait longer to receive the packets from a generation, hence the *coding delay* increases. Consequently, the TCP protocol experiences timeouts and retransmissions even before all the packets from a generation arrive at the network coding layer. Notice that because of this issue, the *square coding* would not benefit from the use of the feedback mechanism to acknowledge the reception of innovative packets. For the experiment illustrated in *Figure 3.13*, the first packets from the transmission are lost before reaching the destination, thus the connection picks up later, after $t = 20$ seconds, and the overall throughput is low.

In the case of *CoMP* regardless of whether the sender has *feedback on decoded rate* (*Figure 3.14*) or *on innovative rate* (*Figure 3.15*), the *coding window* increases in close relationship with the value of the TCP window, and

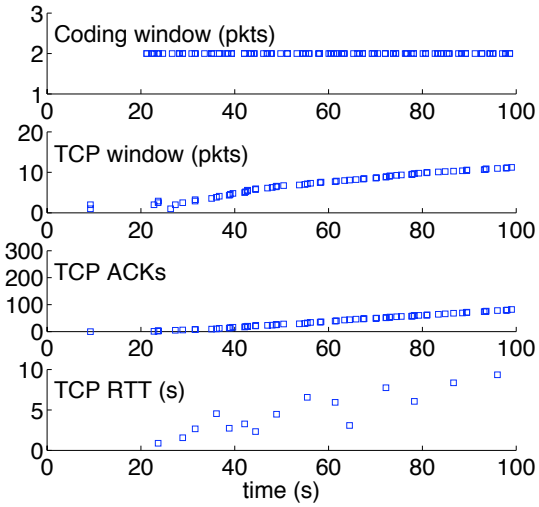


Figure 3.13: *Square coding* – one run for $g_{size} = 2$, $n = 9$ nodes, $p_{loss} = 0.5$.

decreases as TCP ACKs reach to the sender. If the *feedback is on decoded rate* (*Figure 3.14*), due to the lossy links some of the coded packets do not arrive at the destination, thus the destination is not able to decode as it does not have enough independent linear combinations. Eventually, the destination receives the missing degree of freedom and it decodes the native packets in bursts and with some delay, which causes the TCP ACKs to arrive at the sender with an irregular rate. Moreover, this high *decoding delay* determines the TCP sender to time out and eventually reduce its transmission window, leading to a reduction in throughput.

For *CoMP* (i.e. *with feedback on innovative rate*), by allowing the destination to send feedback to the source about the rate of receiving coded packets, instead of the rate of decoding packets, the sender underestimates the RTT and overestimates the capacity on the wireless links. Consequently, the above problem is still not solved, and the TCP sender keeps pushing new packets, at a faster pace than the destination is actually able to decode. Finally, the TCP sender experiences a time out, reduces its transmission window and retransmits outstanding packets. This situation can be seen in *Figure 3.15*, around $t = 20$, $t = 40$ and $t = 80$ seconds.

For *CoMP tri*, notice that the *coding window* increases up to the generation size, as shown in *Figure 3.16*. Thus the number of unknowns introduced in the system is kept within a limit, which allows the destination to decode the

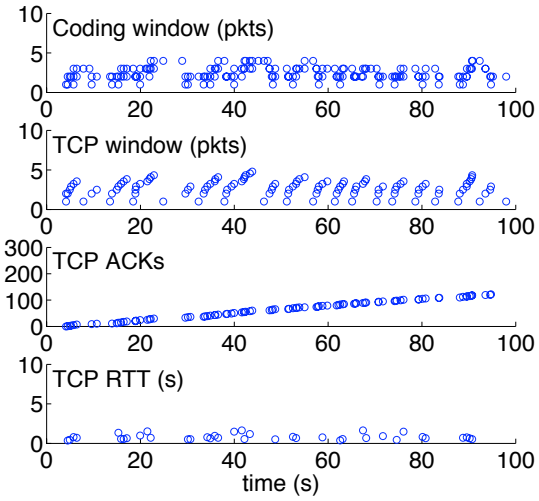


Figure 3.14: *CoMP fd (with feedback on decoded rate)* – one run for $n = 9$ nodes, $p_{loss} = 0.5$.

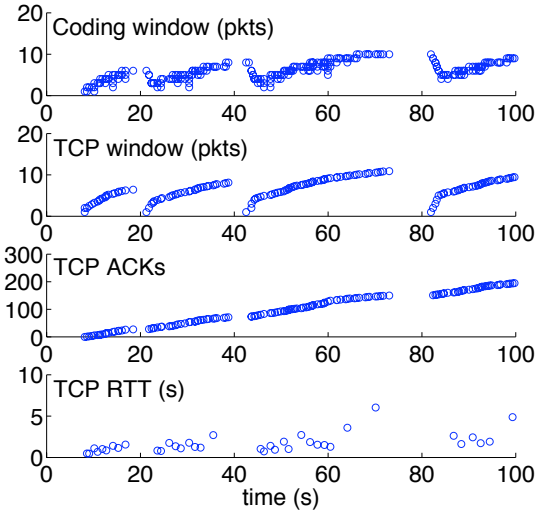


Figure 3.15: *CoMP (with feedback on innovative rate)* – one run for $n = 9$ nodes, $p_{loss} = 0.5$.

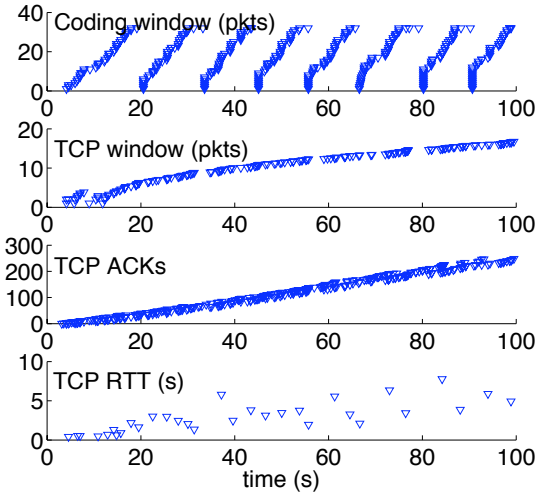


Figure 3.16: *CoMP tri* – one run for $n = 9$ nodes, $p_{loss} = 0.5$.

received linear combinations before the source mixes new native packets and sends them into the network. Further, the TCP ACKs reach the sender on a continuous basis, and the TCP window increases smoothly, maintaing a high throughput, even in the presence of high losses in the network.

For the Roofnet topology, we disable MAC layer retransmissions in order to simulate an extremely volatile environment. In *Figure 3.17* we show the average throughput (in kbps) with the number of nodes on the path from the source to the destination. If $n = 5$, that means the sender and the receiver are 4 hops away. Note that *Square coding* achieves a very low throughput, even if the size of the generation is small, $g_{size} = 2$. The throughput achieved with *CoMP with feedback on decoded rate* drops significantly as $n \geq 4$, while the throughput achieved with full-featured *CoMP* is approximately 3x higher than that with *Square coding* and 2x higher than that with *CoMP tri* when the source and the destination are two hops away. However, as the path length increases, *CoMP tri* achieves the highest throughput; for example, for the topologies where the sender and the receiver are 3 hops away, i.e. $n = 4$, *CoMP tri* achieves a throughput that is 4x higher than that obtained with *Square coding*, 3x higher than that obtained with *CoMP with feedback on decoded rate* and 30% higher than that obtained with *CoMP*. The benefit comes from the fact that the TCP sender estimates the link capacities correctly and sends new packets at a sustainable rate, as opposed to *CoMP*, where the TCP sender pushes new

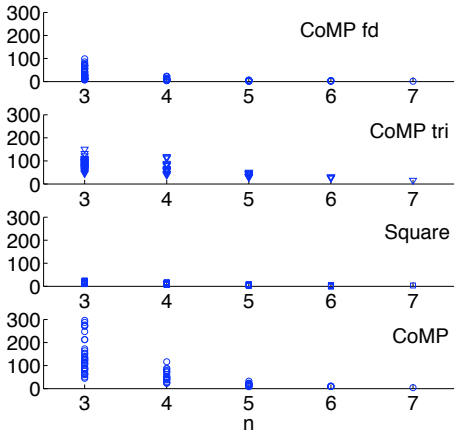


Figure 3.17: Average throughput (in kbps) for the Roofnet topology, using 2 paths between any source-destination pair.

packets in the network at a faster rate than what the receiver is able to decode.

3.6 Concluding remarks

In this chapter we presented *CoMP*, a network coding multipath protocol that improves TCP performance in lossy environments by combining online coding and multipath techniques. *CoMP* uses an online algorithm to estimate current loss conditions in the network and to adjust the rate of sending coded packets. *CoMP* relies on a feedback mechanism for TCP, where destination acknowledges the independent linear combinations that it receives, instead of decoded packets. All these features allow a seamless interaction between *CoMP* and TCP. We show through extensive simulations the advantages of our protocol over the current state-of-the-art in terms of throughput and efficiency.

Furthermore, concerned with the practical aspects of implementing our scheme in the protocol stack, we extended the network coding module of *CoMP* to use *triangular coefficient matrices* for generations of packets, in order to eliminate the *coding delay* and avoid any modification of the TCP layer. Using this feature, our scheme achieves a high throughput, even if the source has knowledge only about the decoding rate at the destination. We compared *CoMP tri* with *Square coding*, *CoMP* (with *feedback on innovative rate*, where the sender is informed about the rate at which innovative packets arrive at destination), and *CoMP fd* (with *feedback on decoded rate*, where the sender is

informed about the rate of decoding at destination). We showed through simulations that *CoMP tri* outperforms *Square coding* and *CoMP fd*. Moreover, we show that for those scenarios where the sender and the receiver communicate over more hops in extremely volatile environments, performing network coding with *triangular coefficient matrices* is better than *online coding with feedback on innovative rate* in terms of throughput. If the communication occurs over a small number of hops, then *CoMP (with feedback on innovative rate)* achieves the highest throughput. The *Square coding* scheme works only if very small generation sizes are used, for sender–receiver pairs that do not communicate over multiple hops.

As future work, we will analyze these network coding schemes in the context of multicast applications. We will also focus on the design of an algorithm to automatically adjust the coding window, taking into account characteristics such as path length or round-trip time.

Part II

Point-to-Multipoint Communications

CHAPTER 4

Achieving Optimal Rates for Degraded Multicasting

Content delivery, i.e. multicasting, is an application where network coding promises to have impact, as significant benefits have been observed both theoretically as well as in practice. The case where all receivers require the exact same content is by now well understood; however, for the (perhaps more realistic) case, where different users require different subsets of the content, although there exist a number of proposed heuristic algorithms, there is in general no exact characterization of the optimal achievable rate region [FS06b].

In this chapter, we provide such a characterization for the degraded two-message set problem, where a source broadcasts two messages to a set of receivers over a combination network with erasure channels. Degraded broadcasting refers to that the "weaker" receivers obtain only a subset of the information that the "stronger" users collect. That is, the weaker users require a message W_1, transmitted at a rate R_1, while the stronger users require not only W_1, but also a second message W_2, transmitted at a rate R_2. Degraded broadcasting is motivated by various scenarios, such as video streaming applications, or broadcasting in the presence of fading. In the first case, users are heterogeneous and have different subscription levels, thus requiring a different resolution of the content [GLL+10]. In the second case, the receivers are not able to receive the whole content due to channel fading, that can be modeled as erasures at higher layers.

The problem we solve is a specific case of a long-standing open question in

multi-user information theory, of delivering a set of degraded messages over a general broadcast channel introduced in [Cov72]. Although special cases have been addressed [Ber74, Gal74, KM77], there is comparatively little understanding when there are more than two users. Recent progress on a particular case of this question has been made in [NE09]. Closer to our work is the one in [SDFP09] that examines two-message broadcasting over a linear deterministic channel; our work differs in that we specifically look at the combination network, incorporate erasures, and provide a simpler achievability scheme.

Our main contributions in this chapter are:

- We present a very simple scheme to achieve the rate region for the two-degraded message-set problem, over the combination network and with three receivers. This scheme assigns source messages (or their linear combinations) to the network edges in polynomial time.

- We provide an analytical comparison with an approach that is oblivious to the topology, and highlight what are the situations where the optimal approach can offer benefits.

A main observation from our work is that, to achieve the optimal rates, we need to take into account topological information, namely, what subset of receivers observes each edge. In addition, in order to achieve the optimal performance, we only need to use very simple binary network coding at a subset of the network edges.

The chapter is organized as follows. We formulate the problem in *Section 4.1* and give the characterization of the rate region $\mathcal{R}_G^\alpha$ for a combination network G, in the presence of erasures of rate α in *Section 4.2*. In *Section 4.3* we introduce an algorithm that uses topological information to achieve any rate pair $(R_1, R_2) \in \mathcal{R}_G^\alpha$. *Section 4.4* shows an analytical comparison between our algorithm and a network coding approach, where the resources are allocated without any knowledge of the topological information. We conclude with some final remarks and directions for future work in *Section 4.5*. For the rest of the chapter, we use the terms "edge" and "resource" interchangeably.

The work in this chapter has as a starting point the work by Saeedi et al. [SDFP09] and the results further presented were developed in a joint work with Saeedi et al. This work has been accepted for publication [GSFL11].

4.1 Problem formulation

The problem of interest is communication of a public message W_1 and a private message W_2 at rates R_1, and R_2 respectively, to a set of three receivers, $\mathcal{U} = \{1, 2, 3\}$. The transmission is performed over a combination network G, illustrated in *Figure 4.1*, where each channel has an erasure probability α and each receiver i has access to r_i edges. Message W_1 is required at all destinations, while message W_2 is only required at one of them, say the third receiver. Under this scenario, we set out to characterize the rate region $\mathcal{R}_G^{\alpha}$ at which messages W_1 and W_2 can be reliably communicated to the three receivers.

In this chapter, we let $\mathcal{E}$ denote the total set of the intermediate edges, and $\mathcal{E}_i \subseteq \mathcal{E}$ denotes the set of the edges visible only to receiver i. Similarly $\mathcal{E}_{ij} \subseteq \mathcal{E}$ contains the edges visible only to receivers i and j and $\mathcal{E}_{ijk}$ is the set of edges visible to all three of the receivers. With this notation, we have that: $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \mathcal{E}_3 \cup \mathcal{E}_{12} \cup \mathcal{E}_{13} \cup \mathcal{E}_{23} \cup \mathcal{E}_{123}$, where each edge $e \in \mathcal{E}$ is visible to at least one receiver and it belongs to exactly one of the defined subsets.

Finally, we assume the size of the field over which the coding operations are performed is large enough, such that the linear combinations sent over the outgoing edges, if chosen randomly, are independent with high probability. Thus, the number of linear independent combinations received by each destination i is equal to r_i, the min-cut to each destination, and it is given by: $r_i = |\mathcal{E}_i| + \sum_{j \in \mathcal{U}, j \neq i} |\mathcal{E}_{ij}| + |\mathcal{E}_{123}|$. In particular,

$$r_1 = |\mathcal{E}_1| + |\mathcal{E}_{12}| + |\mathcal{E}_{13}| + |\mathcal{E}_{123}| \tag{4.1}$$

$$r_2 = |\mathcal{E}_2| + |\mathcal{E}_{12}| + |\mathcal{E}_{23}| + |\mathcal{E}_{123}| \tag{4.2}$$

$$r_3 = |\mathcal{E}_3| + |\mathcal{E}_{13}| + |\mathcal{E}_{23}| + |\mathcal{E}_{123}| \tag{4.3}$$

We also denote with r_{ij} the size of the union of the edges that two destinations i and j, $i \neq j$, observe. The received signal at receiver i is given by $\bar{Y}_i = [y_{i,1} \cdots y_{i,r_i}]^t$ where $y_{i,j}$ is the signal received on the j^{th} incoming edge of destination i. By $\bar{Y}_i^n = [y_{i,1}^n \cdots y_{i,r_i}^n]^t$ we denote the received signals at receiver i during a block length n.

4.2 Capacity region

In this section, we give the characterization of the capacity region for the degraded two message set scenario over a combination network with three re-

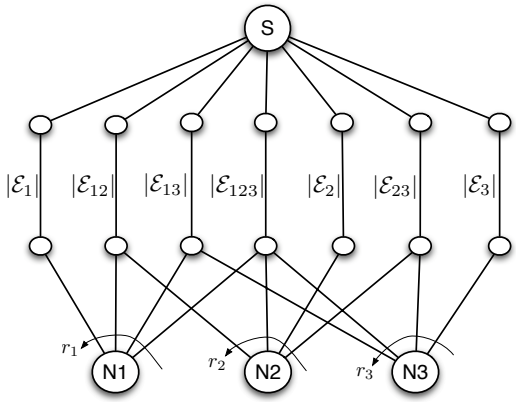


Figure 4.1: Combination network with one source and three receivers. For clarity, we represent every set $\mathcal{E}_{\{.\}}$ using only one edge, and indicate set cardinality on the left side of that edge. Each receiver i has access to r_i edges.

ceivers. We also introduce a polynomial time algorithm which gives the encoding scheme to achieve any rate pair (R_1, R_2) in that rate region.

Theorem 1. *Any achievable rate pair (R_1, R_2) in the degraded two message set scenario, applied over a combination network G with channels of independent erasure probability α lies in the region R_G^α characterized by*

$$R_1 \leq (1-\alpha)\min\{r_i\} \tag{4.4}$$

$$R_1 + R_2 \leq (1-\alpha)r_3 \tag{4.5}$$

$$2R_1 + R_2 \leq (1-\alpha)(r_1 + r_2 + |\mathcal{E}_3|) \tag{4.6}$$

Theorem 2. *Any rate pair $(R_1, R_2) \in \mathcal{R}_G^\alpha$ is achievable using the encoding scheme proposed by Algorithm 7.*

The proof of Theorem 1 is given by Shirin Saeedi, and we refer the reader to [GSFL11]. We provide the proof to Theorem 2 in *Section 4.3*.

4.2.1 Discussion

From the inequalities which characterize $\mathcal{R}_G^\alpha$, (4.4) and (4.5) are straightforward, as they essentially express min-cut conditions, while the third inequality and its effect on the rate region is more interesting. In this section, we take a closer look at Theorem 1 with the goal to better understand what is the result-

ing rate region for different values of the network parameters, $r_i, i \in \{1,2,3\}$ and $\mathcal{E}_{\{.\}}$.

Assume for simplicity that there are no erasures in the network, i.e. $\alpha = 0$, then the rate region $\mathcal{R}_G^0$ is given by:

$$\begin{aligned} R_1 &\leq \min\{r_i\} && (4.7)\\ R_1 + R_2 &\leq r_3 && (4.8)\\ 2R_1 + R_2 &\leq r_1 + r_2 + |\mathcal{E}_3| && (4.9) \end{aligned}$$

Depending on the topological parameters, the rate region could be determined by: (i) all three inequalities, or (ii) inequalities (4.7) and (4.8), or (iii) only inequality (4.8).

Case i The rate region is shown in *Figure 4.2a*. Intuitively the third inequality becomes tight if the r_1 edges to the first destination do not sufficiently overlap with the r_2 edges to the second destination. Therefore we may need to use twice the bottleneck edges in the combination network (hence the factor of 2) for W_1 to reach both these receivers. Then the rate R_2 that we can send to the third receiver is limited by the "leftover" edges,

$$R_2 \leq (r_1 - R_1) + (r_2 - R_1) + |\mathcal{E}_3|, \quad (4.10)$$

i.e. the edges that only the third receiver sees, and the edges remaining after duplicating message W_1 at rate R_1 to reach the first two receivers.

More formally, depending on the parameters of the topology, i.e. the number of edges in each set $\mathcal{E}_{\{.\}}$, the third inequality becomes active only for those topologies where the following situation occurs:

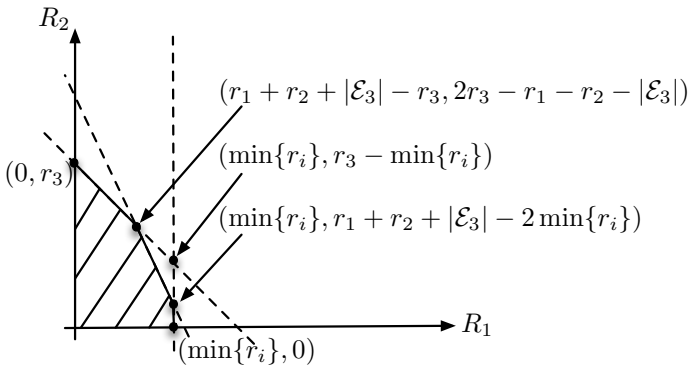
$$\min\{r_1, r_2, r_3\} + r_3 > r_1 + r_2 + |\mathcal{E}_3|. \quad (4.11)$$

Note that if $r_3 = \min\{r_1, r_2, r_3\}$, then the above relation does not hold, since $r_i \geq r_3, i \in \{1,2\}$. Therefore, r_3 does not affect the value of $\min\{r_1, r_2, r_3\}$ and we equivalently have the third inequality active when

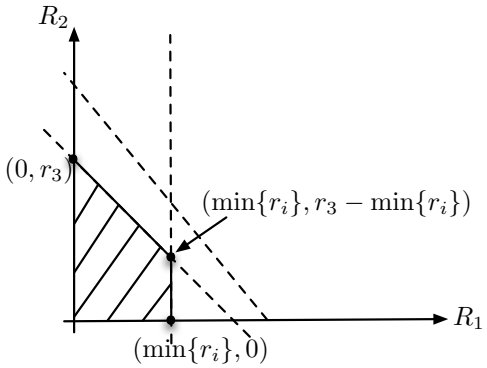
$$\min\{r_1, r_2\} + r_3 > r_1 + r_2 + |\mathcal{E}_3|. \quad (4.12)$$

Replacing the corresponding values of the ranks, we obtain that:

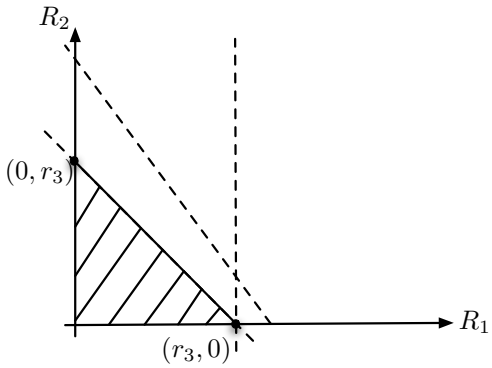
$$\min\{|\mathcal{E}_1| + |\mathcal{E}_{13}|, |\mathcal{E}_2| + |\mathcal{E}_{23}|\} > |\mathcal{E}_1| + |\mathcal{E}_2| + |\mathcal{E}_{12}|. \quad (4.13)$$



(a) Rate region described by all three inequalities



(b) Rate region described by inequalities (4.7) and (4.8)



(c) Rate region described by inequality (4.8)

Figure 4.2: Rate region – discussion. We assume $\alpha = 0$ for simplicity.

Figure 4.3 is an example of such topological parameters. We give in the following an algorithm to verify for a desired combination network if the third inequality becomes active. The proof can be found in [GSFL10]. It turns out that *Figure 4.3* is the canonical combination network with the third inequality active; i.e. *Algorithm 6* returns ACTIVE if and only if *Figure 4.3* is the combination network that remains after the edge eliminations up to that iteration.

Case ii The rate region $\mathcal{R}_G^0$ for this situation is illustrated in *Figure 4.2b*. This case occurs when either receiver 1 or receiver 2 is the bottleneck, and in addition the resources that 1 and 2 have in common are sufficient to deliver message W_1 to both of them, without using extra resources visible to the third receiver that could carry message W_2. More formally, and using the same

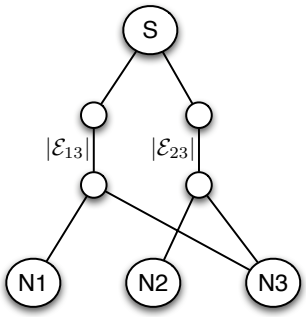


Figure 4.3: Canonical combination network for the case when inequality (4.6) is active. After running *Algorithm 6* on a given combination network, only sets $\mathcal{E}_{13}$ and $\mathcal{E}_{23}$ still contain edges.

Algorithm 6: **Third inequality**. This algorithm returns ACTIVE when the third inequality is active depending on the topological parameters and returns NOT ACTIVE otherwise.

while *TRUE* **do**
  if $|\mathcal{E}_{13}| = 0$ *OR* $|\mathcal{E}_{23}| = 0$ **then**
    return *NOT ACTIVE*
  else if $|\mathcal{E}_3| > 0$ **then**
    $|\mathcal{E}_3| \leftarrow |\mathcal{E}_3| - 1$;
  else if $|\mathcal{E}_{12}| > 0$ *AND* $|\mathcal{E}_{13}| > 0$ *AND* $|\mathcal{E}_{23}| > 0$ **then**
    $|\mathcal{E}_{13}| \leftarrow |\mathcal{E}_{13}| - 1$;
    $|\mathcal{E}_{23}| \leftarrow |\mathcal{E}_{23}| - 1$;
    $|\mathcal{E}_{12}| \leftarrow |\mathcal{E}_{12}| - 1$;
  else if $|\mathcal{E}_{13}| > 0$ *AND* $|\mathcal{E}_2| > 0$ **then**
    $|\mathcal{E}_{13}| \leftarrow |\mathcal{E}_{13}| - 1$;
    $|\mathcal{E}_2| \leftarrow |\mathcal{E}_2| - 1$;
  else if $|\mathcal{E}_{23}| > 0$ *AND* $|\mathcal{E}_1| > 0$ **then**
    $|\mathcal{E}_{23}| \leftarrow |\mathcal{E}_{23}| - 1$;
    $|\mathcal{E}_1| \leftarrow |\mathcal{E}_1| - 1$;
  else
    return *ACTIVE*
  end
end

argument as for the previous case – when the third inequality was tight, we have that:

$$r_3 \quad > \quad \min\{r_i\}$$

and

$$\min\{|\mathcal{E}_1| + |\mathcal{E}_{13}|, |\mathcal{E}_2| + |\mathcal{E}_{23}|\} \quad \leq \quad |\mathcal{E}_1| + |\mathcal{E}_2| + |\mathcal{E}_{12}|.$$

Case iii For those topologies where the third receiver is the bottleneck, in the sense that it has access to the minimum number of resources, i.e. $r_3 = \min\{r_i\}$, only inequality (4.8) is tight, while inequalities (4.7) and (4.9) are redundant. The rate region $\mathcal{R}_G^0$ in this case is shown in *Figure 4.2c*.

4.3 Algorithm description

In this section we introduce an algorithm that uses topological information in order to achieve any desired rate pair $(R_1, R_2) \in \mathcal{R}_G^\alpha$. The algorithm uses the fact that each intermediate edge is essentially one available resource to the set of receivers that are connected to it and can carry linear combinations of W_1 or W_2. We show that we do not need to perform network coding between W_1 and W_2 in order to have an optimal algorithm (our *Algorithm 7* is such an example). For the sake of simplicity we consider the case of no erasures in *Section 4.3.1* and give the sketch of the proof for the case where each channel has an independent and uniform erasure probability of α in *Section 4.3.2*.

The idea of the algorithm is that the source puts linear combinations of symbols of W_1 or of W_2 on each of the edges so that it guarantees decodability of W_1 at all the receivers and decodability of W_2 at the third receiver. We are interested in assigning each resource to carry one of the two messages. We indicate this by coloring the intermediate edges with two colors, t_1 for W_1 and t_2 for W_2, where $t_1 \neq t_2$ and both $t_i > 0, i \in 1, 2$. This edge assignment (edge coloring) is the output of our proposed *Algorithm 7* for a given rate pair $(R_1, R_2) \in \mathcal{R}_G^0$. The algorithm makes use of two methods, which we explain briefly. Function *FindEdge($\mathcal{A}$)* given in *Algorithm 8* returns true if the set $\mathcal{A}$ contains at least an edge that has not been assigned for any message yet. Function *ColorEdge($\mathcal{A}, t_i$)* given in *Algorithm 9* marks an edge of the specified set $\mathcal{A}$ to carry message W_i.

One should note that network coding is actually needed only for step 5 of *Algorithm 7*, when it assigns resources from each of the sets visible to any two receivers, $\mathcal{E}_{ij}$. By selecting an edge from each $\mathcal{E}_{ij}$, and sending a linear

Algorithm 7: **Resource assignment**. This algorithm assigns either t_1 or t_2 to each of the available resources, for a given rate pair $(R_1, R_2) \in \mathcal{R}_G^{\alpha}$.

Input : $(R_1, R_2) \in \mathcal{R}_G^{\alpha}$
Initialize: $c_e \leftarrow 0,\ \forall e \in \mathcal{E}$
while $R_1 > 0$ **do**
  if *FindEdge*($\mathcal{E}_{123}$) **then**
    $R_1 \leftarrow R_1 - 1$;
    ColorEdge($\mathcal{E}_{123}, t_1$)
  else if *FindEdge*($\mathcal{E}_{13}$) *AND FindEdge*($\mathcal{E}_{23}$) *AND FindEdge*($\mathcal{E}_{12}$) **then**
    if $R_1 \geq 2$ **then**
      $R_1 \leftarrow R_1 - 2$;
      ColorEdge($\mathcal{E}_{13}, t_1$);ColorEdge($\mathcal{E}_{23}, t_1$);ColorEdge($\mathcal{E}_{12}, t_1$)
    else
      $R_1 \leftarrow R_1 - 1$;
      ColorEdge($\mathcal{E}_{13}, t_1$); ColorEdge($\mathcal{E}_{12}, t_1$) ;
    end
  else if *FindEdge*($\mathcal{E}_i$) *AND FindEdge*($\mathcal{E}_{jk}$), $\{i, j, k\} = \{1, 2, 3\}$ **then**
    $R_1 \leftarrow R_1 - 1$;
    ColorEdge($\mathcal{E}_i, t_1$); ColorEdge($\mathcal{E}_{jk}, t_1$) ;
  else if *FindEdge*($\mathcal{E}_1$) *AND FindEdge*($\mathcal{E}_2$) *AND FindEdge*($\mathcal{E}_3$) **then**
    $R_1 \leftarrow R_1 - 1$;
    ColorEdge($\mathcal{E}_1, t_1$); ColorEdge($\mathcal{E}_2, t_1$); ColorEdge($\mathcal{E}_3, t_1$);
  else if *FindEdge*($\mathcal{E}_{ij}$) *AND FindEdge*($\mathcal{E}_{ik}$), $\{i, j, k\} = \{1, 2, 3\}$ **then**
    $R_1 \leftarrow R_1 - 1$;
    ColorEdge($\mathcal{E}_{ij}, t_1$); ColorEdge($\mathcal{E}_{ik}, t_1$);
  end
end
Assign R_2 edges from the remaining edges visible to receiver 3 to carry W_2

Algorithm 8: **FindEdge($\mathcal{A}$)**: returns true if $\mathcal{A}$ contains at least an edge that is not assigned yet.

foreach $e \in \mathcal{A}$ **do**
  if $c_e == 0$ **then**
    return TRUE
  end
end
return FALSE

combination of W_1 on each of them, every destination receives a total rate of two. For the remaining situations, it is enough to route by conveniently

Algorithm 9: **ColorEdge**$(\mathcal{A}, t_i)$: marks an edge of the specified set $\mathcal{A}$ to carry W_i.

```
foreach e ∈ A do
    if c_e == 0 then
        c_e ← t_i ;
        return
    end
end
```

selecting one edge from those sets that complement each other, for example sets $\mathcal{E}_2$ and $\mathcal{E}_{13}$ as long as the sets still contain edges that have not been assigned yet.

4.3.1 Algorithm optimality - no erasures

Lemma 1. *Algorithm 7 stops after finite steps.*

Proof. We first prove that after each iteration (inside the while loop) R_1 is decreased by at least 1. We then conclude that *Algorithm 7* stops after at most R_1 iterations. In each iteration, R_1 is decreased if either of the "IF conditions" are satisfied. No "IF condition" is satisfied only when all $|\mathcal{E}_{123}|$, $\min\{|\mathcal{E}_{12}|, |\mathcal{E}_{13}|, |\mathcal{E}_{23}|\}$, $\min\{|\mathcal{E}_i|, |\mathcal{E}_{j,l}|\}$, $\min\{|\mathcal{E}_1|, |\mathcal{E}_2|, |\mathcal{E}_3|\}$ and $\min\{|\mathcal{E}_{i',j'}|, |\mathcal{E}_{i',l'}|\}$ are already assigned which ensures R_1 having been decreased by at least r_1 or r_2. But then since $(R_1, R_2) \in \mathcal{R}_G^0$, R_1 should satisfy $R_1 \leq \min\{r_1, r_2\}$ which means that $R_1 \leq 0$ in the studied iteration and this is a contradiction. □

Lemma 2. *Algorithm 7 is optimal.*

Proof. We prove here that for any $(R_1, R_2) \in \mathcal{R}_G^0$, the assignment proposed by *Algorithm 7* lets (i) all receivers get R_1 random linear combinations of W_1 and (ii) the third receiver further gets R_2 random linear combinations of W_2. The proof is by induction:

Induction Base

Let $R_1 = 0$. *Algorithm 7* assigns W_2 to all the resources. Thus, receiver 3 gets $r_3 \geq R_1 + R_2$ random linear combinations of W_2 and (i) and (ii) both hold.

Induction Hypothesis

Let $R_1 \leq r$ and assume that the assignment given by *Algorithm 7* satifies (i) and (ii) for any $(R_1, R_2) \in \mathcal{R}_G^0$ and over all combination networks.

Induction Step

Assume $R_1 = r + 1$ and $(R_1, R_2) \in \mathcal{R}_G^0$. Run *Algorithm 7* for one iteration to assign message W_1 on the edge(s) $\mathbf{e}$ that it finds, providing each receiver k, $k = 1, 2, 3$ with $r_k^{\mathbf{e}} \geq 1$ linear combinations of W_1. We show that eliminating these edges leaves us with a combination network G' on which resources could be allocated to $(R_1 - \min_k\{r_k^{\mathbf{e}}\})$ rate of message W_1 and R_2 rate of message W_2. To this end, we show that $(R_1 - \min_k\{r_k^{\mathbf{e}}\}, R_2) \in \mathcal{R}_{G'}^0$, where $R_1 - \min_k\{r_k^{\mathbf{e}}\} \leq r$ and $\mathcal{R}_{G'}^0$ is the capacity region of the new combination network G'. We then apply the induction hypothesis (which states that *Algorithm 7* optimally gives the resource assignment on G' for all rate pairs $(R_1', R_2') \in R_{G'}^0$, $R_1' \leq r$) to conclude the optimality of *Algorithm 7*.

We take into account the following cases as suggested by *Algorithm 7* and find the structure of G' which is formed after the edge elimination depending on the topology of the combination network.

- $|\mathcal{E}_{123}| > 0$. The edge to be marked in this case is an edge of $\mathcal{E}_{123}$. It is easy to see that $\min_k\{r_k^{\mathbf{e}}\} = 1$ and the resulting G' has $r_k' = r_k - 1$, $k = 1, 2, 3$, and $|\mathcal{E}_3'| = |\mathcal{E}_3|$.

- $|\mathcal{E}_{123}| = 0$, and $\min\{|\mathcal{E}_{12}|, \mathcal{E}_{13}|, |\mathcal{E}_{23}|\} > 0$. In this case, one edge from each $\mathcal{E}_{ij}$ is marked. We thus have $\min_k\{r_k^{\mathbf{e}}\} = 2$ and G', depending on R_1, has either $r_k' = r_k - 2$, $k = 1, 2, 3$, and $|\mathcal{E}_3'| = |\mathcal{E}_3|$ (if $R_1 \geq 1$) or $r_1' = r_1 - 2$, $r_2' = r_2 - 1$, $r_3' = r_3 - 1$, and $|\mathcal{E}_3'| = |\mathcal{E}_3|$ (if $R_1 = 1$).

- $|\mathcal{E}_{123}| = \min\{|\mathcal{E}_{12}|, |\mathcal{E}_{13}|, |\mathcal{E}_{23}|\} = 0$, and $|\mathcal{E}_i| \& |\mathcal{E}_{j,l}| > 0$ for some $\{i, j, l\} = \{1, 2, 3\}$. In this case, one edge from $\mathcal{E}_i$ and one edge from $\mathcal{E}_{j,l}$ is marked. So $\min_k\{r_k^{\mathbf{e}}\} = 1$ and G' has the following topological parameters: $r_k' = r_k - 1$, $k = 1, 2, 3$, and either $|\mathcal{E}_3'| = |\mathcal{E}_3|$ (if $i \neq 3$) or $|\mathcal{E}_3'| = |\mathcal{E}_3| - 1$ (if $i = 3$).

- $|\mathcal{E}_{123}| = \min\{|\mathcal{E}_{12}|, |\mathcal{E}_{13}|, |\mathcal{E}_{23}|\} = \min\{|\mathcal{E}_i|, |\mathcal{E}_{j,l}|\} = 0$, $\forall\{i, j, l\} = \{1, 2, 3\}$, and $|\mathcal{E}_1| \& |\mathcal{E}_2| \& |\mathcal{E}_3| > 0$. In this case, one edge from each $\mathcal{E}_i$ is marked. Similarly, $\min_k\{r_k^{\mathbf{e}}\} = 1$ and G' has $r_k' = r_k - 1$, $k = 1, 2, 3$, and $|\mathcal{E}_3'| = |\mathcal{E}_3| - 1$.

- $|\mathcal{E}_{123}| = \min\{|\mathcal{E}_{12}|, |\mathcal{E}_{13}|, |\mathcal{E}_{23}|\} = \min\{|\mathcal{E}_i|, |\mathcal{E}_{j,l}|\} = \min\{|\mathcal{E}_1|, |\mathcal{E}_2|, |\mathcal{E}_3|\} = 0,\ \forall\{i,j,l\} = \{1,2,3\}$, and $|\mathcal{E}_{i,j}|\&|\mathcal{E}_{i,l}| > 0$ for some $\{i,j,l\} = \{1,2,3\}$. In this case, we have one edge from $\mathcal{E}_{ij}$ and one edge from $\mathcal{E}_{il}$ marked. $\min_k\{r_k^{\mathbf{e}}\} = 1$ and G' has $r'_i = r_i - 2,\ r'_j = r_j - 2,\ r'_l = r_l - 2$ and $|\mathcal{E}'_3| = |\mathcal{E}_3|$.

For all those cases with $r'_k = r_k - 1,\ k = 1,2,3$, and $|\mathcal{E}'_3| = |\mathcal{E}_3| - 1$, $\mathcal{R}^0_{G'}$ is characterized by

$$\begin{aligned} R'_1 &\leq \min\{r_1, r_2, r_3\} - 1, && (4.14)\\ R'_1 + R'_2 &\leq r_3 - 1, && (4.15)\\ 2R'_1 + R'_2 &\leq r_1 - 1 + r_2 - 1 + |\mathcal{E}_3|. && (4.16) \end{aligned}$$

Furthermore, in all such cases, $\min_k r_k^{\mathbf{e}} = 1$ and so it's easy to verify that $(R_1 - \min_k r_k^{\mathbf{e}}, R_2) \in \mathcal{R}^0_{G'}$ for all $(R_1 = r + 1, R_2) \in \mathcal{R}^0_G$. The same argument should be made for all the other cases. For the sake of brevity we present here the case where $|\mathcal{E}_{123}| = 0$ and $\min\{|\mathcal{E}_{12}|, \mathcal{E}_{13}|, |\mathcal{E}_{23}|\} > 0$ (which is interestingly the only case where routing is not optimal). We consider two cases: $R_1 \geq 2$ and $R_1 = 1$.

- $R_1 \geq 2$: $\mathcal{R}^0_{G'}$ is characterized by

$$\begin{aligned} R'_1 &\leq \min\{r_1, r_2, r_3\} - 2, && (4.17)\\ R'_1 + R'_2 &\leq r_3 - 2, && (4.18)\\ 2R'_1 + R'_2 &\leq r_1 - 2 + r_2 - 2 + |\mathcal{E}_3|. && (4.19) \end{aligned}$$

  Furthermore, $\min_k r_k^{\mathbf{e}} = 2$. It is immediate to see that $(R_1 - \min_k r_k^{\mathbf{e}}, R_2) \in \mathcal{R}^0_{G'}$ for all $(R_1 = r + 1 > 1, R_2) \in \mathcal{R}^0_G$.

- $R_1 = 1$: $\mathcal{R}^0_{G'}$ is characterized by

$$\begin{aligned} R'_1 &\leq \min\{r_1 - 2, r_2 - 1\}, && (4.20)\\ R'_1 + R'_2 &\leq r_3 - 1, && (4.21)\\ 2R'_1 + R'_2 &\leq r_1 - 2 + r_2 - 1 + |\mathcal{E}_3|. && (4.22) \end{aligned}$$

  Furthermore, $\min_k r_k^{\mathbf{e}} = 1$. We prove by contradiction that for all $(R_1 = 1, R_2) \in \mathcal{R}^0_G$, we have $(R_1 - \min_k r_k^{\mathbf{e}} = 0, R_2) \in \mathcal{R}^0_{G'}$. Assume that $(0, R_2) \notin \mathcal{R}^0_{G'}$ for some R_2 which satisfies $(1, R_2) \in \mathcal{R}^0_G$. Then

$$\min\begin{Bmatrix} r_3 - 1, \\ r_1{+}r_2{+}|\mathcal{E}_3|{-}3 \end{Bmatrix} < \min\begin{Bmatrix} r_3 - 1, \\ r_1{+}r_2{+}|\mathcal{E}_3|{-}2 \end{Bmatrix}. \qquad (4.23)$$

We show in the following that to have (4.23), we should have $r_1 + r_2 - 3 + |\mathcal{E}_3| < r_3 - 1 < r_1 + r_2 - 2 + |\mathcal{E}_3|$ which is a contradiction (for our assumed integer values): The right hand side can be simplified to $r_3 - 1$ and furthermore

$$\begin{aligned} r_3 - 1 &\overset{(1)}{=} |\mathcal{E}_3| + |\mathcal{E}_{13}| + |\mathcal{E}_{23}| - 1 && (4.24)\\ &\leq |\mathcal{E}_3| + |\mathcal{E}_{13}| + |\mathcal{E}_{23}| + |\mathcal{E}_1| + \\ &\quad + |\mathcal{E}_2| + 2(|\mathcal{E}_{12}| - 1) - 1 && (4.25)\\ &\overset{(2)}{=} r_1 - 1 + r_2 - 3 + |\mathcal{E}_3| && (4.26)\\ &< r_1 - 1 + r_2 - 2 + |\mathcal{E}_3|, && (4.27) \end{aligned}$$

where (1) and (2) are both by the assumption of $|\mathcal{E}_{123}| = 0$. The left hand side is thus not equal to $r_3 - 1$, forcing $r_1 - 1 + r_2 - 3 + |\mathcal{E}_3| < r_3 - 1 < r_1 - 1 + r_2 - 2 + |\mathcal{E}_3|$: contradiction. So $(R_1 - \min_k r_k^{\mathbf{e}} = 0, R_2) \in \mathcal{R}^0_{G'}$ for all $(R_1 = 1, R_2) \in \mathcal{R}^0_G$.

The reader is refered to [GSFL10] for the analysis of $(R_1 - \min_k r_k^{\mathbf{e}}, R_2) \in \mathcal{R}^0_{G'}$ in the other cases. □

4.3.2 Algorithm optimality - erasures

In this section we assume an erasure probability $\alpha > 0$ for all the channels of the combination network independently and that messages W_1 and W_2 of rates $(R_1, R_2) \in \mathcal{R}^\alpha_G$ are to be communicated over the combination network. We use a random code of rate $(1 - \alpha)$ and encode the nR_1 symbols of W_1 to $\frac{nR_1}{1-\alpha}$ symbols and similarly W_2 symbols to $\frac{nR_2}{1-\alpha}$ symbols. Linear combinations of encoded W_1 and also of encoded W_2 symbols are now of a rate smaller than $1 - \alpha$ and can thus be communicated with arbitrary small error probability to intermediate nodes. We can thus as before apply *Algorithm 7* with a rate pair $(\frac{nR_1}{1-\alpha}, \frac{nR_2}{1-\alpha})$. We just have to show that the re-constructed messages $\hat{W}_1$ and $\hat{W}_2$ are such that

$$Pr\{\hat{W}_i \neq W_i\} \overset{n\to\infty}{\to} 0. \tag{4.28}$$

Since the receivers are provided with random linear combinations of encoded message W_1 and random linear combinations of encoded message W_2, (4.28) holds if the following two conditions are satisfied with high probability:

- The number of non-erased W_1 carrying signals received at each receiver is greater than or equal to nR_1 with high probability, and

- The number of non-erased W_2 carrying signals received at receiver 3 is greater than or equal to nR_2 with high probability.

Consider the received vector $\bar{Y}_i^n$ at receiver i. By the algorithm analysis in *Section 4.3.1*, we know that each receiver i is connected to at least $\frac{nR_1}{1-\alpha}$ edges which carry linear combinations of the randomly encoded W_1 (with high probability). Pick the set (of cardinality $\frac{nR_1}{1-\alpha}$) of those edges carrying the aforementioned $\frac{nR_1}{1-\alpha}$ linear combinations. By some abuse of notation, call them $Y_1, \cdots, Y_{\frac{nR_1}{1-\alpha}}$. Assign to each Y_k a random variable Z_k defined as

$$Z_k = \begin{cases} 0 & \text{if } Y_k \text{ is erased} \\ 1 & \text{otherwise} \end{cases} . \tag{4.29}$$

Since $Pr\{|\sum_k Z_k - \frac{nR_1}{1-\alpha} \times (1-\alpha)| \geq \epsilon\} \to 0$ when $n \to \infty$, the number of non-erased W_1 carrying signals received at each receiver is greater than or equal to nR_1 with high probability. Using a similar argument for W_2, we conclude the achievability of the rate pair $(R_1, R_2) \in \mathcal{R}_G^\alpha$.

4.4 Algorithm evaluation

In this section we compare the encoding scheme given by *Algorithm 7* described in previous sections, with a network coding-based scheme which we denote by *NCrand*. For simplicity, we assume throughout this section that the erasure probability $\alpha = 0$.

For the *NCrand* scheme, the source has only information about the min-cut of each receiver, and it does not know which edge is available to what receiver. The server uses all the available resources and for each message $W_k, k \in \{1, 2\}$ it randomly allocates a number of edges, proportional to the rate R_k that should be delivered. This means that for any rate pair $(R_1, R_2) \in \mathcal{R}_G^0$, during each time slot, the server selects randomly $\frac{R_1}{R_1+R_2} r_{123}$ edges to send W_1 and $\frac{R_2}{R_1+R_2} r_{123}$ edges to send W_2. Then, each destination i receives $r_i \frac{R_1}{R_1+R_2}$ linear combinations of W_1, but at most R_1 linear combinations are linearly independent. Therefore, the useful rate of W_1 at receiver i with $i \in \{1, 2, 3\}$ is given by:

$$S_{1,i} = R_1 \min\left\{1, \frac{r_i}{R_1 + R_2}\right\}. \tag{4.30}$$

Using a similar argument, the useful rate of message W_2 at the third receiver is equal to:

$$S_{2,3} = R_2 \min\left\{1, \frac{r_3}{R_1 + R_2}\right\} = R_2, \tag{4.31}$$

using inequality (4.5) from the characterization of the rate region. Notice that $S_{2,1}$ and $S_{2,2}$ are not of interest, since only the third receiver should decode message W_2.

Consider we use the network during T time slots. For any rate pair $(R_1, R_2) \in \mathcal{R}_G^0$, the *Algorithm 7* delivers a total rate of $T(R_1 + R_2)$, as in each time slot it is able to assign the resources such that to achieve the desired rate pair. In order to deliver the same total rate with *NCrand*, the server needs T_r time slots, where $T_r = \max\{T_1, T_2\}$. Further, T_1 is the total number of time slots needed to deliver message W_1 to all receivers:

$$T_1 = \max\left\{\frac{TR_1}{S_{1,1}}, \frac{TR_1}{S_{1,2}}, \frac{TR_1}{S_{1,3}}\right\} \tag{4.32}$$

$$= \frac{T}{\min\left\{1, \frac{\min_i r_i}{R_1+R_2}\right\}} \tag{4.33}$$

T_2 is the number of time slots needed to deliver message W_2 to the third receiver:

$$T_2 = \frac{TR_2}{R_2} = T. \tag{4.34}$$

Next, we define the following function to measure the benefit of using *Algorithm 7* over the *NCrand* scheme:

$$f(R_1, R_2) = \frac{T_r}{T} \tag{4.35}$$

$$= \frac{\max\{T_1, T_2\}}{T} \tag{4.36}$$

$$= \max\left\{\frac{1}{\min\left\{1, \frac{\min_i\{r_i\}}{R_1+R_2}\right\}}, 1\right\} \tag{4.37}$$

for any rate pair $(R_1, R_2) \in \mathcal{R}_G^0$. If $f(R_1, R_2)$ takes higher values, this means the time needed to deliver a desired rate is shorter for the scheme proposed by *Algorithm 7* as compared to the *NCrand* approach.

Algorithm 7 provides benefits over the other approach if $f(R_1, R_2) > 1$, which occurs for the case when $R_1 + R_2 > \min_i\{r_i\}$. Note that in this situation,

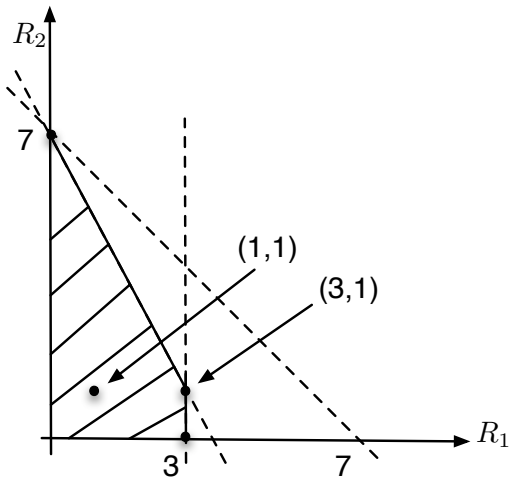


Figure 4.4: Example of the rate region for the combination network from *Figure 4.3*, where $|\mathcal{E}_{13}| = 3$ and $|\mathcal{E}_{23}| = 4$, while the other edge sets $\mathcal{E}_{\{.\}}$ are empty.

the bottleneck is either receiver 1 or receiver 2, since $r_3 \geq R_1 + R_2$ from inequality (4.5) from the rate region. Intuitively, if we consider that receiver 1 has access to fewer resources than the others, with *NCrand* the server may select the resources visible to 1 to carry W_2. Consequently, the leftover edges to which receiver 1 has access, are not enough to deliver message W_1 to him in one time slot. If $R_1 + R_2 \leq \min_i\{r_i\}$, then the *NCrand* delivers the desired rate pair per time slot, as our algorithm, and $f(R_1, R_2) = 1$.

Example

For the topology given in *Figure 4.3*, consider the network parameters are as follows: $|\mathcal{E}_{13}| = 3$ and $|\mathcal{E}_{23}| = 4$, while the other edge sets $\mathcal{E}_{\{.\}}$ have no elements. Then we have that $r_1 = 3$, $r_2 = 4$, and $r_3 = 7$, with the rate region depicted in *Figure 4.4*. In order to deliver rate pair $(R_1, R_2) = (3, 1)$, *Algorithm 7* outperforms the *NCrand* approach by 33%, with $f(3, 1) = \frac{4}{3}$. However, for rate pair $(R_1, R_2) = (1, 1)$, both schemes use the same number of time slots, and $f(1, 1) = 1$.

4.5 Concluding remarks

In this chapter we studied the degraded two message set problem, over a combination network G and in the presence of erasures of rate α. We gave a characterization of the rate region, $\mathcal{R}_G^\alpha$, and introduced an algorithm that achieves it by using topological information. Further we compared our algorithm to an approach oblivious to the network topology that selects the resources at random, and found out that the benefits obtained with the proposed algorithm depends both on the available resources and the rate pair that we want to achieve. In particular, relying on the knowledge about the network topology, the server can deliver messages W_1 and W_2 even at the highest rates from the rate region, using *Algorithm 7*. Without topological knowledge, the server can only achieve low rate pairs.

As future work, we consider extending the algorithm to the case of multicasting to a larger set of receivers and carry on a practical evaluation.

CHAPTER 5

Enabling Live Streaming for Heterogenous Users in Multicast Scenarios

The rapid growth in mobile users over the last years is increasing the demand for services traditionally restricted to wired networks, such as video streaming. While there has been abundant research aiming at ensuring a reasonable quality of video experience to wireless receivers, it is still not clear how to provide quality video streaming to a heterogeneous set of receivers with different subscription levels. Although layered coding [HSLG99] can ensure graceful degradation in the presence of packet losses and differentiated service provision to distinct users, its conjugation with network coding in a wireless setting has not been yet fully explored.

As an example, consider the scenario shown in *Figure 5.1*, in which nodes A, B and C are interested in a video stream served by node S, but they have paid for different video qualities (different layers of a multi-resolution video stream). Node S can connect to the receivers through three relay nodes in wireless range, but with poor channel quality. Due to the unreliability of the wireless medium, S needs to retransmit the lost packets using the feedback received from A, B and C to ensure reliable video transmission. This task becomes even more challenging when feedback messages do not arrive in a reliable manner. Moreover, the relays need to synchronize and schedule transmissions

to ensure every receiver gets all the packets without duplicates. However, layered coding is likely to yield prioritization and scheduling problems in wireless scenarios. For instance, [KHKS04] has shown that even the apparently simple prioritization of the base layer is not a trivial task. Under this scenario, video quality can be decreased, with video frames not delivered in a timely fashion and therefore skipped.

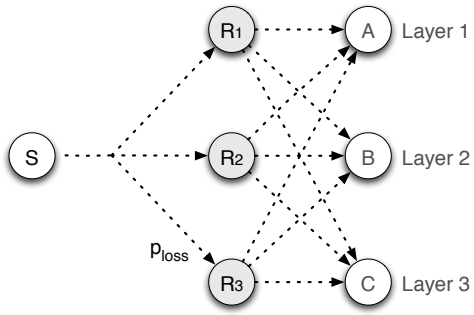


Figure 5.1: A source S streams video to 3 sink nodes A, B and C through relay nodes R_1, R_2 and R_3 in a wireless setting. The probability of dropping a packet in each link (in dashed) is p_{loss}. The sinks subscribed for different video quality, thus one must devise mechanisms to ensure reliable delivery over the wireless medium.

Network coding (i.e. algebraic mixings of packets in a network [ALY00]) is regarded as a promising approach to tackle the problems above. Random Linear Network Coding (RLNC) is a completely distributed fashion to implement network coding protocols. Nodes draw several coefficients at random and use them to form linear combinations of incoming packets [HKM+03]. The resulting packet is sent along with the global encoding vector, which enables the receivers to decode by means of Gaussian elimination. The benefits of network coding for wireless communications have been shown in several works, such as [FKM+07], [KRH+06] and [JLK08]. Network coding can also minimize the decoding delay by using feedback [CMWB08], making it suitable for multimedia streaming [SM07], [SRB05], [FdMC08].

Our main goal is to propose a RLNC-based system architecture for layered video streaming and perform a reality check of its performance in a high-loss scenario with unreliable feedback. The main features of this architecture are credit-based redundancy control and carefully matched video layers and network codes. The decoder uses an adaptive playing policy and feedback is used to limit the number of transmissions at the server. Based on a detailed

simulation study we compare the performance of the proposed scheme with that of a traditional RLNC approach, as well as of an approach without network coding, when the receivers send *realistic feedback*. As a benchmark, we also show the performance of the schemes for the case of *perfect feedback*. For the case of *perfect feedback*, we consider that the control packets are not delayed or lost and the server has perfect knowledge about the state of the buffer at each receiver. The results are valid for a general multicast setting in which several heterogenous devices subscribe to multi-resolution streaming video in a lossy wireless network.

The chapter is organized as follows. In *Section 5.1* we give an overview of the network coding multiresolution scheme under consideration. We discuss some practical aspects of the scheme in *Section 5.2*. In *Section 5.3* we present the simulation settings and the results of our evaluation. We conclude with some final remarks in *Section 5.4*.

This work has been published in [GLL+10] and [LGB+10]. In the latter, a security scheme is considered as well, to ensure that only entitled users (e.g. those who payed for it) decode a subset of the content.

5.1 Network coding for video streaming

We consider a wireless network where the source and the relay nodes only have access to the identifiers of the sinks (e.g. the IP addresses), without centralized knowledge of the network topology or of the encoding functions. We also adopt the model of video layers from [LSP+07b], illustrated in *Figure 5.2*. Video data is divided into groups of pictures (GoPs) with a constant duration of 1 second. In this chapter, we use the terms video segment and GoP interchangeably. The data is then encoded into L layers; for clarity, we consider that each layer is divided into a fixed number of packets, denoted by m. Each layer is dependent on all previous layers (that is, layers $1, \ldots, (l-1)$ are necessary to decode layer l).

We now introduce the proposed scheme and elaborate on its main properties.

5.1.1 Scheme operation

For each GoP, the *source* generates an $n \times n$ lower-triangular matrix $\mathbf{A}$, in which n is the number of packets in the GoP, and $n = Lm$. Matrix $\mathbf{A}$ is used for encoding at the source only, and has the shape shown in *Figure 5.3*. Each

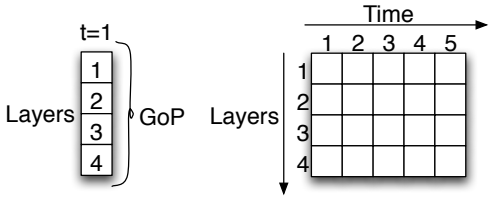


Figure 5.2: Layer model. The video data is divided into groups of pictures (GoP) with the duration of 1 second. GoPs are then subdivided into layers.

non-zero entry of $\mathbf{A}$ is an element a_{ij} chosen uniformly at random from all non-zero elements of the field $\mathbb{F}_q \backslash \{0\}$.

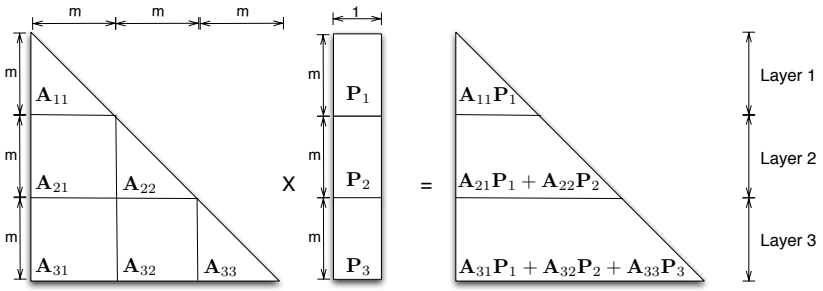


Figure 5.3: The coding operations: using a triangular coefficient matrix yields a nested structure for the layers.

The GoP is then divided into vectors $\mathbf{P}_l$ with $l = 1, \cdots L$, where each vector contains the m packets that belong to layer l. The coded packets are composed by the header (which includes the coefficient vector) and the payload. The payload is obtained by multiplying matrix $\mathbf{A}$ with the vector $\mathbf{P} = [\mathbf{P}_1\ \mathbf{P}_2\ \cdots\ \mathbf{P}_L]^T$. Note that, because of the nested structure of coding determined by the triangular matrix, each packet of layer l includes packets from layers $l, l-1, \ldots 1$. Moreover, when encoding one packet of layer l with a packet of layer $j > l$, the resulting packet belongs to layer j.

The *relays* first identify the layer of a packet by looking at the corresponding coefficient vector, as follows. Let i be the highest index of a non-zero element in the coefficient vector, then the layer to which the packet belongs is determined as: $l = \lceil \frac{i}{m} \rceil$. The relays further encode packets according to the rules of standard RLNC protocols, by mixing a received linear combination with packets of the same or lower layers only.

The *receivers* apply Gaussian elimination following standard RLNC in order

to decode the native packets.

Example

In the following we give an example on how to use this coding scheme. We assume there are $L = 3$ layers in our system, and each layer contains $m = 3$ packets. Then the sender generates the lower triangular coefficient matrix $\mathbf{A}$ by randomly selecting the coefficients from a finite field, as shown below:

$$\mathbf{A} = \left(\begin{array}{c|c|c} \mathbf{A}_{11} & & \\ \hline \mathbf{A}_{21} & \mathbf{A}_{22} & \\ \hline \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{array}\right)$$

$$= \left(\begin{array}{ccc|ccc|ccc} a_{11} & 0 & 0 & & & & & & \\ a_{21} & a_{22} & 0 & & & & & & \\ a_{31} & a_{32} & a_{33} & & & & & & \\ \hline a_{41} & a_{42} & a_{43} & a_{44} & 0 & 0 & & & \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & 0 & & & \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} & & & \\ \hline a_{71} & a_{72} & a_{73} & a_{74} & a_{75} & a_{76} & a_{77} & 0 & 0 \\ a_{81} & a_{82} & a_{83} & a_{84} & a_{85} & a_{86} & a_{87} & a_{88} & 0 \\ a_{91} & a_{92} & a_{93} & a_{94} & a_{95} & a_{96} & a_{97} & a_{98} & a_{99} \end{array}\right), \quad (5.1)$$

where we only marked the blocks shown in *Figure 5.3* – $\mathbf{A}_{11}$, $\mathbf{A}_{21}$, $\mathbf{A}_{22}$, $\mathbf{A}_{31}$, $\mathbf{A}_{32}$, $\mathbf{A}_{33}$, while the other blocks have all elements equal to zero. Let the vector of packets corresponding to one GoP be given by:

$$\mathbf{P} = \left(\begin{array}{c} \mathbf{P}_1 \\ \hline \mathbf{P}_2 \\ \hline \mathbf{P}_3 \end{array}\right) = \left(\begin{array}{c} p_1 \\ p_2 \\ p_3 \\ \hline p_4 \\ p_5 \\ p_6 \\ \hline p_7 \\ p_8 \\ p_9 \end{array}\right), \quad (5.2)$$

where $\mathbf{P}_l$ contains the packets that belong to layer $l, 1 \leq l \leq L$. Then, the source S will encode the GoP by multiplying the vector of packets $\mathbf{P}$ with $\mathbf{A}$ as follows:

$$\mathbf{A} \times \mathbf{P} = \left(\begin{array}{c} \mathbf{P}_1^c \\ \hline \mathbf{P}_2^c \\ \hline \mathbf{P}_3^c \end{array}\right) = \left(\begin{array}{c} p_1^c \\ p_2^c \\ p_3^c \\ \hline p_4^c \\ p_5^c \\ p_6^c \\ \hline p_7^c \\ p_8^c \\ p_9^c \end{array}\right), \tag{5.3}$$

where by $\mathbf{P}_l^c$ we denote the vector of coded packets corresponding to layer $l, 1 \leq l \leq L$, and by p_i^c we denote the i-th linear combination generated by the node. Replacing equations (5.1) and (5.2) in the above relation, we get that:

$$\begin{aligned} \mathbf{A} \times \mathbf{P} &= \left(\begin{array}{c} \mathbf{A}_{11}\mathbf{P}_1 \\ \hline \mathbf{A}_{21}\mathbf{P}_1 + \mathbf{A}_{22}\mathbf{P}_2 \\ \hline \mathbf{A}_{31}\mathbf{P}_1 + \mathbf{A}_{32}\mathbf{P}_2 + \mathbf{A}_{33}\mathbf{P}_3 \end{array}\right) \\ &= \left(\begin{array}{l} a_{11}p_1 \\ a_{21}p_1 + a_{22}p_2 \\ a_{31}p_1 + a_{32}p_2 + a_{33}p_3 \\ \hline a_{41}p_1 + a_{42}p_2 + a_{43}p_3 + a_{44}p_4 \\ a_{51}p_1 + a_{52}p_2 + a_{53}p_3 + a_{54}p_4 + a_{55}p_5 \\ a_{61}p_1 + a_{62}p_2 + a_{63}p_3 + a_{64}p_4 + a_{65}p_5 + a_{66}p_6 \\ \hline a_{71}p_1 + a_{72}p_2 + a_{73}p_3 + a_{74}p_4 + a_{75}p_5 + a_{76}p_6 + a_{77}p_7 \\ a_{81}p_1 + a_{82}p_2 + a_{83}p_3 + a_{84}p_4 + a_{85}p_5 + a_{86}p_6 + a_{87}p_7 + a_{88}p_8 \\ a_{91}p_1 + a_{92}p_2 + a_{93}p_3 + a_{94}p_4 + a_{95}p_5 + a_{96}p_6 + a_{97}p_7 + a_{98}p_8 + a_{99}p_9 \end{array}\right) \end{aligned} \tag{5.4}$$

Note that the coded packets corresponding for example to layer 2 (i.e. p_4^c, p_5^c, p_6^c), are those corresponding to lines 4, 5 and 6 of the resulting vector in (6.2).

Next, assume a relay receives one coded packet for layer 1, and one coded packet for layer 2. Let these two packets be p_2^c and p_4^c:

$$\begin{aligned} p_2^c &= a_{21}p_1 + a_{22}p_2 \\ p_4^c &= a_{41}p_1 + a_{42}p_2 + a_{43}p_3 + a_{44}p_4. \end{aligned}$$

In order to send a linear combination for layer 2, the relay multiplies each of the coded packets by a random coefficient, b_{11} and b_{12} respectively, adds them together and forwards the resulting coded packet q_1^c, as shown below:

$$\begin{aligned} q_1^c &= b_{11}p_2^c + b_{12}p_4^c \\ &= (b_{11}a_{21} + b_{12}a_{41})p_1 + (b_{11}a_{22} + b_{12}a_{42})p_2 + b_{12}a_{43}p_3 + b_{12}a_{44}p_4. \end{aligned}$$

This last equation illustrates two essential aspects of RLNC: i) intermediate nodes can generate new coded packets without the need to perform decoding operations, and ii) a linear combination of coded packets is in fact a linear combination of the original uncoded packets (i.e. q_1^c is a linear combination of native packets p_1, p_2, p_3 and p_4).

5.1.2 Scheme properties

Note that traditional RLNC [HKM+03, GHK+07] mixes all packets by using a full square matrix. This, however, is not suitable for layered coding, since it is not possible to extract individual layers unless one matrix is used for each layer. The triangular matrix coding effectively mixes the layers, allowing for differentiated recovery of successive layers by nodes with different access levels, while relying on the dissemination of lower-level packets to achieve the resilience necessary for higher-level packets to be delivered in a timely fashion. Moreover, the triangular matrix form provides priority to the base layer, as all upper layer packets contain the base layer. Thus, the common prioritization and scheduling of lower layers is solved in a natural way. In *Section 5.3* we compare this scheme with traditional RLNC addressing scheduling and prioritization issues.

The choice of a triangular matrix further meets an important requirement: it allows us to remove the arbitrary *coding delay* introduced by the typical RLNC with square matrices at the source, since the source can code packets as soon as they are generated and does not have to wait for the end of the generation to send them. Note that if the triangular structure of the matrix is preserved through the network, the sinks are able to decode on-the-fly by means of forward substitution (if there are no losses in the network). Although an alternative strategy – *online network coding* [SSM09a] – has been shown to provide the same advantage, it requires full feedback at intermediate nodes. Furthermore, it is not clear how to combine it with layered coding without adding to the scheduling and synchronization problems.

5.2 System setup

This section introduces the practical scheme that is used for evaluation in *Section 5.3*. The system architecture is shown in *Figure 5.4*. We now discuss each

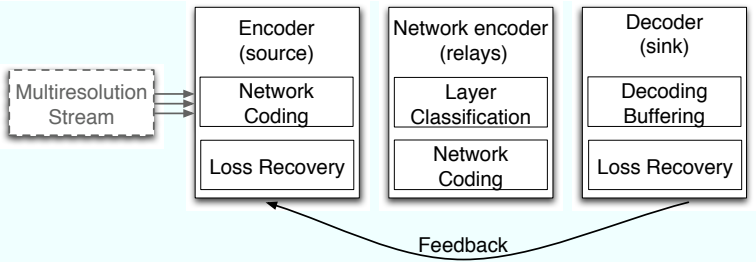


Figure 5.4: Modules of a potential system implementation. Generation of a multiresolution stream is in dashed since it is an external entity to the evaluated system.

of its components.

5.2.1 Source encoder

The *source encoder* includes the *network coding* and the *loss recovery* modules.

For the network coding operations, the source proceeds as explained in *Section 5.1.1*. Furthermore, an important aspect of the encoder is to control the rate at which intermediate nodes generate and send linear combinations to the receivers. If a relay generates and forwards a linear combination every time he receives an innovative packet from the server, then many redundant packets may arrive at the destinations. To solve this issue, the server generates one or more credits for each coded packet, which are further assigned to one or more of the intermediate relays. Next, only the relays who receive also a credit associated with the packet are allowed to send a linear combination.

We borrow the concept of credits from [GHK+07], but we extend it to take into account the case of delivery to *multiple receivers*. The idea is to enable a sufficient number of the downstream neighbors to forward linear combinations such that every coded packet transmitted by the source results in an increase in the degree of freedom at each receiver. Assume that the source S is delivering layer l of one GoP, and let the subset of users subscribed to decoding layer l be denoted by $\mathcal{R}_l$. Further, let the set of downstream neighbors be denoted by $\mathcal{N}$. Then, each neighbor $n_i \in \mathcal{N}$ is assigned a weight w_i directly proportional to the number of receivers it can reach, and the list $\mathcal{N}$ is ordered after decreasing values of this weight. Next, the source starts to assign credits to the neighbors from the beginning of the list, until all receivers are reachable through the selected forwarders. The process, also explained in *Algorithm 10*, is carried on in the same way at intermediate relays.

Algorithm 10: **Forwarder selection for multicast scenarios.** When forwarding a coded packet for layer l, a node X selects as many forwarders as needed, such that the degree of freedom is increased at all receivers of layer l.

Input: layer l; subset of users subscribed to this layer, $\mathcal{R}_l$; the set of downstream neighbors of node X, $\mathcal{N}$

foreach $n_i \in \mathcal{N}$ **do**
  $w_i \leftarrow$ #users n_i can reach
end
order $\mathcal{N}$ by decreasing values of w_i ;
$idx \leftarrow 0$;
while $\mathcal{R}_l \neq \emptyset$ **do**
  assign a credit to n_{idx} ;
  remove from $\mathcal{R}_l$ the users that are reachable through n_{idx} ;
  $idx \leftarrow idx + 1$;
end

Let us take an example to better understand the forwarder selection for multicast scenarios. Consider the topology in *Figure 5.5*, where the links in dashed indicate which transmissions can be heard by each of the nodes. When the source S sends a coded packet for layer 1, since all receivers should decode layer 1, then S will assign credits to both R_1 – which can reach A and B, and R_3 – which can reach C. However, for a layer 3 packet, the source will assign only one credit to neighbor R_3, thus enabling only R_3 to generate and forward a linear combination.

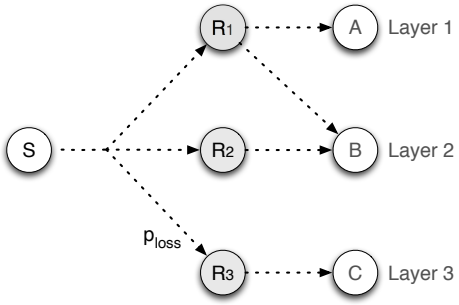


Figure 5.5: The source S streams video to 3 heterogenous sink nodes A, B and C through relay nodes R_1, R_2 and R_3 in a wireless setting. Links in dashed show which nodes are within transmission range.

Note that this approach is not energy-efficient, in the sense that it does not

try to minimize the number of credits assigned to the downstream neighbors for each coded packet – thus it does not minimize the number of transmissions at the next hops. However, since the scenario that we focus on refers to *live streaming*, we strive to achieve a *reliable and timely delivery of the content to all receivers*.

Since the transmissions occur over the volatile wireless environment, the *loss recovery* module is instrumental in order to provide quality video streaming. Thus, after transmitting a complete generation, and before streaming the next one, the server starts the loss recovery process in a feed-forward manner. To recover lost packets, the server sends redundant linear combinations for each layer, mixing all the packets of that layer. This process continues until all the receivers subscribed to that layer can decode the generation.

5.2.2 Network (relay) encoder

The *network encoder* is a component of the wireless relays of the network and includes the *layer classification* and the *network coding* modules. As mentioned in *Section 5.1*, the relays first classify a newly received linear combination as belonging to a layer l. After classifying the packet, a relay generates and forwards a linear combination if he received a credit assigned to that packet. A packet of layer l should only be combined with packets of the same or lower layers, i.e. $l, l-1, \ldots, 1$. This is done in order to maintain the diversity of layers in the network, because when combining a packet of layer l with one belonging to layer $l+1$, the layer of the resulting packet is $l+1$.

5.2.3 Decoder

The *decoder* is a component of the receiver that includes *decoding and buffering* and *feedback*. When enough packets are received, the receiver performs Gaussian elimination to decode the packets.

Since, in the presented scheme, relay nodes perform coding on the packets of the same (and lower) layers, the shape of the triangular matrix sent by the source is not kept through the network. Thus, a received packet, even if innovative in terms of rank, might not be decodable immediately and should be stored at a decoding buffer at the receiver. This decoding buffer takes into account the maximum allowable delay of the video stream, similar to the play buffer at the receivers, and will preemptively flush the current undecoded packets if the delay requirement is not met. Once a full layer is decoded, it is stored in the playback buffer.

A node starts the playback once it decodes a number of segments in the lowest quality. At timestep k the node plays segment k in the quality in which it is available. If the segment was not decoded (not even in the lowest quality), then the node stops the playback process and starts buffering. If after some buffering timeout, the node decodes segment k, then it plays it in the quality in which it is available; otherwise, the node skips segment k and plays the next one.

5.2.4 Feedback mechanism

We consider a system with minimal feedback, in order to free the wireless channels from unnecessary transmissions. The receivers send positive feedback to the server whenever they decode a segment in the desired quality. For example, a layer 3 receiver sends a unique feedback packet when it has decoded layers 1, 2 and 3.

When the server receives a feedback packet from a layer l receiver for segment k, it updates the information for the loss recovery process as follows. If all the receivers of layer l have decoded, then the server will send redundant packets for layer $l+1$, provided that l is not the highest layer, i.e. $l < L$, where L denotes the number of layers in the system. If l is the highest layer in the system, i.e. $l = L$, then the server moves with the error recovery at the next segment, that is it will send redundant packets for layer 1 of segment $k+1$.

Finally, the feedback packets are protected against loss in the unreliable medium by *hop-by-hop retransmissions*. Whenever a node N_1 receives a feedback packet from node N_2, N_1 sends an *ack* to N_2. If the *ack* does not arrive within a timeout τ, then N_2 retransmits the feedback packet.

5.3 Evaluation

We evaluate the performance of the protocol described in *Section 5.2* via simulation in the multi-hop multi-path scenario from *Figure 5.1*, in which the server S sends video to 3 heterogenous receivers A, B and C, through relays R_1, R_2 and R_3, over wireless links. In this section we will show the performance of the scheme in terms of throughput and robustness to losses, and its ability to deliver quality video to a heterogeneous set of receivers, when the feedback from the receivers is *realistic*, i.e. it is sent over unreliable links.

We compare the layered network coding model (*scheme NC1*) with standard RLNC (*scheme NC2*) and an implementation without network coding (*scheme WoNC*). In *scheme NC2* the server sends a different stream for every layer.

Each segment is encoded in different qualities, using a full coefficient matrix for each layer. Relay nodes perform RLNC operations on the received packets that belong to the same generation and to the same or lower layers. In this case, since a sink of layer L needs to receive a full-rank matrix for layers $1, 2, \ldots L$, sinks acknowledge each layer that they decode. Error recovery is similar to that for *scheme NC1*, described in *Section 5.2.1*. In *scheme WoNC*, the server sends the native packets without coding them and the intermediate nodes just forward uncoded packets normally. The sinks send as feedback the *ids* of the packets they received. If some packets are lost, the server retransmits them.

As a benchmark, we also show the performance of the schemes for the case of *perfect feedback*. For the *perfect feedback* case, we consider the control packets are not delayed or lost and the server has perfect knowledge about the state of the buffer at each receiver.

5.3.1 Simulation setup

We use the network simulator *ns-2.33* [ns2], with the default random number generator for this version. The network coding libraries are independently programmed. The video stream is a constant bit rate traffic over UDP, where the server is streaming at 480 kbps during 100 seconds. Each layer has a fixed size of $m = 20$ packets and we consider $L = 3$ layers for the system, which yields a generation of 60 packets, corresponding to 1 second of video. The packet size is 1000 bytes. As a propagation model, we use *two-ray ground* and we consider the loss probability p_{loss} as a simulation parameter. Since it was shown that RTS/CTS has a negative impact on the performance, we disable it for all experiments. In order to simulate heavy loss conditions, we also disable MAC layer retransmissions. The rate at the MAC layer is 11 Mbps.

The receivers start to playback the video stream once they have decoded at least 5 segments of the lowest quality. The buffering timeout for a segment that has not been decoded until its playback deadline arrives is set to 1 second. We fit the timeout for feedback retransmissions τ through experiments and set it to 200 ms. In order to take full advantage of the broadcast nature of the wireless medium, the relays listen to transmitted packets in promiscuous mode.

We consider the following metrics:

- the *rate played* at the receivers;
- the *load on the server*, defined as the ratio between the total rate sent by the server and the streaming rate;

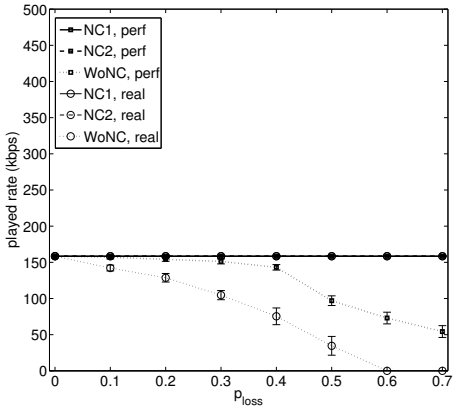
- the *decoding delay*, expressed as the time elapsed from receiving the first packet of a segment until that segment is decoded;
- the percentage of *segments skipped* at playback;
- the percentage of segments played in *lower quality* than the one requested;
- the *initial buffering delay*, defined as the time interval from receiving the first packet to the beginning of the playback;
- the *playback quality*, defined as the average quality in which each segment is played.

In all plots, each point is the average of 10 runs and the vertical lines show the standard deviation. When the behavior is very similar for all 3 layers, we only show the plot for layer 3. The behavior for layers 1 and 2 is always slightly better, since layer 3 receivers need to receive more packets than lower layer nodes.

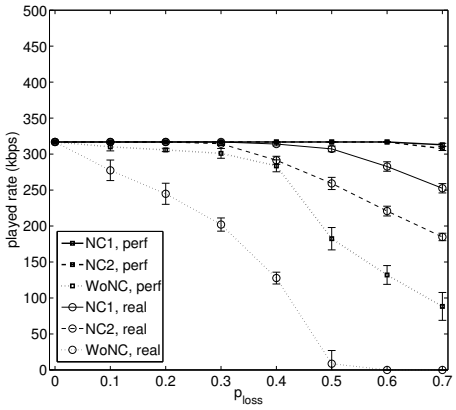
5.3.2 Results

Figure 5.6 shows the rate played by each receiver vs. loss probability. When *feedback is perfect*, *Scheme NC1* and *Scheme NC2* achieve the maximum played rate for each layer due to the inherent reliability of network coding, even if the probability of loss increases. The performance of *Scheme WoNC* decreases as p_{loss} increases because the receivers send less feedback and the server does fewer retransmissions, thus the lost packets are not recovered. For the case of *realistic feedback*, the best performance is achieved by *Scheme NC1*. *Scheme NC2* is more affected by losses because it needs more feedback. As explained above, each receiver sends a control packet when it decodes each layer allowing the server to progress with the loss recovery process, thus this scheme is more sensitive to feedback packets being lost. *Scheme WoNC* performs even worse in this case, because as the loss increases fewer data packets that trigger the feedback mechanism arrive at destinations. Next, these feedback packets may not reach the server due to the unreliable medium, and the server does not retransmit lost packets.

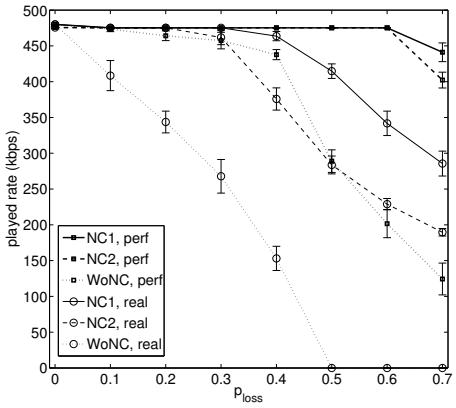
We can see in *Figure 5.7* that the load on the server grows exponentially as the loss probability increases (for the case of *perfect feedback*). The network coding schemes must send less coded packets to recover losses, because a linear combination may recover different losses at different receivers, while for *Scheme WoNC* the server needs to retransmit exact packets. The exponential



(a) Layer 1



(b) Layer 2



(c) Layer 3

Figure 5.6: Played rate in function of loss probability p_{loss}, for the scheme described in *Section 5.1* (*NC1*), three streams with network coding (*NC2*) and without network coding (*WoNC*). The circles on the curves denote the case of *realistic feedback* and the squares denote the case of *perfect feedback*.

behavior of the load on server is similar for *realistic feedback* for *Scheme NC1* and *Scheme NC2*. Note that with *Scheme NC2* the server sends more packets than with *Scheme NC1* due to its sensitivity to lost feedback. Consider for example that all receivers decoded layer 1, but the server received only the feedback from nodes A and B, then the server will continue to send redundant packets for layer 1 that are not needed anymore. For *scheme WoNC* the load is

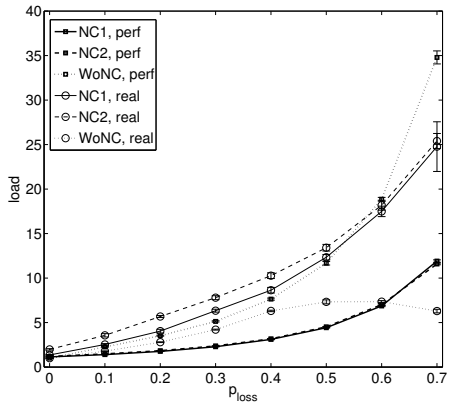


Figure 5.7: The load on the server in function of the loss probability p_{loss}.

significantly lower because the server retransmits packets only when it receives feedback from the receivers. Since fewer data packets reach the destinations, then fewer feedback packets need to be sent (which at their turn may be lost in the channel). Thus, in *scheme WoNC* the server sends less packets overall.

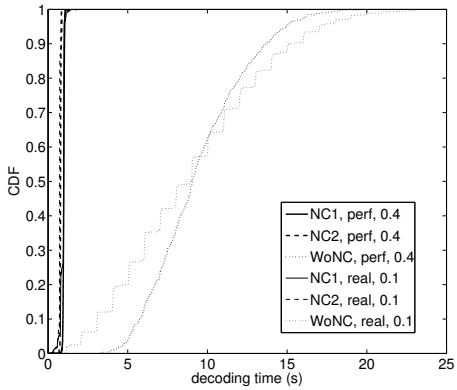


Figure 5.8: CDF of decoding delay for layer 3, for *perfect feedback* and loss probability $p_{loss} = 0.4$, and for *realistic feedback* and $p_{loss} = 0.1$.

Figure 5.8 shows that the network coding approaches are able to decode segments within a second, for both *perfect* and *realistic feedback*, as the server sends redundant linear combinations in a feed-forward manner. *Scheme WoNC* needs a longer decoding time, because the server waits for the feedback before retransmitting. Note that for the *real* case, the decoding delay evolves in a

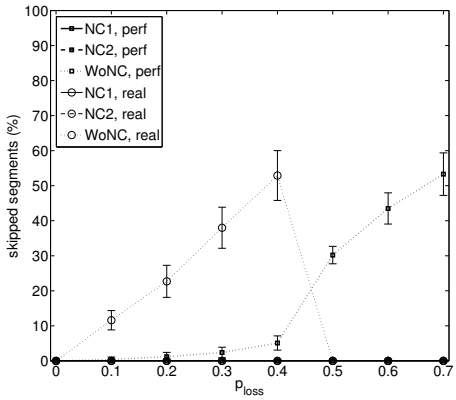


Figure 5.9: The percentage of skipped segments with the probability of loss, p_{loss}, for layer 3, for *perfect feedback* and for *realistic feedback*.

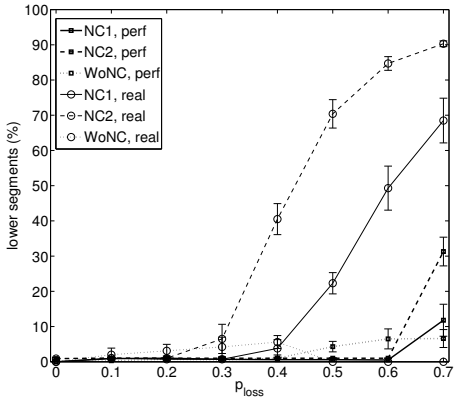


Figure 5.10: The percentage of segments played in lower quality in function of the probability of loss p_{loss} for layer 3, for *perfect feedback* and for *realistic feedback*.

ladder, due to the fact that feedback packets are lost and retransmitted hop-by-hop.

Figure 5.9 and *Figure 5.10* show the percentage of segments that are skipped and played in lower quality, respectively, for layer 3. The results for the other layers are similar and we omit them due to lack of space. Note that with network coding, no segments are skipped for any layers, neither for *perfect feedback*, nor for *realistic feedback*. For *perfect feedback* with *Scheme*

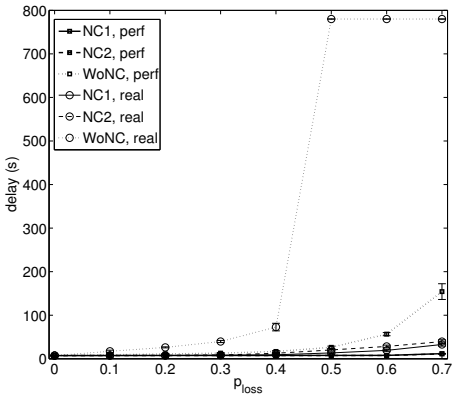


Figure 5.11: Initial buffering delay with the loss probability p_{loss}, for layer 3.

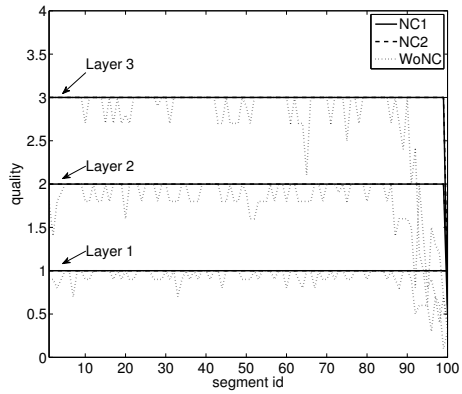


Figure 5.12: Played quality for the case of *perfect feedback* and $p_{loss} = 0.4$.

WoNC the receiver starts to skip segments as the p_{loss} increases beyond 0.4 because it is not able to decode in a timely manner. For *realistic feedback* the percentage of skips increases with the probability of loss up to the point of $p_{loss} = 0.5$ where the receiver is unable to decode and play anything (see also *Figure 5.6c*). Consequently it does not skip any segment either. Note that no segments are played in lower quality with *Scheme NC1* and *Scheme NC2* in the case of *perfect feedback*. However, for *realistic feedback* the percentage of lower quality segments increases with the probability of loss. With *Scheme WoNC* the receiver plays very few packets in lower quality because the packets retransmitted by the server do not arrive in due time, thus most of them are

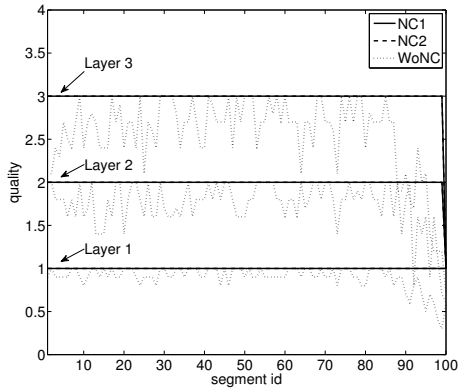


Figure 5.13: Played quality for the case of *realistic feedback* and $p_{loss} = 0.1$.

skipped.

We can see in *Figure 5.11* that for *perfect* and *realistic feedback*, the receivers buffer a shorter time before starting the playback for both *Scheme NC1* and *Scheme NC2*. The initial buffering delay grows slowly with the probability of loss, because a single network coded packet can recover multiple losses. For *scheme WoNC* with *realistic feedback*, when losses are high the receivers are not able to decode anything, thus they never start to play the file.

Figure 5.12 and *Figure 5.13* show the average quality in which every segment is played, for the case of *perfect feedback* with $p_{loss} = 0.4$, and the case of *realistic feedback* with $p_{loss} = 0.1$, respectively. A skipped segment accounts as played in a quality equal to 0. Note that both network coding approaches show a high resilience to errors and the video file is constantly played in the desired quality by each receiver, for both types of *feedback*, *perfect* or *realistic*. On the other hand, with *Scheme WoNC* the played quality is lower because the lost packets are not recovered in a timely manner. For the *realistic feedback* case, the control packets that are lost further deteriorate the played quality, such that for $p_{loss} = 0.1$, the played quality is similar to the one obtained for *perfect feedback* at $p_{loss} = 0.4$.

5.4 Concluding remarks

We evaluated the performance of RLNC for layered video coding, by designing and implementing a specific system solution for a lossy wireless scenario. Our results show that network coding approaches perform better in high loss sce-

narios and mixing several layers yields higher performance gains. In particular, by generating the encoding matrix at the source in a specific shape and allowing intermediate nodes to perform the traditional network coding operations yet still prioritizing base layers of the video, we are able to achieve gains in buffering delay, percentage of skipped segments and variability of the quality played at the sinks (even if the feedback is not perfect). The implementation of such an architecture in a real scenario was shown to be feasible.

As part of our future work, we are considering the comparison of the RLNC approach with online network coding schemes, as well as selective discarding of video frames.

Part III

Multipoint-to-Multipoint Communications

CHAPTER 6

Providing Incentives to Heterogenous Users for Live Streaming

With the popularization of multimedia services in dynamic networks of heterogeneous devices, the need arises for reliable schemes for distributing data among users. Peer-to-peer (P2P) networks are becoming an increasingly popular solution for alleviating the load of servers by taking advantage of the clients' resources. These peers share their bandwidth, storage and processing capabilities with other peers in the network. However, in existing P2P systems peers are not incentivized to share their resources in the system. Promoting incentives is not a trivial task, since users are typically interested in downloading data, and not contributing bandwidth to the common good. This situation is further aggravated when not all users are interested in the same data, but in a subset of the source content.

In fact, the propagation of devices with heterogeneous capabilities regarding display graphics, processing power capabilities and bandwidth, calls for new solutions to assure fairness and quality of service for all peers in the network simultaneously. To ensure graceful degradation in the presence of packet losses and differentiated service provision to distinct users, typical video codecs, such as the MPEG family, adopt a multi-resolution source coding approach to generate a scalable video stream with multiple layers. The quality of experience

for a user basically depends on the number of layers it is capable of recovering [TF00]. Although these differentiated quality levels can be used as an incentive for sharing the streaming data [LSP+07b] (that is, nodes are interested in uploading to their peers in order to receive the maximum layer possible and obtain higher quality video), this strategy can be harder to implement in a network where certain users cannot take advantage of the increased quality that the scalable video can provide. Furthermore, in layered coding, layers must be received in order to be decoded sucessfully; prioritizing lower layers of the video and simultaneously promoting incentives can introduce non-trivial scheduling and synchronizing problems in a P2P network [LSP+07b]. Even the use of multiple description codes, in which video quality is proportional to the number of descriptions received, can yield scheduling problems at the receivers [LSP+07a]. In addition, traditional incentive mechanisms relying on tit-for-tat strategies, do not work for streaming systems, in that not all users can perform a direct trade. Intuitively, a user who plays the video at a point behind the playback point of his neighbors, does not have pieces of information that could be of interest for them.

Network coding (i.e. algebraic mixing of packets in the intermediate nodes of a network) has been recently shown to provide an elegant solution to scheduling problems, as well as reducing the number of control messages exchanged in the network [WL07b, FL08]. Random Linear Network Coding (RLNC) can be implemented in a distributed fashion, which makes it particularly suitable as a framework for dynamic and unstable networks, such as delay tolerant networks [WL05] and content distribution networks [GR05, DGWR07]. In particular, its operation is completely desynchronized and local, since each node forwards random linear combinations independently of the information present at other nodes. Additionally, when collecting random combinations of packets, there is a high probability of getting a linearly independent packet, and thus, the problem of redundancy caused by traditional flooding approaches is diluted, since there is no need to download one particular fragment; instead, any linearly independent fragment is likely to bring innovative information [DGWR07].

Reference [CWCC08] uses network coding to reduce network traffic and server load by leveraging on end host's buffer space, however in a tree-based system. The work in [WL07a] shows that network coding reduces the bandwidth usage in the network. Reference [LPDG06] includes a scheme for peer-to-peer live streaming with network coding, where the nodes exchange their coding vectors in order to determine exactly how much innovative data they can send to a neighboring peer. A different approach is used in [AGG+07],

where network coding is used judiciously to solve scheduling problems within segments of a video-on-demand protocol. Finally, a more general approach is proposed in [TF07], where raptor codes for video streaming are re-encoded at intermediate nodes to take advantage of path diversity in the overlay network.

Motivated by these observations, we take a different look at the benefits that network coding could provide to a live streaming system. In particular, we set out to develop a network coding based scheme for layered P2P live video streaming, that also efficiently creates an incentive scheme for fostering cooperation among users. Our main contributions are:

- We propose a network coding based live video streaming scheme, which accounts for incentives to sharing and heterogeneous users in the network through layered video.
- Under this setting, through the use of network coding, we are able to implement an incentive strategy with minimum scheduling and overhead. Users receive a number of layers proportional to the ones they share.
- Through extensive simulations, we show that our scheme based on a particular network coding technique naturally accounts for several common problems in P2P layered video streaming systems, such as the complex scheduling needed for ordering packets and synchronizing between serving peers and the extra coding that is needed in order to prioritize the base layer of the video.

The remainder of the chapter is organized as follows. First, *Section 6.1* describes the model we consider for our system. *Section 6.2* provides a general overview of the proposed system. We present the proof-of-concept, performance evaluation, system considerations and simulation results in *Section 6.3*. The chapter concludes with some final remarks in *Section 6.4*.

This work has been accepted for publication [GLLB11].

6.1 System model

In order to generically evaluate the performance of our coding and incentive system, we consider a live streaming setting in a network containing of three main entities: the *server*, the *tracker* and the *peers*. Our setting is represented in *Figure 6.1*, and consists of an underlay and an overlay network. In the underlay network, the server connects to a node representing the Internet, which is then connected to each of the peers. In the overlay network, the peers

connect among themselves to share data with the goal of alleviating the load on the server. We do not consider delays caused by the representation of the Internet, as well as by multiple-hop issues. For the purpose of our analysis, we assume that the functionalities of the tracker are carried on by the server, as well. We refer to the nodes that are connected to each other in the overlay network as *neighbors*.

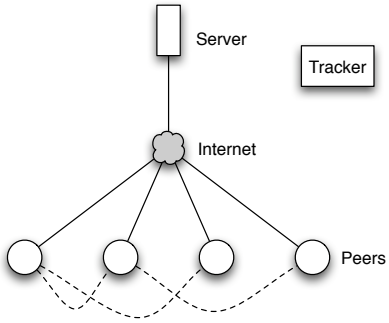


Figure 6.1: Network model. The connections for the underlay network are represented in full, while the links for the overlay network (logical connections) are represented in dashed.

We consider the video file is encoded using layered coding, and thus, there are several levels of playback quality. Regarding video layers, we adopt the model from [LSP+07b], which we also used in *Chapter 5* and here we give a brief overview. The video data is divided into groups of pictures (GoPs) with a constant duration of 1 second. This data is then encoded into L layers; each layer is divided into a certain number of packets, denoted by m, for coding with network coding. An illustration of this setting is given in *Figure 5.2*. We consider one segment to contain one GoP. Each segment is encoded in one network coding generation, so we use the terms GoP, segment and generation interchangeably in the paper.

We adopt the network coding model from [HKM+03], whereby nodes draw several coefficients at random from a finite field and use them to form linear combinations of incoming packets. The resulting coded packets are sent along with the global encoding vector, which is the set of linear transformations that the original packets go through on the path through the network. These encoding vectors are grouped in a coefficient matrix, which enables the receivers to decode by means of Gaussian elimination.

In order to accommodate layered coding, we make the encoding matrix

block lower triangular, as illustrated in *Figure 6.2*. Blocks $\mathbf{A}_{ij}$ with $i, j \in \{1, 2, 3\}$ and $j \leq i$ represent coefficient matrices, with randomly selected elements from a finite field. The blocks that are not marked have all elements equal to 0. Vectors $\mathbf{P}_l$ contain the packets belonging to layer l. First, the server encodes the packets from the base layer, followed by the packets belonging to the upper layers, in order of refinement level. Thus, in our system, a packet of layer l mixes packets from layers $l, l-1, \ldots, 1$.

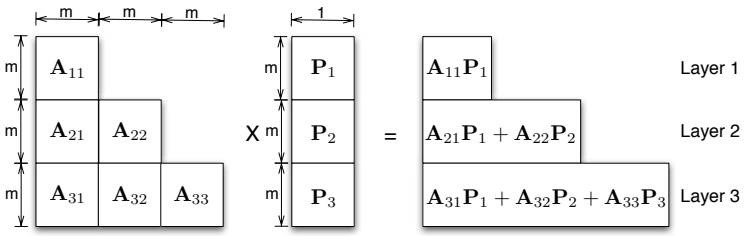


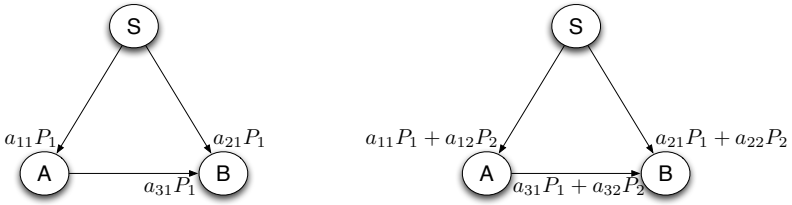
Figure 6.2: Block triangular matrix, which provides a nested structure for coding layers. A packet of layer l mixes packets from layers $l, l-1, \ldots, 1$.

Note that the choice of a block lower triangular matrix as an ecoding matrix allows us to automatically give priority to the base layer: in fact, all upper layer packets contain the base layer. Thus, a common problem with using layers in P2P networks – prioritizing the base layer – is solved in a natural way by using network coding with this specific form of the encoding matrix.

The reader may wonder why we did not use lower triangular matrices for encoding, as we used in *Chapter 5*. The key difference is the fact that in *Chapter 5* we are addressing a *point-to-multipoint scenario* – with a server delivering content to all receivers, whereas in this chapter we focus on a *multipoint-to-multipoint scenario* – with users contributing to each other in order to deliver the content. In the later case, extra care must be taken in order to ensure that the packets exchanged by the users are *useful* to each other. In other words, the packets should be *innovative* to avoid wasting network resources by sending redundant packets, and to enable the implementation of an efficient incentive system, as we will discuss in the next sections. In the multicast scenario from *Chapter 5*, we do not face this issue, as the server is always able to send innovative coded packets of the source content to the receivers that have not decoded yet. Therefore, for the P2P case, we pay special attention to redundancy problems at the cost of higher delay (i.e. the source waits longer, to receive all packets from a layer before starting to upload linear combinations).

In the following, let us take a simple example to illustrate the redundancy problem for P2P scenarios. Consider a setting with a server S and two peers A and B, connected in the overlay network. Further assume that there are $m = 2$ packets in a layer, and the upload rate at the server is higher than the streaming rate, that is, coded packets are uploaded at a faster pace than that at which native packets are produced.

With triangular coefficient matrices, the server S would start to upload linear combinations as soon as the first packet is available. Thus, it may upload a linear combination $a_{11}P_1$ to receiver A, and another linear combination $a_{21}P_1$ to receiver B. Further, one of the two receivers, say A, may as well upload a linear combination of the available information to its neighbor B, as shown in *Figure 6.3a*. However, the coded packet sent by A, $a_{31}P_1$ does not bring any innovative information to B and it will be discarded.



(a) Triangular matrix

(b) Block triangular matrix

Figure 6.3: Simple setting to illustrate the redundancy problem. With triangular coefficient matrices, peer B receives one innovative packet and one redundant packet. Using block triangular coefficient matrices, peer B receives two innovative packets.

If block triangular matrices are used, then the server would start to encode native packets only after all the packets from the layer are available. Therefore, the server would upload a linear combination $a_{11}P_1 + a_{12}P_2$ to node A, and another linear combination $a_{21}P_1 + a_{22}P_2$ to node B, as illustrated in *Figure 6.3b*. Next, A may upload a coded packet $a_{31}P_1 + a_{32}P_2$ to B, thus node B receives two innovative packets.

Example

We now give an example on how to use this coding scheme, as opposed to the one based on triangular coefficient matrices described in *Chapter 5*. We assume again that there are $L = 3$ layers in our system, and each layer contains

$m = 3$ packets. Then the sender generates the block lower triangular coefficient matrix $\mathbf{A}$ by randomly selecting the coefficients from a finite field, as shown below:

$$\mathbf{A} = \left(\begin{array}{c|c|c} \mathbf{A}_{11} & & \\ \hline \mathbf{A}_{21} & \mathbf{A}_{22} & \\ \hline \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{array}\right)$$

$$= \left(\begin{array}{ccc|ccc|ccc} a_{11} & a_{12} & a_{13} & & & & & & \\ a_{21} & a_{22} & a_{23} & & & & & & \\ a_{31} & a_{32} & a_{33} & & & & & & \\ \hline a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & & & \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} & & & \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} & & & \\ \hline a_{71} & a_{72} & a_{73} & a_{74} & a_{75} & a_{76} & a_{77} & a_{78} & a_{79} \\ a_{81} & a_{82} & a_{83} & a_{84} & a_{85} & a_{86} & a_{87} & a_{88} & a_{89} \\ a_{91} & a_{92} & a_{93} & a_{94} & a_{95} & a_{96} & a_{97} & a_{98} & a_{99} \end{array}\right), \quad (6.1)$$

where we only marked the blocks shown in *Figure 6.2* – $\mathbf{A}_{11}$, $\mathbf{A}_{21}$, $\mathbf{A}_{22}$, $\mathbf{A}_{31}$, $\mathbf{A}_{32}$, $\mathbf{A}_{33}$, while the other blocks have all elements equal to zero. Contrary to the scheme in *Chapter 5* where blocks $\mathbf{A}_{ii}, 1 \leq i \leq 3$ are lower triangular, in this case blocks $\mathbf{A}_{ii}$ are full matrices, with non-zero elements.

Let the vector of packets corresponding to one GoP be given by equation (5.2), and let the server encode the GoP according to equation (5.3), then by replacing relation (6.1) in the latter, we obtain that:

$$\mathbf{A} \times \mathbf{P} = \left(\begin{array}{c} \mathbf{A}_{11}\mathbf{P}_1 \\ \hline \mathbf{A}_{21}\mathbf{P}_1 + \mathbf{A}_{22}\mathbf{P}_2 \\ \hline \mathbf{A}_{31}\mathbf{P}_1 + \mathbf{A}_{32}\mathbf{P}_2 + \mathbf{A}_{33}\mathbf{P}_3 \end{array}\right)$$

$$= \left(\begin{array}{l} a_{11}p_1 + a_{12}p_2 + a_{13}p_3 \\ a_{21}p_1 + a_{22}p_2 + a_{23}p_3 \\ a_{31}p_1 + a_{32}p_2 + a_{33}p_3 \\ \hline a_{41}p_1 + a_{42}p_2 + a_{43}p_3 + a_{44}p_4 + a_{45}p_5 + a_{46}p_6 \\ a_{51}p_1 + a_{52}p_2 + a_{53}p_3 + a_{54}p_4 + a_{55}p_5 + a_{56}p_6 \\ a_{61}p_1 + a_{62}p_2 + a_{63}p_3 + a_{64}p_4 + a_{65}p_5 + a_{66}p_6 \\ \hline a_{71}p_1 + a_{72}p_2 + a_{73}p_3 + a_{74}p_4 + a_{75}p_5 + a_{76}p_6 + a_{77}p_7 + a_{78}p_8 + a_{79}p_9 \\ a_{81}p_1 + a_{82}p_2 + a_{83}p_3 + a_{84}p_4 + a_{85}p_5 + a_{86}p_6 + a_{87}p_7 + a_{88}p_8 + a_{89}p_9 \\ a_{91}p_1 + a_{92}p_2 + a_{93}p_3 + a_{94}p_4 + a_{95}p_5 + a_{96}p_6 + a_{97}p_7 + a_{98}p_8 + a_{99}p_9 \end{array}\right) \quad (6.2)$$

Note that the coded packets corresponding for example to layer 2 (i.e. p_4^c, p_5^c, p_6^c), are those corresponding to lines 4, 5 and 6 of the resulting vector in (6.2).

Next, assume a relay receives one coded packet for layer 1, and one coded packet for layer 2. As in the example from previous chapter, let these two packets be p_2^c and p_4^c, which in this case are given by:

$$\begin{aligned} p_2^c &= a_{21}p_1 + a_{22}p_2 + a_{23}p_3 \\ p_4^c &= a_{41}p_1 + a_{42}p_2 + a_{43}p_3 + a_{44}p_4 + a_{45}p_5 + a_{46}p_6. \end{aligned}$$

In order to send a linear combination for layer 2, the relay multiplies each of the coded packets by a random coefficient, b_{11} and b_{12} respectively, adds them together and forwards the resulting coded packet q_1^c, as shown below:

$$\begin{aligned} q_1^c &= b_{11}p_2^c + b_{12}p_4^c \\ &= (b_{11}a_{21} + b_{12}a_{41})p_1 + (b_{11}a_{22} + b_{12}a_{42})p_2 + (b_{11}a_{23} + b_{12}a_{43})p_3 \\ &+ b_{12}a_{44}p_4 + b_{12}a_{45}p_5 + b_{12}a_{46}p_6. \end{aligned}$$

6.2 Architecture and system aspects

We start by giving a general overview of our system. As mentioned above, we aim at providing a distributed way for P2P mesh networks to achieve (a) streaming video quality with heterogeneous users, (b) incentive mechanisms and (c) resilience through the use of network coding in conjunction with layered streaming.

6.2.1 State information and protocol messages

Every peer in the network keeps the following state information:

- *list of neighbors*, together with statistics about the total number of packets received from and sent to each of them;
- *segment availability*, which refers to the list of nodes who have packets belonging to a segment;
- *list of data requests*, stored and served in the order of reception;
- *coding buffer*, which contains the segments to serve to its peers;

- *decoding buffer*, which contains the segments it has received, but are not decoded yet;

- *playback buffer*, which contains decoded segments that have not been played yet.

From the information mentioned above, the server keeps only the *list of data requests* and the *coding buffer*. Moreover, it keeps a list with the receivers, mapped by the layer they are interested in decoding.

Our system relies on the exchange of the following messages:

Peer Request When a node joins the network, it sends a *Peer Request* to the server, which also includes the layer that the peer is interested in decoding. A node re-requests more peers when it remains behind the streaming point by more than a specified number of segments.

Peer List After receiving a *Peer Request*, the server adds the sender to its list of receivers and answers back with a *Peer List* message, which contains a list of nodes, prioritized by their layers, and the *id* of the segment that is currently being streamed.

Data Request Once a node receives a *Peer List* from the server, it updates the *list of neighbors* and it sends a *Data Request* packet to a subset of its neighbors. A node re-requests data periodically with some timeout, τ. When receiving a *Data Request*, a peer adds it at the end of the list of requests and removes any older request from the same neighbor.

Data Packet Whenever a node uploads a packet, it selects one of the queueing *data requests*, and serves a packet for the requested segment.

Have Data A node informs his neighbors that it has data from a segment by sending *Have Data* messages. This control message also includes the service time, i.e. the time interval a new request would stay in the queue, before being served. A peer A computes the service time as:

$$t_s^A = \frac{\sum_i n_{k,l}^i}{u_A}, \tag{6.3}$$

where i iterates through all the requests currently queued by the node. u_A represents the upload rate at node A in packets per second, and $n^i_{k,l}$ represents the number of packets that node i requested from A for layer l of segment k.

The component blocks of our system are: *requesting peers*, *requesting data*, *serving data*, and *playing data*. Since we discussed the *requesting peers* mechanism above with the *Peer Request* message, in the following sections we only focus on the other blocks.

6.2.2 Requesting data

In order to request data packets for segment k, a node A first selects a subset of its neighbors, taking into account the *segment availability*. In particular, from the neighbors who announced having segment k, the node picks r neighbors at random, with a probability inversely proportional to the service time advertised by each neighbor. This means that a neighbor who has already received a high number of requests and has its upload capacity saturated will be less likely to receive more requests, as any new request will be served after all those that are already in the queue. If none of the neighbors have information about segment k, then node A directs its request to the server.

Each node requests one segment at a time. A *Data Request* includes the following information:

- the *id* of the node sending the request, i;
- the *id* of the segment being requested, k;
- the *layer* that the node wants to decode, l;
- the *number of linear combinations* that the node needs in order to decode the specified layer, $n^i_{k,l}$.

If after some timeout τ the layer has not been decoded, the node issues another request, by repeating the same process.

6.2.3 Serving data

We first introduce some notation, as follows. The number of packets sent by a node A to a node B for layer l of segment k is represented by $s_{k,l}(A \rightarrow B)$. $N(A \rightarrow B)$ denotes the total number of innovative packets that node A sent to node B. $r_{k,l}(A)$ gives the rank of the coefficient matrix for layer l of segment k at node A.

A node B processes the *data requests* according to First Come First Serve (FCFS) policy. Initially, all peers enjoy a free ride period, which means that node B will upload to a neighbor A a number of packets, for free. After that free limit has been reached, node B uploads to node A a linear combination for layer l of segment k only if the following two conditions are met simultaneously:

$$\frac{N(A \rightarrow B)}{N(B \rightarrow A)} \geq T_u \tag{6.4}$$

$$s_{k,l}(B \rightarrow A) \leq \min(r_{k,l}(B), n^A_{k,l}) \tag{6.5}$$

Equation (6.4) accounts for the incentives in our system and it means that B uploads to A only if the ratio of the contribution of A to him to the contribution that B made to A is higher than a specified threshold, T_u. For example, if $T_u = 1$ (which would be the case of tit-for-tat), B will upload a packet to A only if A sent him at least as many packets as B sent to A. Equation (6.5) ensures that B uploads to A innovative coded packets. Obviously, B can send only as many innovative packets as he himself has in its *coding buffer* for a specified layer of a segment, which yields $s_{k,l}(B \rightarrow A) \leq r_{k,l}(B)$. Moreover, B should upload to A only as many packets as A requested, hence $s_{k,l}(B \rightarrow A) \leq n^A_{k,l}$.

Note that the server uploads packets for the received requests in the FCFS order, regardless of the condition on incentives (6.4) or the condition on innovative data (6.5), as it can always provide innovative packets to the nodes that have not decoded yet a particular segment.

6.2.4 Playing data

Peers try to decode segments as new packets arrive to the *decoding buffer*. Once a full layer is decoded, it is stored in the *playback buffer*. A node starts to play the video file once it decodes a number of segments in the lowest quality and a minimum time has passed since the node started to receive packets. Imposing a minimum limit on the initial buffering time is instrumental to ensure a smooth playback rate. In particular, a node may receive the first packets very fast, however the rate that he decodes may lower in time, as more peers join the network or the incentive mechanism is enforced.

At timestep k the node plays segment k in the quality in which it is available. If the segment was not decoded (not even in the lowest quality), then the node stops the playback process and starts buffering. If after some buffering timeout, the node decodes segment k, then it plays it in the quality in which it is available; otherwise, the node skips segment k and plays the next one. For the

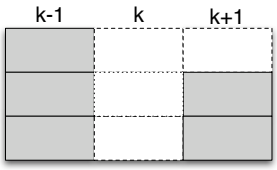


Figure 6.4: Playback buffer: layers in solid have been decoded, while layers in dashed still need to be decoded.

example in *Figure 6.4*, segment $k-1$ is played in the highest quality, segment k is skipped – if no layer is decoded not even after the buffering timeout, and segment $k+1$ is played in a lower quality.

6.3 Performance evaluation

We evaluate the performance of our system in several scenarios of practical interest, such as heterogenous users and homogenous users in the network. We use the *ns-2.33 network simulator* [ns2], with the default random number generator for this version. The network coding libraries are independently programmed.

We consider a P2P network of 200 peers, with all users present in the network from the moment the server starts streaming the file. The video stream is a constant bit rate traffic over UDP, where the server is streaming at 480 kbps during 100 seconds. Each layer has a fixed size of $m = 20$ packets and we consider 3 layers for the system, which yields a generation of 60 packets, corresponding to 1 second of video. The packet size is 1000 bytes. The server sends a list of 30 neighbors when answering a *Peer Request*. Every $\tau = 0.4$ seconds, a node sends *Data Requests* to a subset of $r = 10$ of his neighbors. The receivers start to playback the video stream once they have decoded at least 5 segments. The buffering timeout for a segment that has not been decoded until its playback deadline arrives is set to 1 second. The contribution threshold T_u is set to 1, which means a node A uploads to a neighbor B only if the rate A sent to B is equal to the rate A received from B. A node re-requests more peers when it remains behind the streaming point by more than 5 segments. A summary of the simulation parameters is given in *Table 6.1*.

We keep track of the following metrics:

- the *rate played* at each user;

Table 6.1: Simulation parameters

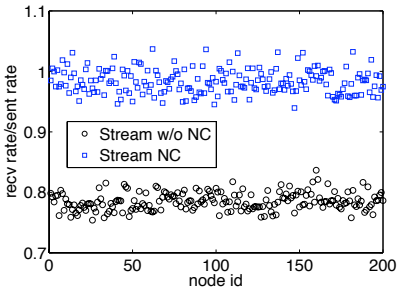
R	Streaming rate; fixed at 480 kbps
N	Number of peers in the network; fixed at 200
n_p	Number of peers sent by the server in a *Peer List*; fixed at 30 peers
r	Number of requests a peer sends at a time; fixed at 10 requests
τ	Request timeout; fixed at 0.4 s
n_d	Number of decoded segments before the playback starts; fixed at 5 segments
d	Buffering delay for a missing segment; fixed at 1 second
T_u	Contribution threshold (incentives); fixed at 1
L	The number of layers in our system; fixed at 3, unless otherwise stated
m	The number of packets in a layer; fixed at 20

- the percentage of *segments skipped* at playback;
- the percentage of *segments played in a lower quality* than the one desired by the user;
- the *buffering delay*, which includes the initial buffering delay (before the playback starts) and the time a node waits to decode missing segments;
- the *decoding delay* for a segment k, defined as the time elapsed since a node receives the first piece of information regarding segment k, until the node is able to decode the segment;
- the *fairness index*, defined as the ratio of the innovative rate received to the rate sent by each user.

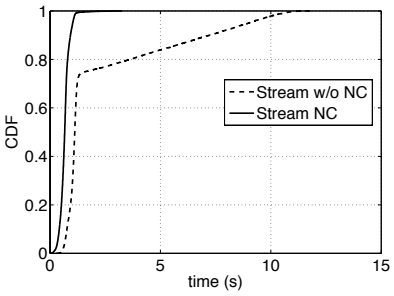
We denote our protocol described in previous sections by *Stream NC*, and compare it with an approach where users request specific, uncoded packets, and send *Have* messages to announce the available segments. To the best of our knowledge, this approach (that we further denote by *Stream w/o NC*) is similar to *PPLive*. However, for the sake of our comparison, we also implemented incentives and differentiated levels of video quality for *Stream w/o NC*. For all simulations, we assume each node has perfect knowledge regarding the contents of the buffer at his neighbors.

6.3.1 Homogenous users

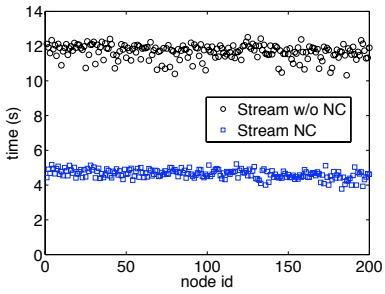
For the case of *homogenous users*, we consider there is only one layer of video quality, and all users have the same upload capacity, equal to 2x the streaming rate.



(a) Fairness index



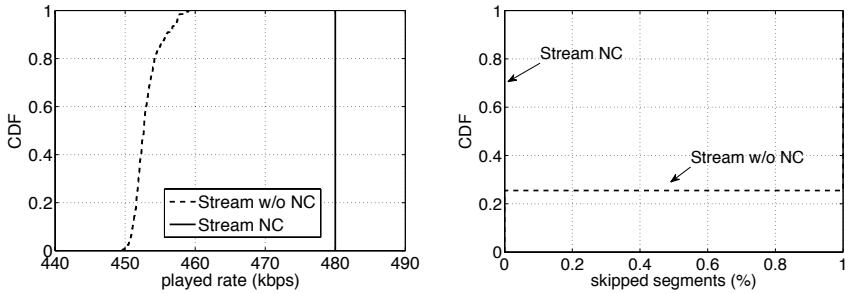
(b) Decoding delay



(c) Buffer delay

Figure 6.5: Homogenous users – delay and incentives

Note from *Figure 6.5a* that with *Stream NC* the fairness index is close to 1, which means the nodes contribute to the system the same rate as the one they receive. Intuitively, using network coding a node can easily upload an innovative packet to another peer. With *Stream w/o NC*, a node A may upload duplicate packets to a peer B, thus the rate that node B needs to "pay back" to A is lower. Overall, the useful rate that a node receives from its peers equals about 0.8 the rate that he contributes to the system. Using network coding, a node is able to decode segments faster, as it only needs to receive a number of independent linear combinations. As shown in *Figure 6.5b*, 99% of the segments across all users are decoded within 1.2 seconds. Without network coding, even if the peers know the exact contents of the neighbors' buffers,



(a) Played rate (b) Skipped segments

Figure 6.6: Homogenous users – playback quality

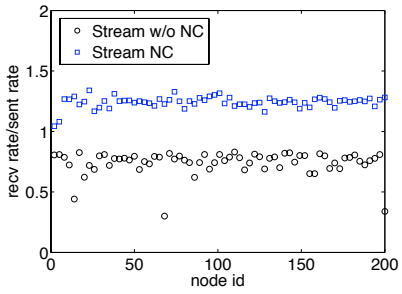
there still can happen that two different peers coincide in uploading the same packet to a node. Therefore a node needs to wait longer to receive specific packets, which yields a decoding delay of 1.2 seconds for 75% of the segments, and up to 12 seconds for the rest of 25% of the segments.

Moreover, the short decoding delay enables the nodes to start the playback after only 5 seconds with *Stream NC* and play the file smoothly, without interruptions. On the other hand, with *Stream w/o NC* the nodes not only wait longer to start playing, but also buffer for segments that are not decoded before their playback deadline, which yields a total buffering delay 2x higher than with *Stream NC*, as shown in *Figure 6.5c*. Note from *Figure 6.6* that with our scheme the nodes are able to play the file at the same rate as the server is streaming, without skipping any segment, while with *Stream w/o NC*, around 70% of the users skip 1% of the segments, hence the average played rate is lower.

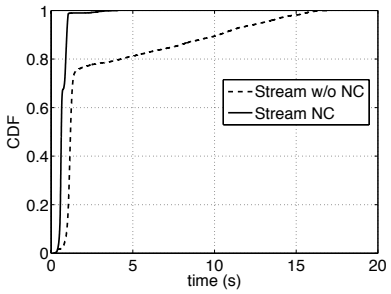
6.3.2 Heterogenous users

For the case of *heterogenous users*, there are $L = 3$ layers of video quality, with one third of users being interested in decoding each layer. The upload rate is set to 2x the rate each user is interested in decoding. Since the behavior is very similar for all 3 layers, we only show the plots for layer 3.

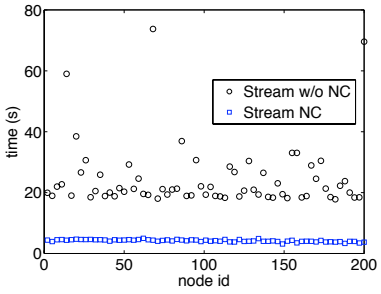
Notice from *Figure 6.7a* that with our scheme, the fairness index is around 1.2, which means the users of layer 3 receive from the system a rate higher than their contribution. This effect is due to the fact that users interested in layer 1 quality are able to decode the segments faster, and consequently they are more involved in distributing layer 1 further to their peers. Hence, the upload



(a) Fairness index



(b) Decoding delay



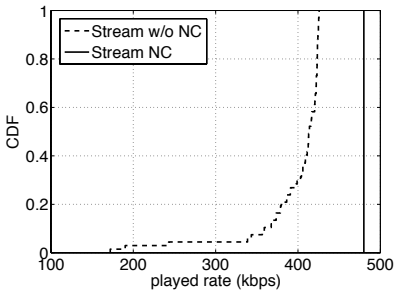
(c) Buffer delay

Figure 6.7: Heterogenous users, layer 3 – delay and incentives

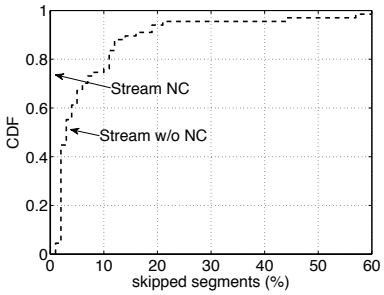
capacity for layer 3 receivers is not saturated, as they receive fewer requests to serve. With *Stream w/o NC* this effect is not observed, because the nodes need to receive particular packets, and even if they require different levels of quality, the receivers of lower layers do not decode significantly faster.

As seen for *homogenous users*, in this case the nodes as well decode the segments faster with *Stream NC*. In particular, 98% of the segments are decoded within 1.2 seconds, which allows the users to start playing the file within around 5 seconds after the server started to stream the video. If network coding is not employed, then only 70% of the segments are decoded within 1.4 seconds, as shown in *Figure 6.7b*. Furthermore, due to the incentive conditions, which limit the neighbors' contribution, the nodes are not able to decode the segments in a timely manner with *Stream w/o NC*, thus the total buffering delay is about 4x higher than in the case of *Stream NC* (*Figure 6.7c*).

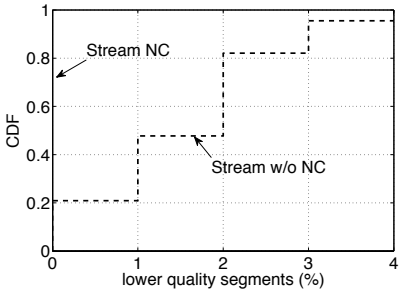
Note from *Figure 6.8* that with *Stream NC* the users are able to play the whole file, at a rate equal to the streaming rate, without any segments being skipped or played in a lower quality. On the other hand, with *Stream w/o NC*



(a) Played rate



(b) Skipped segments



(c) Lower quality segments

Figure 6.8: Heterogenous users, layer 3 – playback quality

only 70% play a rate higher than 400 kbps. Using network coding, the peers can contribute innovative linear combinations to each other, while if peers upload exact packets, this task is more challenging. In other words, a peer that is badly connected, in the sense that it plays the file behind the playback point of his neighbors, will not be able to provide useful packets to his neighbors, and consequently his neighbors will not upload to him once the incentive condition is not fulfilled. Due to these situations, 75% of the users skip 10% of the segments (*Figure 6.8b*), while 50% of peers play 2% of the segments in lower quality (*Figure 6.8c*).

6.4 Concluding remarks

In this paper we introduced a network coding-based solution for live streaming in P2P networks. Our solution relies on the inherent characteristics of network coding in order to provide incentives for collaboration between peers, while ensuring a node's contribution to the system is matched with the played quality

of the video file. We evaluate our scheme through simulations, for the case of both homogenous and heterogenous users present in the network. The results show that the proposed solution achieves a shorter buffer delay and a smooth playback rate, while also ensuring fairness, as opposed to a live streaming scheme similar to PPLive.

As future work, we will focus on evaluating the resilience of our protocol to network dynamics issues, such as flash crowds, and peer churn. We will consider the case when the bandwidth supply at the peers does not meet the demanded rate.

CHAPTER 7

Conclusions

Motivated by existing work showing network coding as a promising approach to have an impact on network design and architecture, in this thesis, we set out to push the state-of-the-art forward and to identify the benefits that network coding could bring by incorporating it at various levels of the protocol stack, for different communication paradigms.

In the area of *point-to-point communications* we introduced a multipath scheme [GLR10, GLR11] that relies on network coding to solve scheduling problems and provide reliability to TCP sessions over wireless links. Our scheme performs error control, by estimating the links' quality online, and congestion control, by using backpressure. Compared to similar approaches from the literature, our protocol is more efficient and yields higher throughput, even in highly volatile environments.

For *point-to-multipoint communications* we presented a scheme to assign the available resources to optimally achieve all the rate pairs from the rate region of the degraded multicast – where users require different subsets of the source content – of a two message set problem [GSFL10, GSFL11]. Relying on topological information, our scheme uses simple coding operations at a subset of the nodes from the newtork. We showed that if the sender does not have complete knowledge about the underlying topology, then only low rate pairs can be achieved.

Degraded multicasting is motivated by scenarios such as streaming to users with differentiated service subscription. In this area, we proposed a system architecture for video streaming to heterogenous users, in a wireless multicast

setting [GLL+10, LGB+10]. Our scheme matches layered video with a specific network coding technique, to elegantly solve the problem of scheduling and prioritizing the base layer. We showed that our system achieves gains in terms of buffering delay, and quality played at the sinks, even if the feedback received by the sender is limited and unreliable.

In the area of *multipoint-to-multipoint communications* we presented a system to provide incentives in a peer-to-peer setting, where users have heterogenous demands [GLLB11]. Our solution relies on layered coding to account for varied quality requirements, and on the inherent characteristics of network coding in order to foster collaboration between peers. The evaluation showed that our system ensures fairness, i.e. a node's contribution to the system is matched with the played quality of the video file. In addition, even if strict delay constraints need to be satisfied, our scheme can achieve an efficient tit-for-tat exchange, if nodes have perfect knowledge on the decoding buffer at their peers.

The results that we presented in this thesis show that network coding enables a reliable transport of data, even over lossy wireless links, or in scenarios with strict delay requirements such as live streaming, and facilitates the implementation of an efficient incentive system, even in the presence of heterogenous users. Thus, network coding can solve the challenges faced by next generation networks in order to support advanced information transport.

7.1 Future work

In this section we point out open research directions, corresponding to each chapter, that stem out from this thesis.

- for *point-to-point communications* (*Chapter 3*), can one design a coding scheme that fully takes advantage of the inherent properties of random linear network coding, and at the same time keeps the coding complexity low (i.e. the number of coding operations performed in the network is kept at a minimum)?

- for *point-to-multipoint communications*, the case of degraded multicasting (*Chapter 4*), what would be the cost of obtaining the information about the underlying toplogy? What would be the tradeoff between the granularity of the topological information and the performance of the achievability scheme for a random topology? Moreover, how would the rate region change if more receivers were present in the network and/or

more messages should be delivered? In addition, how can one achieve the rate region for that case?

- for *point-to-multipoint communications*, the case of live streaming to heterogenous users in a wireless multicast setting (*Chapter 5*), what would be the implications of trying to schedule packet transmissions such that to minimize the energy consumption? What would be the tradeoff between the quality of service (i.e. buffering delay, playback skips) and the energy savings, if an energy-efficient approach would be used? How would the proposed scheme behave over random topologies?

- for *multipoint-to-multipoint communications* (*Chapter 6*), how would the proposed incentive scheme behave in the case of user dynamics (i.e. peers joining and leaving the network at random)? How should the scheme be extended to account for the distribution of multiple video files? Can we devise an analytical framework to model the proposed system? What would be the tradeoff between the incentive threshold and the quality of service (i.e. buffering delay, playback skips) for the realistic case when users have unreliable information on the state of the decoding buffer at their peers?

APPENDIX A

List of Publications

- **A Layered Network Coding Solution for Incentives in Peer-to-Peer Live Streaming**, ***Steluta Gheorghiu***, Luisa Lima, Alberto Lopez Toledo, Joao Barros. IEEE International Symposium on Network Coding (NetCod) 2011

- **A Network Coding Scheme for Seamless Interaction with TCP**, ***Steluta Gheorghiu***, Alberto Lopez Toledo, Pablo Rodriguez. IEEE International Symposium on Network Coding (NetCod) 2011 (poster)

- **Degraded Multicasting with Network Coding over the Combination Network**, ***Steluta Gheorghiu***, Shirin Saeedi Bidokhti, Christina Fragouli, Alberto Lopez Toledo. IEEE International Symposium on Network Coding (NetCod) 2011; also as Technical Report EPFL-REPORT-152016, EPFL, September 2010

- **On the Performance of Network Coding in Multi-Resolution Wireless Video Streaming**, ***Steluta Gheorghiu***, Luisa Lima, Alberto Lopez Toledo, Joao Barros, Muriel Medard. IEEE International Symposium on Network Coding (NetCod) 2010

- **Secure Network Coding for Multi-Resolution Wireless Video Streaming**, Luisa Lima, ***Steluta Gheorghiu***, Joao Barros, Muriel Medard, Alberto Lopez Toledo. IEEE Journal on Selected Areas in Communications, Wireless Video Transmission, 2010, vol. 28

- **Multipath TCP with Network Coding for Wireless Mesh Networks**, ***Steluta Gheorghiu***, Alberto Lopez Toledo, Pablo Rodriguez. IEEE International Conference on Communications (ICC) 2010, WNS
- **Multipath Code Casting for Wireless Mesh Networks**, Christos Gkantsidis, Wenjun Hu, Bozidar Radunovic, Peter Key, Pablo Rodriguez, ***Steluta Gheorghiu***. ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT) 2007; also as Technical Report MSR-TR-2007-67, Microsoft Research, June 2007
- **Analytical Evaluation of the Overhead Generated by a Routing Scheme with Subnets for MANETs**, Johann Lopez, ***Steluta Gheorghiu***, Jose M. Barcelo. EuroNGI Workshop 2006: 126-143
- **TrafficNet: A L2 Network Architecture for Road-to-Vehicle Communication**, David Fuste-Vilella, Jose-Miguel Pulido, Jorge Garcia-Vidal, ***Steluta Gheorghiu***. EuroNGI Workshop 2006: 43-61

Bibliography

[AGG+07] Siddhartha Annapureddy, Saikat Guha, Christos Gkantsidis, Dinan Gunawardena, and Pablo Rodriguez Rodriguez. Is High-Quality VOD Feasible Using P2P Swarming? In *Proceedings of the 16th international conference on World Wide Web - WWW '07*, New York, New York, USA, May 2007. ACM Press. 37, 116

[ALY00] R. Ahlswede, S.-Y.R. Li, and R.W. Yeung. Network Information Flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, July 2000. 22, 29, 42, 94

[ATW02] John G. Apostolopoulos, Wai-tian Tan, and Susie J. Wee. Performance of a Multiple Description Streaming Media Content Delivery Network. In *IEEE International Conference on Image Processing*, pages 189–192. IEEE, 2002. 36

[AW05] I.F. Akyildiz and X. Wang. A Survey on Wireless Mesh Networks. *IEEE Communications Magazine*, 43(9):S23–S30, September 2005. 41

[Ber74] P. Bergmans. A Simple Converse for Broadcast Channels with Additive White Gaussian Noise. *IEEE Transactions on Information Theory*, 20(2):279–280, March 1974. 35, 76

[Bit] BitTorrent. In *http://www.bittorrent.com/*. 21

[BM05] Sanjit Biswas and Robert Morris. ExOR: Opportunistic Multi-Hop Routing for Wireless Networks. In *SIGCOMM*, volume 35, page 133. ACM, October 2005. 32

[BPSK97] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan, and Randy H. Katz. A Comparison of Mechanisms for Improving

TCP Performance over Wireless Links. *IEEE/ACM Transactions on Networking*, 5(6):756–769, 1997. 41

[Cet07] C. Cetinkaya. Improving the Efficiency of Multipath Traffic via Opportunistic Traffic Scheduling. *Computer Networks*, 51(8):2181–2197, June 2007. 41

[CHG03] J. Chakareski, S. Han, and B. Girod. Layered Coding vs. Multiple Descriptions for Video Streaming over Multiple Paths. In *Proceedings of the 11th ACM International Conference on Multimedia (MULTIMEDIA '03)*, New York, New York, USA, November 2003. ACM Press. 36

[CJKK07] Szymon Chachulski, Michael Jennings, Sachin Katti, and Dina Katabi. Trading Structure for Randomness in Wireless Opportunistic Routing. In *SIGCOMM*, volume 37, 2007. 33, 34, 42, 43, 49, 52, 54, 57, 58

[CMWB08] Rui A. Costa, Daniele Munaretto, Joerg Widmer, and Joao Barros. Informed Network Coding for Minimum Decoding Delay. In *2008 5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, pages 80–91. IEEE, September 2008. 94

[Cov72] T. Cover. Broadcast Channels. *IEEE Transactions on Information Theory*, 18(1):2–14, January 1972. 35, 76

[CWCC08] Tein-Yaw Chung, Chih-Cheng Wang, Yung-Mu Chen, and Yang-Hui Chang. PNECOS: A Peer-to-Peer Network Coding Streaming System. In *2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC 2008)*, pages 379–384. IEEE, June 2008. 37, 116

[CWJ03] Philip A. Chou, Yunnan Wu, and Kamal Jain. Practical Network Coding. In *Allerton Conference on Communication, Control, and Computing*, 2003. 49

[CWP03] Philip A. Chou, Helen J. Wang, and Venkata N. Padmanabhan. Layered Multiple Description Coding. In *Packet Video Workshop*, 2003. 36

[DEH+05] Supratim Deb, Michelle Effros, Tracey Ho, David R. Karger, Ralf Koetter, Desmond S. Lun Lun, Muriel Médard, and Niranjan Rat-

nakar. Network Coding for Wireless Applications: A Brief Tutorial. In *International Workshop on Wireless Ad Hoc Networks*, 2005. 33

[DFZ05] R. Dougherty, C. Freiling, and K. Zeger. Insufficiency of Linear Coding in Network Information Flow. *IEEE Transactions on Information Theory*, 51(8):2745–2759, August 2005. 30

[DGWR07] A. G. Dimakis, P. B. Godfrey, M. J. Wainwright, and K. Ramchandran. Network Coding for Distributed Storage Systems. In *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pages 2000–2008. IEEE, May 2007. 36, 116

[FdMC08] Pascal Frossard, Juan Carlos de Martin, and M. Reha Civanlar. Media Streaming With Network Diversity. *Proceedings of the IEEE*, 96(1):39–53, January 2008. 94

[FKM+07] Christina Fragouli, Dina Katabi, Athina Markopoulou, Muriel Medard, and Hariharan Rahul. Wireless Network Coding: Opportunities & Challenges. In *MILCOM 2007 - IEEE Military Communications Conference*, pages 1–8. IEEE, October 2007. 94

[FL08] Chen Feng and Baochun Li. *On Large-Scale Peer-to-Peer Streaming Systems with Network Coding*. ACM Press, New York, New York, USA, October 2008. 37, 116

[FLSC04] Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. Robust Incentive Techniques for Peer-to-Peer Networks. In *Proceedings of the 5th ACM Conference on Electronic Commerce (EC '04)*, page 102, New York, New York, USA, May 2004. ACM Press. 38

[FS06a] Christina Fragouli and Emina Soljanin. Decentralized Network Coding. In *IEEE Information Theory Workshop*, pages 25–29. IEEE, 2006. 30

[FS06b] Christina Fragouli and Emina Soljanin. Network Coding Fundamentals. *Foundations and Trends in Networking*, 2(1):1–133, January 2006. 33, 35, 75

[Gal74] R. G. Gallager. Capacity and Coding for Degraded Broadcast Channels. *Problemy Peredachi Informatsii*, 10(3):185—-193, 1974. 35, 76

[GHK+07] Christos Gkantsidis, Wenjun Hu, Peter Key, Bozidar Radunovic, Pablo Rodriguez, and Steluta Gheorghiu. Multipath Code Casting for Wireless Mesh Networks. In *Proceedings of the 2007 ACM CoNEXT conference*, pages 1–12, New York, New York, 2007. ACM. 33, 36, 42, 43, 49, 52, 54, 57, 99, 100

[GLL+10] Steluta Gheorghiu, Luisa Lima, Alberto Lopez Toledo, Joao Barros, and Muriel Medard. On the Performance of Network Coding in Multi-Resolution Wireless Video Streaming. In *2010 IEEE International Symposium on Network Coding (NetCod)*, pages 1–6. IEEE, June 2010. 27, 35, 75, 95, 134

[GLLB11] Steluta Gheorghiu, Luisa Lima, Alberto Lopez Toledo, and João Barros. A Layered Network Coding Solution for Incentives in Peer-to-Peer Live Streaming. In *2011 IEEE International Symposium on Network Coding (NetCod)*, 2011. 27, 117, 134

[GLR10] Steluta Gheorghiu, Alberto Lopez Toledo, and Pablo Rodriguez. Multipath TCP with Network Coding for Wireless Mesh Networks. In *2010 IEEE International Conference on Communications*, pages 1–5. IEEE, May 2010. 26, 44, 133

[GLR11] Steluta Gheorghiu, Alberto Lopez Toledo, and Pablo Rodriguez. A Network Coding Scheme for Seamless Interaction with TCP. In *2011 IEEE International Symposium on Network Coding (NetCod)*, 2011. 26, 44, 133

[GR05] C. Gkantsidis and P.R. Rodriguez. Network Coding for Large Scale Content Distribution. In *Proceedings of IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005))*, pages 2235–2245. IEEE, 2005. 36, 116

[GSFL10] Steluta Gheorghiu, Shirin Saeedi Bidokhti, Christina Fragouli, and Alberto Lopez Toledo. Degraded Multicasting with Network Coding over the Combination Network - Infoscience. Technical report, EPFL, 2010. 80, 87, 133

[GSFL11] Steluta Gheorghiu, Shirin Saeedi Bidokhti, Christina Fragouli, and Alberto Lopez Toledo. Degraded Multicasting with Network Coding over the Combination Network. In *2011 IEEE International Symposium on Network Coding (NetCod)*, 2011. 27, 76, 78, 133

[GZL03] Jiang Guo, Ying Zhu, and Baochun Li. CodedStream: Live Media Streaming with Overlay Coded Multicast. In *Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking (MMCN 2004)*, pages 28–39, 2003. 37

[HKM+03] T. Ho, R. Koetter, M. Medard, D.R. Karger, and M. Effros. The Benefits of Coding over Routing in a Randomized Setting. In *IEEE International Symposium on Information Theory*. IEEE, 2003. 30, 94, 99, 118

[HKM05] Nicholas J. A. Harvey, David R. Karger, and Kazuo Murota. Deterministic Network Coding by Matrix Completion. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'05)*, pages 489–498, January 2005. 30

[HSH+06] Huaizhong Han, Srinivas Shakkottai, C. V. Hollot, R. Srikant, and Don Towsley. Multi-Path TCP: A Joint Congestion Control and Routing Scheme to Exploit Path Diversity in the Internet. *IEEE/ACM Transactions on Networking*, 14(6):1260–1271, December 2006. 41

[HSLG99] Uwe Horn, K. Stuhlmüller, M. Link, and B. Girod. Robust Internet Video Transmission Based on Scalable Coding and Unequal Error Protection. *Signal Processing: Image Communication*, 15:77–94, 1999. 93

[iPl] BBC - BBC Internet Blog: BBC iPlayer goes HD, adds higher quality streams, releases iPlayer Desktop out of Labs. In *http://www.bbc.co.uk/blogs/bbcinternet/2009/04/bbc_iplayer_goes_hd_adds_highe.html*. 21

[JLK08] J. Jin, B. Li, and T. Kong. Is Random Network Coding Helpful in WiMAX? In *2008 IEEE INFOCOM - The 27th Conference on Computer Communications*, pages 2162–2170. IEEE, April 2008. 94

[KGK07] Sachin Katti, Shyamnath Gollakota, and Dina Katabi. Embracing Wireless Interference: Analog Network Coding. In *Proceedings of SIGCOMM*, SIGCOMM '07, pages 397–408, New York, 2007. ACM. 33, 34

[KHKS04] Jan Kritzner, Uwe Horn, Markus Kampmann, and Joachim Sachs. Priority Based Packet Scheduling withTunable Reliability for Wireless Streaming. *Lecture Notes in Computer Science*, 3079:707–717, 2004. 94

[KM77] J. Korner and K. Marton. General Broadcast Channels with Degraded Message Sets. *IEEE Transactions on Information Theory*, 23(1):60–64, January 1977. 35, 76

[KMT06] Peter Key, Laurent Massoulie, and Don Towsley. Combining Multipath Routing and Congestion Control for Robustness. In *40th Annual Conference on Information Sciences and Systems, 2006*, pages 345—-350, 2006. 41

[KRH+06] Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Médard, and Jon Crowcroft. XORs in the Air: Practical Wireless Network Coding. In *SIGCOMM*, volume 36, pages 243–254, 2006. 33, 34, 94

[LGB+10] Luisa Lima, Steluta Gheorghiu, Joao Barros, Muriel Medard, and Alberto Lopez Toledo. Secure Network Coding for Multi-Resolution Wireless Video Streaming. *IEEE Journal on Selected Areas in Communications*, 28(3):377–388, April 2010. 27, 95, 134

[LL04] Zongpeng Li and Baochun Li. Network Coding in Undirected Networks. In *38th Conference on Information Science and Systems*, 2004. 29

[LL06] Zongpeng Li and Baochun Li. Improving Throughput in Multihop Wireless Networks. *IEEE Transactions on Vehicular Technology*, 55(3):762–773, 2006. 41

[LMK05] D. S. Lun, M. Medard, and R. Koetter. Efficient Operation of Wireless Packet Networks using Network Coding. In *International Workshop on Convergent Technologies (IWCT)*, 2005. 33

[LPCR02] Youngseok Lee, Ilkyu Park, Yanghee Choi, and A. Multipath Routing. Improving TCP Performance in Multipath Packet Forwarding Networks. *Journal of Communications and Networks*, 4:148–157, 2002. 33, 41

[LPDG06] Yajie Liu, Yuxing Peng, Wenhua Dou, and Bo Guo. Network Coding for Peer-to-Peer Live Media Streaming. In *2006 Fifth International Conference on Grid and Cooperative Computing (GCC'06)*, pages 149–155. IEEE, October 2006. 37, 116

[LSP+07a] Zhengye Liu, Yanming Shen, Shivendra S. Panwar, Keith W. Ross, and Yao Wang. P2P Video Live Streaming with MDC: Providing Incentives for Redistribution. In *2007 IEEE International Conference on Multimedia and Expo*, pages 48–51. IEEE, July 2007. 38, 116

[LSP+07b] Zhengye Liu, Yanming Shen, Shivendra S. Panwar, Keith W. Ross, and Yao Wang. Using Layered Video to Provide Incentives in P2P Live Streaming. In *Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV - P2P-TV '07*, New York, New York, USA, August 2007. ACM Press. 38, 95, 116, 118

[LW06] Alberto Lopez Toledo and Xiaodong Wang. Efficient Multipath in Sensor Networks using Diffusion and Network Coding. In *2006 40th Annual Conference on Information Sciences and Systems*, pages 87–92. IEEE, March 2006. 58

[LY03] S.-Y.R. Li and R.W. Yeung. Linear Network Coding. *IEEE Transactions on Information Theory*, 49(2):371–381, February 2003. 29

[Nap] Napster. In *http://www.napster.com*. 21

[NCO04] Vu Thanh Nguyen, Ee Chien Chang, and Wei Tsang Ooi. Layered coding with good allocation outperforms multiple description coding over multiple paths. In *2004 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1067–1070. IEEE, 2004. 36

[NE09] Chandra Nair and Abbas El Gamal. The Capacity Region of a Class of Three-Receiver Broadcast Channels with Degraded Message Sets. *IEEE Transactions on Information Theory*, 55(10):4479–4493, October 2009. 35, 76

[Net] Netflix. In *http://www.netflix.com/*. 21

[NNC07] Kien Nguyen, Thinh Nguyen, and Sen-ching Cheung. Peer-to-Peer Streaming with Hierarchical Network Coding. In *2007 IEEE*

International Conference on Multimedia and Expo, pages 396–399. IEEE, July 2007. 37

[ns2] The ns-2 network simulator. In *http://nsnam.isi.edu/nsnam/index.php/User_Information*. 43, 44, 57, 64, 66, 104, 126

[PIKA08] Michael Piatek, Tomas Isdal, Arvind Krishnamurthy, and Thomas Anderson. One hop reputations for peer to peer file sharing workloads. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI'08)*, pages 1–14, April 2008. 38

[PPL] PPLive — CrunchBase Profile. In *http://www.crunchbase.com/company/pplive*. 21

[PRSR06] Lucian Popa, Costin Raiciu, Ion Stoica, and David Rosenblum. Reducing Congestion Effects in Wireless Networks by Multipath Routing. In *Proceedings of the 2006 IEEE International Conference on Network Protocols*, pages 96–105. IEEE, November 2006. 32

[PWC03] V.N. Padmanabhan, H.J. Wang, and P.A. Chou. Resilient peer-to-peer streaming. In *11th IEEE International Conference on Network Protocols*, pages 16–27. IEEE Comput. Soc, 2003. 36, 38

[PWCS02] Venkata N. Padmanabhan, Helen J. Wang, Philip A. Chou, and Kunwadee Sripanidkulchai. Distributing streaming media content using cooperative networking. In *Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '02)*, page 177, New York, New York, USA, May 2002. ACM Press. 36

[RGGK08] Božidar Radunović, Christos Gkantsidis, Dinan Gunawardena, and Peter Key. Horizon: balancing tcp over multiple paths in wireless mesh network. In *International Conference on Mobile Computing and Networking*, pages 247–258, 2008. 33, 42, 43, 57

[RN04] Vincent Roca and Christoph Neumann. Design, Evaluation and Comparison of Four Large Block FEC Codes, LDPC, LDGM, LDGM Staircase and LDGM Triangle, plus a Reed-Solomon Small Block FEC Codec, 2004. 65

[Roo] Roofnet. In *http://pdos.csail.mit.edu/roofnet/doku.php*. 58, 66

[SDFP09] Shirin Saeedi, Suhas Diggavi, Christina Fragouli, and Vinod Prabhakaran. On degraded two message set broadcasting. In *2009 IEEE Information Theory Workshop*, pages 406–410. IEEE, 2009. 35, 76

[SE06] Yalin Sagduyu and Anthony Ephremides. Some Optimization Trade-offs in Wireless Network Coding. In *40th Annual Conference on Information Sciences and Systems*, pages 6–11. IEEE, March 2006. 33

[SFC08] Thomas Silverston, Olivier Fourmaux, and Jon Crowcroft. Towards an Incentive Mechanism for Peer-to-Peer Multimedia Live Streaming Systems. In *8th International Conference on Peer-to-Peer Computing*, pages 125–128. IEEE, September 2008. 38

[SM07] Hulya Seferoglu and Athina Markopoulou. Opportunistic network coding for video streaming over wireless. In *Packet Video 2007*, pages 191–200. IEEE, November 2007. 94

[SOPJ00] Raghavendra Singh, Antonio Ortega, Lionel Perret, and Wenqing Jiang. Comparison of Multiple Description Coding and Layered Coding based on Network Simulations. In *Image and Video Communications and Processing*, pages 929–939. SPIE, 2000. 35, 36

[Spo] Spotify. In *http://www.spotify.com/uk/about/features/*. 21

[SRB05] Niveditha Sundaram, Parameswaran Ramanathan, and Suman Banerjee. Multirate media stream using network coding. *Proceedings of 43rd Annual Allerton Conference on Communication, Control, and Computing*, 2005. 94

[SSM09a] Jay Kumar Sundararajan, Devavrat Shah, and Muriel Medard. Feedback-based online network coding. In *CoRR*, April 2009. 99

[SSM+09b] Jay Kumar Sundararajan, Devavrat Shah, Muriel Médard, Michael Mitzenmacher, and Joao Barros. Network coding meets TCP. In *IEEE INFOCOM*, 2009. 34, 43, 50, 51, 57, 63

[TE92] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936–1948, 1992. 54

[TF00] A.S. Tosun and W.-C. Feng. Efficient multi-layer coding and encryption of MPEG video streams. In *2000 IEEE International Conference on Multimedia and Expo (ICME2000)*, pages 119–122. IEEE, 2000. 35, 116

[TF07] Nikolaos Thomos and Pascal Frossard. Raptor network video coding. In *Proceedings of the International Workshop on Mobile Video - MV '07*, New York, New York, USA, September 2007. ACM Press. 37, 117

[WL05] Jörg Widmer and Jean-Yves Le Boudec. Network coding for efficient communication in extreme networks. In *Proceeding of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking - WDTN '05*, pages 284–291, New York, New York, USA, August 2005. ACM Press. 36, 116

[WL07a] Mea Wang and Baochun Li. Network Coding in Live Peer-to-Peer Streaming. *IEEE Transactions on Multimedia*, 9(8):1554–1567, December 2007. 36, 116

[WL07b] Mea Wang and Baochun Li. R2: Random Push with Random Network Coding in Live Peer-to-Peer Streaming. *IEEE Journal on Selected Areas in Communications*, 25(9):1655–1666, December 2007. 37, 116

[WPLM02] Yao Wang, Shivendra Panwar, Shunan Lin, and Shiwen Mao. Wireless Video Transport Using Path Diversity: Multiple Description Vs. Layered Coding. In *International Conference on Image Processing*, pages 21–24. IEEE, 2002. 36

[ZYZ+06] J. Zhao, F. Yang, Q. Zhang, Z. Zhang, and F. Zhang. LION: Layered Overlay Multicast With Network Coding. *IEEE Transactions on Multimedia*, 8(5):1021–1032, October 2006. 38