U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

# Coding over Multiple Wireless Interfaces

**André Manuel Longo Moreira**

Supervisor: Prof. Dr. Daniel Enrique Lucani Roetter

Co-Supervisor: Prof. Dr. Luis Miguel Pinho de Almeida

Programa Doutoral em Engenharia Electrotécnica e de Computadores

March, 2017

Faculdade de Engenharia da Universidade do Porto

# Coding over Multiple Wireless Interfaces

**André Manuel Longo Moreira**

Dissertation submitted to Faculdade de Engenharia da Universidade do Porto
to obtain the degree of

**Doctor Philosophiae in Electronic & Computer Engineering**

March, 2017

*Este trabalho foi parcialmente financiado pela Fundação para a Ciência e a Tecnologia através da bolsa de doutoramento SFRH/BD/81726/2011, e dos projectos PTDC/EEI-TEL/3006/2012 (CodeStream) e UID/EEA/50008/2013*

*Aos meus pais e irmão,*
*a ti Ana,*
*e aos amigos.*

# Abstract

The unique advantages of wireless communications make them an appealing solution to a growing number of applications. Thus, an increased number of devices and new sets of requirements must be satisfied. To support such ubiquitous communications, next generation 5G networks are expected to withstand 1000-times the capacity of current mobile networks and provide much lower end-to-end latencies, among other ambitious requirements. At the same time, the set of coexisting technologies is richer than ever. Our view is that the increased demands for low latency, highly reliable, and high data rate communication requires existing and new systems to explore alternative methods that go beyond a single active wireless communication link to compensate for highly volatile channel conditions. In particular, this work will consider using multiple wireless interfaces simultaneously, possibly with heterogeneous technologies, to boost information transfer. Then, we advocate that packet-level coding, e.g. network coding, is critical to achieve efficient and practical multipath scheduling mechanisms. To assess the validity of our claims, we investigate the delay performance of coded multipath schemes over a single wireless hop, and provide contributions to minimize different kinds of delay.

Our first set of contributions concerns block transmission delays. The first contribution is a practical implementation of file transfer using multiple interfaces on mobile devices. The developed application showcases the challenges as well as the benefits of aggregating multiple interfaces, and the benefits of coding against packet losses. The second contribution provides a fundamental understanding of the transmission of a block of packets. In this case, we derive a lower bound for the amount of information that can be sent with a given reliability over asymmetric and time-constrained interfaces. Leveraging these results, we devise optimal and heuristic policies for a couple of coded schemes, whose evaluation shows that coding and splitting packets across multiple interfaces is at the heart of minimizing the transmission time.

Our second set of contributions deals with stream transmissions and looks at packet level delays. Our first take on this subject considers stream transmission between two nodes over time-varying links and addresses the fundamental trade-off between queuing delay and energy consumption. The proposed solution merges online network coding and opportunistic scheduling under a decision theoretic framework. A key observation is that channel diversity plays a very important role on the individual packet service delay. Finally, we analyze the in-order delivery delay of packets to an application, proposing coded schemes that aim to minimize the worst-case delay in limited feedback conditions.

Based on our observations, we conclude that communicating over multiple interfaces in simultaneous offers clear benefits that stem not only from (i) increased bandwidth but also (ii) diversity. Also, coding techniques are fundamental in seizing such benefits by (i) ensuring that bandwidth is efficiently utilized in limited feedback conditions, and (ii) greatly simplifying the design of scheduling mechanisms in the presence of asymmetric channel characteristics.

# Resumo

As vantagens únicas das comunicações sem fios fazem destas uma solução atrativa para um número crescente de aplicações. Assim, um maior número de dispositivos e conjuntos de requisitos tem de ser satisfeitos. De forma a suportar estas comunicações ubíquas, espera-se que as redes 5G de próxima geração sejam capazes de suportar 1000-vezes a capacidade das redes móveis atuais e apresentar latências ponto-a-ponto reduzidas, entre outros requisitos ambiciosos. Ao mesmo tempo, o conjunto de tecnologias coexistentes nunca foi tão variado. No nosso ponto de vista, as exigências crescentes por comunicações de baixa latência, altamente fiáveis, e com elevadas taxas de transmissão requerem que os sistemas atuais e do futuro explorem métodos alternativos que ultrapassem uma única ligação sem fios ativa, de forma a compensar condições de canal altamente voláteis. Em particular, este trabalho considera o uso de múltiplas interfaces sem fios em simultâneo, possivelmente usando tecnologias heterogéneas, de forma a melhorar a transferência de informação. Depois, defendemos que codificação ao nível do pacote, e.g. codificação em rede, é fundamental para alcançar mecanismos de escalonamento multi-caminho eficientes e práticos. Para avaliar a validade das nossas alegações, investigámos o desempenho em termos de atraso de esquemas de transmissão multi-caminho codificados, ao longo de um único salto de rede, e proporcionámos contribuições para minimizar diferentes tipos de atrasos.

O primeiro grupo de contribuições diz respeito a atrasos de transmissão por blocos. A primeira contribuição é uma implementação prática de transferência de ficheiros usando múltiplas interfaces em dispositivos móveis. A aplicação desenvolvida demonstra os desafios, assim como os benefícios de agregar múltiplas interfaces, e os benefícios de codificar contra as perdas de pacotes. A segunda contribuição, fornece uma compreensão fundamental acerca da transmissão de um bloco de pacotes. Neste caso, derivamos um limite inferior para a quantidade de informação que pode ser enviada com uma dada fiabilidade através de interfaces assimétricas e restringidas temporalmente. Construindo em cima destes resultados, determinamos políticas ótimas e heurísticas para um par de esquemas codificados, cuja avaliação revela que codificar e dividir pacotes por múltiplas interfaces é central para minimizar o tempo de transmissão.

O segundo grupo de contribuições lida com transmissões de fluxos de informação e analisa atrasos a nível do pacote. A nossa primeira investida neste tópico considera uma transmissão entre dois nós através de ligações variáveis no tempo e aborda o compromisso fundamental entre o tempo de espera dos pacotes e o consumo de energia. A solução proposta junta uma variante online de codificação em rede com escalonamento oportunístico num modelo de decisão teórico. Uma observação chave é que a diversidade de canais desempenha um papel bastante importante no atraso de serviço de cada dos pacotes. Finalmente, analisamos a distribuição do atraso de entrega por ordem de pacotes à aplicação, propondo esquemas que procuram minimizar o pior case do atraso em condições de feedback limitado.

Baseado nas nossas observações, concluímos que comunicar através de múltiplas interfaces em simultâneo oferece claros benefícios que derivam não só da (i) largura de banda acrescida, mas também da (ii) diversidade. Mais ainda, técnicas de codificação são fundamentais para capturar tais benefícios (i) assegurando que a largura de banda é utilizada de forma eficiente em condições de feedback limitado, e (ii) simplificando bastante o design de mecanismos de escalonamento na presença de características de canal assimétricas.

# Agradecimentos

E lá foram mais cinco anos, a somar a outros tantos nesta casa. Foi um percurso feito de altos e baixos, com muita aprendizagem e muita gente à mistura!

Em primeiro lugar gostaria de agradecer aos meus orientadores, Daniel Lucani e Luis Almeida, sem nenhum dos quais este trabalho teria sido possível. Ao Daniel por ter visto potencial em mim e me ter convidado a iniciar esta jornada, assim como pelos conhecimentos, métodos e confiança que me foi transmitindo. Ao Luis, por me ter acolhido, pela sua simpatia, sinceridade e incentivo, bem como pela sua incansável dedicação ao laboratório DaRTES, fazendo deste não só um local de trabalho mas também um local de convívio saudável e agradável.

Gostaria também de agradecer a todos os meus colegas, Alessandro Rucco, Ana Rita Pereira, Aqsa Aslam, Behzad Zamani, David Gessner, Marlos Marques, Shuai Wan, Sílvia Bessa e Zahid Iqbal, entre outros, que foram passando pelo laboratório, oriundos dos mais variados lugares, e com quem tive a oportunidade de conviver e aprender. Depois, em particular, àqueles com quem mantive uma relação especial de amizade. Ao Carlos Pereira, que iniciou este percurso comigo e com quem tive inúmeras conversas. Ao Luis Oliveira, sempre bem disposto e disposto a ajudar. À Inés Alvarez e ao Sydney Carvalho, pela energia positiva que trouxeram ao nosso grupo e pelos laços de amizade que criaram connosco. Ao Luis Pinto, amigo de infância que tive a sorte de ter também como colega durante esta etapa, tornando o ambiente de trabalho mais familiar. E claro está, aos restantes amigos, Mafalda, Mário, Raquel, Ivo, Iracy, Rui, Daniela, Hugo, assim como aos mais pequenos, Caio, Inês, Sofia e Vasco, que tornaram a minha vida mais alegre.

Naturalmente, quero também agradecer à minha família, à que já tinha e à acrescida. À minha madrinha, Fernanda, que sempre me ajudou e me serviu de exemplo. Ao meu irmão, Rafael, que teve sempre um sorriso amigo para mim e com quem pude contar sempre. E um obrigado do fundo do coração aos meus pais, Cassilda e Manuel, por terem feito de mim o que sou hoje e que me trouxe até aqui!

Por último, um agradecimento muito especial à minha noiva, Ana, que esteve sempre ao meu lado, nos bons e nos maus momentos, dando-me a estabilidade, serenidade e ânimo necessários! ♡

Obrigado a todos!
André Moreira

# Publications

A. Moreira, D. E. Lucani, “Coded Schemes for Asymmetric Wireless Interfaces: Theory and Practice,” *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 22, pp. 171–184, Feb. 2015.

A. Moreira, D. E. Lucani, “On coding for asymmetric wireless interfaces,” in *Proc. IEEE Int. Symp. Network Coding*. Cambridge, MA, June 2012, pp. 149–154.

A. Moreira, D. E. Lucani, “Opportunistic Online Network Coding for Multiple Wireless Channels: Send, Probe, or Wait,” *IEEE Transactions on Communications*, (submitted)

A. Moreira, L. Almeida, D. E. Lucani, “Merging network coding with feedback management in multicast streaming,” *ACM SIGBED Rev.*, vol. 12, no. 3, pp. 49–52, June 2015.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **3G** | 3rd Generation of Wireless Mobile Telecommunications Technology |
| **4G** | 4th Generation of Wireless Mobile Telecommunications Technology |
| **5G** | 5th Generation of Wireless Mobile Telecommunications Technology |
| **ACK** | Acknowledgment |
| **API** | Application Public Interface |
| **ARQ** | Automatic Repeat reQuest |
| **BT** | Bluetooth |
| **CMT** | Concurrent Multipath Transfer |
| **CSI** | Channel State Information |
| **D2D** | Device-to-Device |
| **FEC** | Forward Error Correction |
| **FEUP** | Faculdade de Engenharia da Universidade do Porto |
| **FIFO** | First-In, First-Out |
| **FSMC** | Finite-State Markov Chain |
| **GE** | Gilbert-Elliot |
| **GF** | Galois Field |
| **MAC** | Medium Access Control |
| **MDP** | Markov Decision Process |
| **MIMO** | Multiple-Input Multiple Output |
| **mmWave** | Millimeter-Wave |
| **MPTCP** | Multipath Transmission Control Protocol |
| **NAT** | Network Address Translation |
| **NC** | Network Coding |
| **NFC** | Near-Field Communication |
| **OS** | Operating System |
| **PEC** | Packet Erasure Channel |
| **PER** | Packet Error Rate |
| **PGM** | Probabilistic Graph Model |
| **POMDP** | Partially Observable Markov Decision Process |

| | |
|---|---|
| **PPTP** | Point-to-Point Tunneling Protocol |
| **RFCOMM** | Radio Frequency Communication |
| **RLNC** | Random Linear Network Coding |
| **RTT** | Round-Trip Time |
| **SCTP** | Stream Control Transmission Protocol |
| **TCP** | Transmission Control Protocol |
| **UDP** | User Datagram Protocol |
| **VPN** | Virtual Private Network |
| **WLAN** | Wireless Local Area Network |
| **WPAN** | Wireless Personal Area Network |

# Chapter 1

# Introduction

## 1.1 Motivation

Wireless communication stands as one of the core technologies of the modern world. Its nature enables ubiquitous communication and access to information, but also makes it a fragile support for communication. Based on its widespread adoption though, it seems clear that the convenience it provides far compensates the efforts in developing new solutions to attenuate its weaknesses. So much that wireless communication technologies are continuously evolving, as users keep demanding improved performance and more devices become interconnected. To satisfy the needs predicted in the near future, next generation 5G networks are expected to withstand 1000-times the current capacity and provide much lower end-to-end latencies, among other ambitious requirements. In order to achieve these goals, new techniques must be developed and the limits of technology must be pushed. With that in mind, this work leverages coding techniques to exploit the diversity of wireless technologies, in order to improve performance over the wireless segment, with particular emphasis on delay.

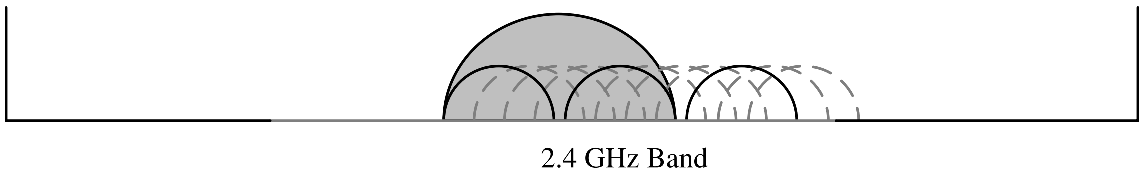
## 1.2 The Wireless Mosaic

Wireless communication serves a myriad of applications such as Internet access, file transfer, messaging, streaming, sensing, and many others. The need to support such a variety of applications along with their different requirements has made it difficult to design a wireless system that can efficiently satisfy all requirements in simultaneous. This led to a fragmentation of wireless technologies and development of a large collection of standards. Today, wireless communication encompasses systems that span from body area networks to global cellular and satellite networks, operating over different frequency bands and employing different transmission techniques. The IEEE 802.11 standard targets wireless local area networks (WLANs), and within it different amendments focus on particular applications. IEEE 802.15 defines standards for wireless personal area networks (WPANs), which have short-range and low-energy requirements, underlying protocols such as Bluetooth and ZigBee. Cellular networks have their own set of technologies, with LTE being the flagship for the 4th generation of mobile communication technologies. The list continues and this trend is expected to prevail as new technologies are developed and integrated with the existing ones. One striking example is the emergence of millimeter-wave (mmWave) communications, set to operate in the 60 GHz unlicensed band, and expected to play a major role in future networks. This panorama reveals a heterogeneous network architecture where a variety of radio access technologies with distinct characteristics and capabilities coexist with each other [1]. Future 5G networks are expected to further stress this trend with much denser and diverse cell and access point deployments. Moreover, it is not only on the infrastructure side that this trend can be observed. Many devices come equipped with multiple communication interfaces, with smartphones being a clear and familiar example. These small devices bring 4G, Wi-Fi and Bluetooth, which provides them with multiple connectivity options. Despite the variety of resources, the current paradigm consists in establishing communication over a single radio interface. We believe that this solution is not consistent with the unreliable and dynamic nature of wireless links. This work stands for a different approach that is based on exploiting this diversity. To that end, it proposes multipath transmission schemes over multiple wireless interfaces with network coding as a key enabler.

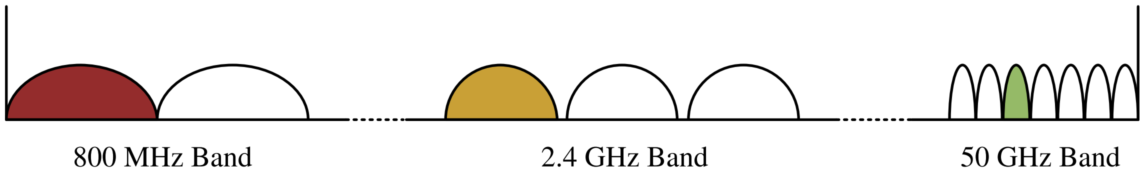
## 1.3 Thesis Statement

In establishing our claims, we provide a primitive intuition on how the ideas explored in this work are expected to synergize and deliver benefits in the field of wireless communications.

The *radio spectrum* is a scarce resource that must be wisely managed. It is organized in limited frequency intervals called *channels*, whose bandwidth inherently limits the maximum rate of communication. Since channel width is standardized for each technology, applications are bound to it, even though some applications require more bandwidth than others. In this context, aggregation techniques offer a flexible and cheap solution to support applications that require higher bandwidth by aggregating multiple channels together. This is the basis for modern channel bonding techniques present in IEEE 802.11n and IEEE 802.11ac, which bundle two adjacent channels together in the same band for data communication (Figure 1.1a). With heterogeneous technologies, there is an added benefit in that channels do not need to be adjacent or even in the same band, providing further flexibility and robustness to problems in a particular zone of the spectrum (Figure 1.1b). Another major issue with wireless communications is the lack of *reliability*. The sheer openness of the wireless medium leaves signals susceptible to all kinds of physical phenomena, inducing signal quality changes over time, which are exacerbated in mobility scenarios where connectivity changes are likely to occur. Naturally, these effects are felt in all aspects of communication performance. Besides long-term *throughput*, nodes need to retransmit information leading to higher *energy* consumption and packet transmission *delay*. Clearly, keeping a communication active over multiple channels has the advantage of aggregating the



(a) Channel Aggregation



(b) Interface Aggregation

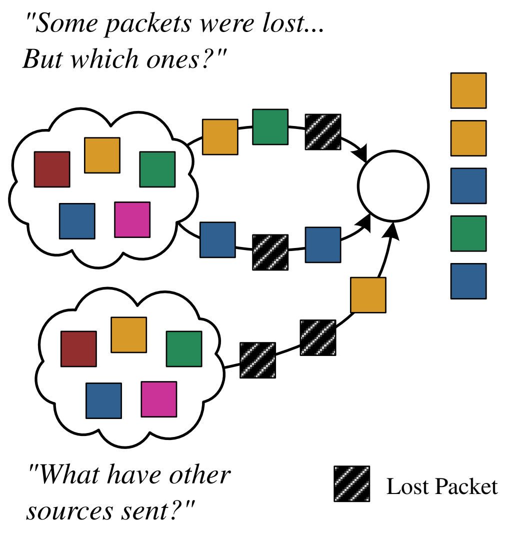
Figure 1.1: Spectrum utilization view of different channel aggregation approaches, comparing traditional channel bonding techniques (1.1a) with interface aggregation techniques (1.1b). The latter provides increased flexibility and robustness by aggregating bands with different characteristics.

bandwidth of both. But on top of that, keeping a diverse set of connections active increases the chances of having at least one channel with good signal quality, especially if these are uncorrelated. Even if the average throughput is not increased, applications benefit from smoother distribution on the joint channel quality, being able to send packets more regularly and thus with reduced delay. These insights lead us to our first claim.

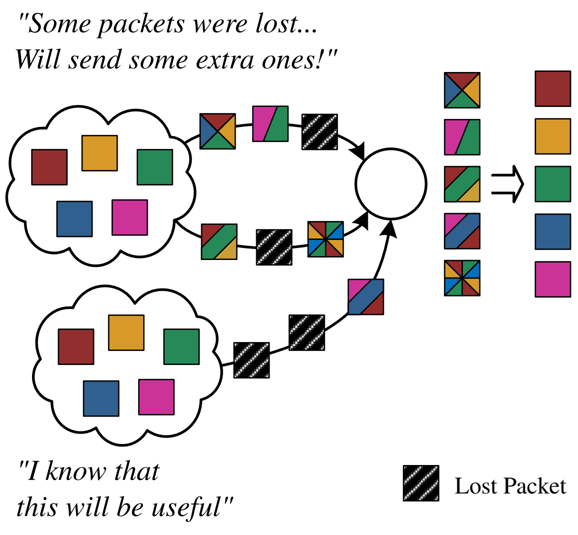
> **Claim 1.** *Wireless systems can greatly benefit from exploiting different wireless links in simultaneous to boost performance and robustness.*

While interface aggregation techniques look promising, achieving an efficient use of them entails nontrivial challenges. In the first place, there is the wireless nature of the considered paths which by itself imposes considerable challenges, among which we highlight the occurrence of *random packet errors* and the half-duplex communication constraint which usually implies *limited feedback information*. This means that scheduling has to be done with limited information about the communication's state. Then, compared to a single path, the packet scheduling problem gains an extra dimension, as decisions are not just confined to when to send packets and how much redundancy, but also over which interface to send these. Scheduling over this new dimension is a challenging problem in face of the asymmetric path characteristics. Generally, interfaces present different bit rates, round-trip times, packet erasure rates and energy consumption, which turn scheduling into a complex decision problem.

This is where *network coding* (NC) comes to the rescue. In basic terms, NC involves the mixing of original data packets into units of information called *coded packets*. In a sense, each coded packet



(a) Uncoded Scheduling


"Some packets were lost...
Will send some extra ones!"
"I know that this will be useful"
Lost Packet

(b) Coded Scheduling

Figure 1.2: Comparison of the packet scheduling problem when using traditional transmissions (1.2a) versus coded transmissions (1.2b) over lossy half-duplex channels.

acts as a wildcard, able to replace any of the original packets encoded in it. The advantages stemming from this very simple idea are multiple. For one, network coding works as a forward error correction mechanism where redundant coded packets can be sent in advance to compensate for lost packets, capable of delivering new information with minimal feedback. Furthermore, the rateless nature of the code, meaning that an arbitrary number of combinations can be generated from the original data, is particularly suitable to wireless communication scenarios where the error rate can change during the transmission. In terms of multipath scheduling, network coding can also greatly simplify the problem. The polymorphic character of coded packets enables stateless schemes, where the source does not need to keep track of which specific packets have been sent over each path (Figure 1.2). This is also useful if data is being transmitted from multiple sources as these will not need to coordinate which packets will be sent by each one. This can very well happen in multihoming scenarios where devices receive data from heterogeneous radio technologies. This leads us to our second claim.

**Claim 2.** *Network coding is an enabler to deliver flexible, practical and efficient mechanisms to manage multiple heterogeneous communication interfaces.*

Our work shall provide the support for these two claims, which are presented separately based on the following reasoning. The first claim, asserts that wireless systems can benefit from interface aggregation mechanisms which, being true, is valid regardless of the underlying techniques employed. Beyond that, it is our belief that network coding techniques are a natural and fitting solution to this problem, and that is what we will also attempt to convey and demonstrate throughout this thesis.

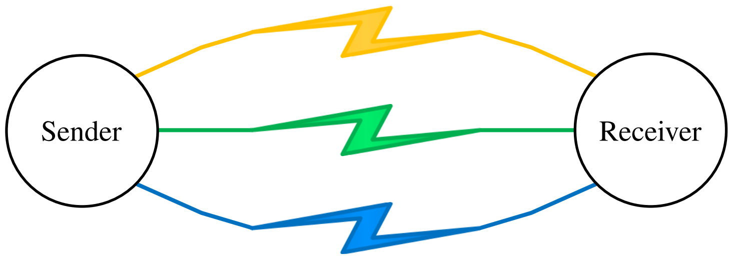


Figure 1.3: Reference model considered in this work, where one sender and one receiver communicate over multiple wireless channels.

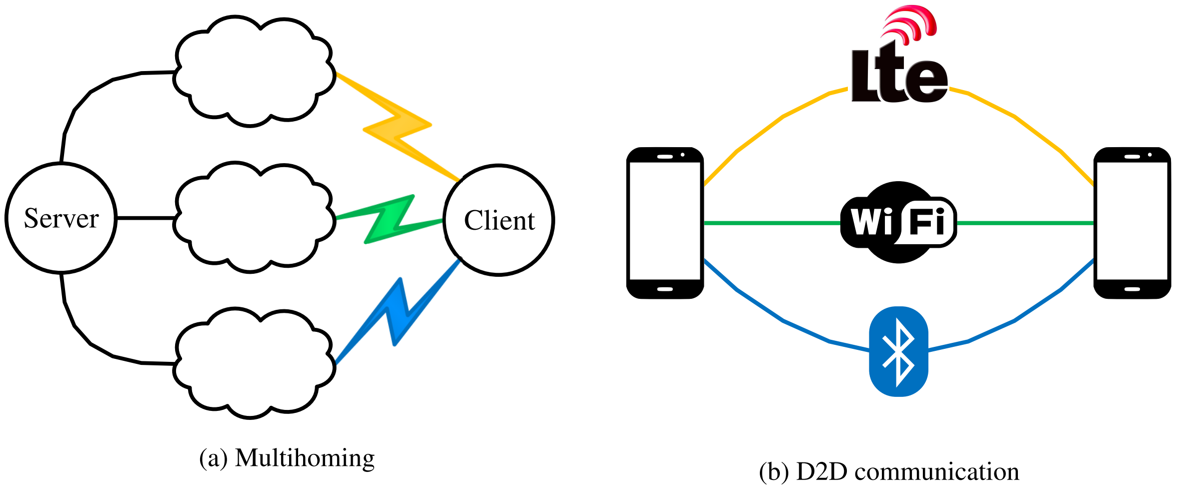
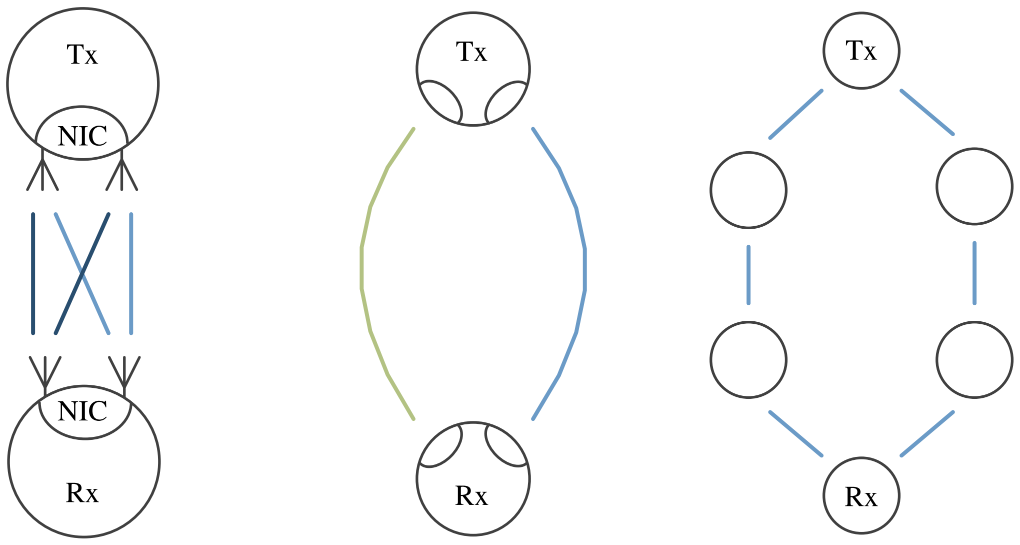


Figure 1.4: Two use cases for our work are multihoming (1.4a) and device-to-device (d2d) communication (1.4b). In either case, wireless plays a major role and multiple technologies are available.

## 1.4 Scope

This thesis studies wireless communications over a single hop, as represented by the reference model in Figure 1.3. The wireless link is the fundamental building block in many scenarios of interest, such as last mile access (Figure 1.4a) and device-to-device communication (Figure 1.4b). Therefore, understanding this elementary part is the basis for developing more complex protocols. This seemingly simple setting serves as a magnifying glass to examine communications over multiple wireless links, the most distinctive aspect of this work in terms of network configuration. In this work, we use the terms *path* and *link* interchangeably to refer to a logical communication link between two nodes. In turn, the term *channel* is used to refer to the physical support of a given link.



(a) Physical-layer (b) Link-layer (c) Network-layer

Figure 1.5: Diversity exploited at different communication layers.

The idea of exploiting diversity is not new, but it remains rather unexplored at the link layer (Figure 1.5b). At the physical layer (Figure 1.5a), MIMO systems use multiple antennas to exploit multipath propagation and thus mitigate problems related to fading. At the network layer (Figure 1.5c), wireless protocols exploit multipath routing to circumvent deadspots. Yet, a careful investigation of data transmission over multiple wireless interfaces is lacking. As argued in [2], diversity across all layers can provide enhanced reliability and performance. Indeed, these different diversity layers are not mutually exclusive but can be used in conjunction with each other.

The proposed approach is based on network coding premises which brings several advantages in terms of system design and performance. Although a single hop is considered, such hop may be part of a more complex system, reinforcing the suitability of network coding techniques. Based on this assumption, most of our results can be generalized to the cases where sender and receiver are not necessarily source and sink, but rather intermediate nodes in a network.

Despite the uncertainty that it undertakes, wireless communication is increasingly being adopted in time-sensitive applications, where short response times are not just a matter of convenience but actual effectiveness and safety. Whatever the driving factor, delay is a core metric in today's fast pacing world, with direct reflection on the perceived quality of communications. But it is also one with multiple scales. Depending on the application, one may be concerned with file transmission times down to individual packet delays. On top of that, coding operations add yet another layer of delay that is related with the extra time needed to decode information. All these aspects need to be jointly considered in the design of adequate solutions.

## 1.5 Outline

This work is organized as follows. The background theory on network coding is provided in Chapter 2, covering its origins, benefits and main principles of operation. Review of related work follows in Chapter 3. After these introductory chapters, a series of four chapters presents the main body of work.

Chapter 4 starts with a real showcase of the proposed ideas on commercial devices, presenting an Android application for file transfer over multiple interfaces. The architecture of the proposed protocol is presented as well as experimental results that show the benefits stemming from bandwidth aggregation, robustness to individual link failures, and resilience to packet losses. It also discusses the obstacles found in such platforms to implement the intended solution.

Chapter 5 builds some theoretical foundations for coded transmissions over asymmetric wireless interfaces. In particular, it develops bounds on the amount of information that can be transmitted with some target reliability over multiple interfaces in a limited time interval. These results are used to guide the scheduling of two coded schemes that aim to minimize the time to complete the transmission, subsequently evaluated in the context of high latency links.

Chapter 6 analyzes online network coding for unicast streaming over multiple channels with time-varying quality. The focus lies on balancing packet transmission delay and energy consumption using opportunistic schemes under imperfect knowledge conditions. The problem is solved with recourse to a decision theoretic framework that yields the policies for different trade-offs of the referred metrics. The results evidentiate that the advantages of using multiple uncorrelated links reach well beyond bandwidth aggregation, and are also a product of more diverse channel conditions.

Chapter 7 considers a multicast streaming scenario and fills the gap in terms of delay analysis, investigating the distribution of the in-order decoding delay under a limited feedback scenario.

All chapters address different nuances of coded transmissions having delay performance as an anchor point, with each chapter dealing with a different kind of delay. From this perspective, the chapters are organized from coarser to finer kinds of delay, as illustrated in Figure 1.6.

Following this presentation, Chapter 8 draws the general conclusions, evaluates the thesis and points some directions for future work.

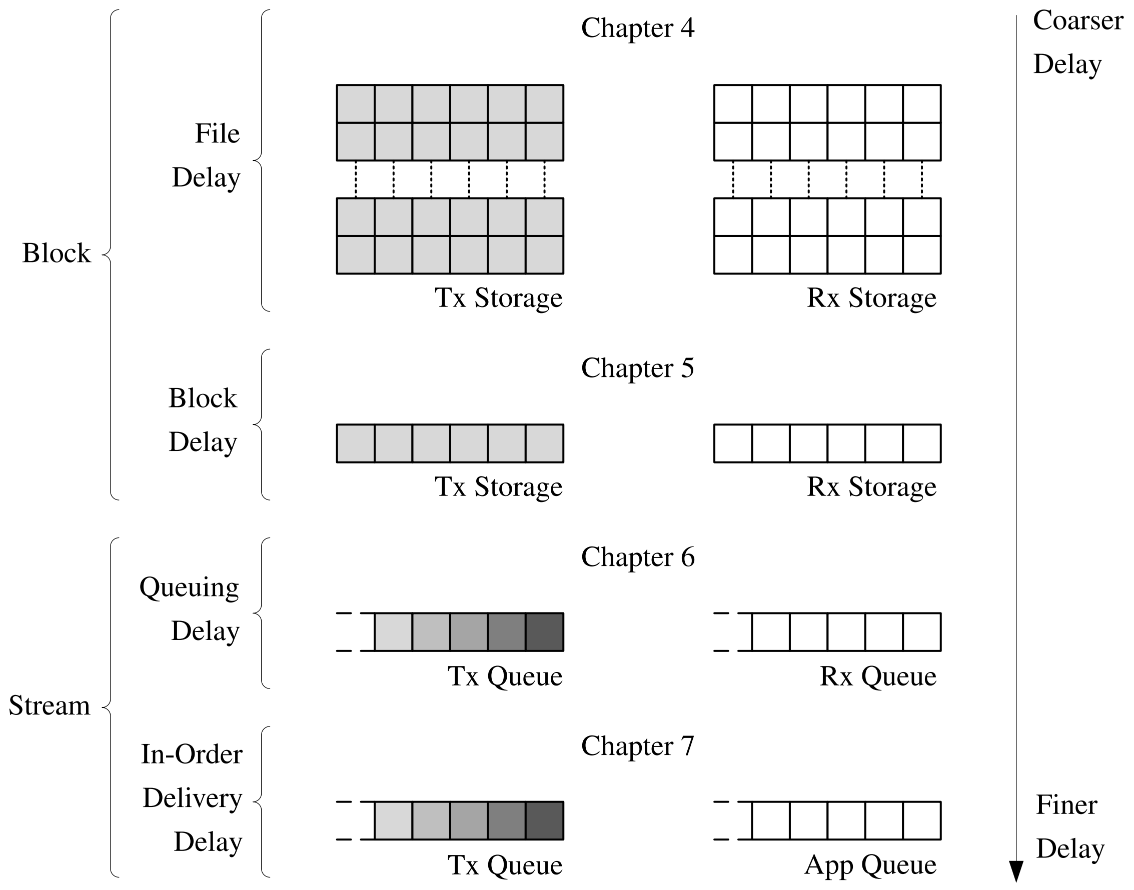


Figure 1.6: Thesis outline from a delay perspective.

# Chapter 2

# Network Coding

In this chapter we introduce network coding (NC), a technique with unique features introduced by Ahlswede *et al.* [3]. The essence of the network coding concept is that nodes in a network should combine the packets they receive and transmit those combinations instead of the original packets. This simple idea has been shown to have clear information theoretic benefits, the first one being the possibility to reach the multicast capacity of a network. Yet, it breaks with the traditional *store-and-forward* paradigm where intermediate nodes in a network simply forward the packets they receive, suggesting a *code-and-forward* approach. In order for such a disruptive concept to make it into the well established communication systems, it must provide sound arguments, so its benefits must be well studied in theory and demonstrated in practice. Since its inception, network coding has received an ever growing attention from the research community and a wide collection of works has shown its benefits in different applications and configurations. Throughout this chapter, we will provide an overview of this revolutionary technique, going over the aspects that are most relevant for this work.

## 2.1 Background

The seminal work of Ahlswede *et al.* [3] on network coding introduced the class of network information flow problems. The core of these problems is to determine the admissible coding rate region for some network configuration. The main result of this work states that network coding allows to achieve the max-flow min-cut capacity for multicast scenarios with a single source. The question back then was whether and how such codes could be constructed. Three years later, a series of works [4, 5, 6, 7, 8, 9] provided results that underlie much of today's practical code constructions. First, Li *et al.* [4] showed that linear codes are sufficient to achieve the multicast capacity. Afterwards, Koetter *et al.* [5] provided an algebraic framework for using linear codes. Centralized algorithms for computing codes in polynomial-time were proposed independently by Jaggi *et al.* [6] and Sanders *et al.* [7], and later consolidated in [10]. A decentralized solution was given by Ho *et al.* [8, 11], who showed that codes generated at random in each node are able to asymptotically achieve the multicast capacity, given sufficiently large field sizes. The solution, known as Random Linear Network Coding (RLNC), allies simplicity and high performance, standing as a reference technique in the field. Still, receivers must somehow have knowledge of the encoding functions in order to decode the information. A first practical approach to network coding was then provided by Chou *et al.* [9], who suggested appending the *encoding information* to the coded packets, and the grouping of packets into *generations*. These features allow for completely decentralized and asynchronous operation, providing robustness against varying network conditions.

While these works considered networks with no losses, where throughput benefits are achieved by simply coding at intermediate nodes in the network, they also assembled the core machinery that was subsequently used by future works, who evolved the field and showed the benefits of network coding in different applications.

## 2.2 Finite Fields

Since its introduction, network coding has expanded from simple XOR operations to more general combinations of packets. These operations involve digital computations carried out over finite fields. A finite field, also known as Galois field (GF), is an algebraic structure with a finite number of elements, and well defined addition ($+$) and multiplication ($\cdot$) operations, that obey the traditional rules of arithmetic. As a consequence, methods for solving linear systems of equations, such as Gauss-Jordan elimination, are applicable to finite fields. An appealing property of finite fields in the context of digital systems is their closure under addition and multiplication, i.e., for any given finite field $\mathbb{F}$ and all $a, b \in \mathbb{F}$, we have that $a + b \in \mathbb{F}$ and $a \cdot b \in \mathbb{F}$. This means that the outcome of an operation is guaranteed to be of the same size of the operands, and thus can be represented using the same number of bits. The number of elements in a finite field is known as its order, also referred to as field size in NC terminology. Only finite fields whose order is a prime power exist, and there exists exactly one finite field for each prime power. The finite field of order $p^n$, where $p$ is a prime and $n$ is a positive integer, is denoted as $GF(p^n)$. In the particular case where $p = 2$, these are called binary fields, and are particularly relevant in digital systems as they allow for efficient implementation.

## 2.3 Linear Network Coding

Network coding is a technique based on the combination of information. From this basic principle, one can imagine an arbitrary number of ways to perform such combinations with different levels of complexity. But the most commonly used code construction is simple and based on linear operations. In linear network coding, coded packets are formed as linear combinations of some set of original packets. As shown by Li *et al.* [4], such constructions are sufficient to achieve the multicast capacity.

Indeed, linear network codes are extremely appealing as they ally powerful results with simplicity. For that reason, our work is itself entirely reliant on linear codes. In this section, we review the basic machinery behind linear network coding, from the perspective of the different roles played by nodes in a network.

### 2.3.1 Source

Let $\mathbf{x} = [x_1, \ldots, x_k]$ be the set of *original packets*, also known as source block. A coded packet $y$ can be generated as

$$y = \sum_{j=1}^{k} g_j x_j. \tag{2.1}$$

The coding coefficients can be represented as a coding vector $\mathbf{g} = [g_1, \ldots, g_k]$ that describes the composition of the coded packet in terms of the original packets. In packet erasure networks, a transmitting node typically generates and transmits $n = k + r$ packets, where $r$ is some number of redundant packets to cope with losses. Through different choices of coefficients, deterministic or random, a node can generate a potentially infinite number of coded packets. Hence, since the code rate $k/n$ is not fixed, these codes fall within the class of *rateless codes*. It is worth noting that coded packets are nothing more than mixtures of the original packets so no new information is created. Therefore, the number of linearly independent packets will always be limited to $k$, the size of the original information. As will become clear, there is a big advantage in sending such mixtures rather than repeating the original packets.

### 2.3.2 Sink

On the receiving side, and upon receiving a set of coded packets $\mathbf{y} = [y_1, \ldots, y_m]$ along with the corresponding coding vectors $\mathbf{g}_1, \ldots, \mathbf{g}_m$, a receiver can establish the following relationship

$$\mathbf{y} = \begin{bmatrix} g_{11} & \cdots & g_{1k} \\ \vdots & \ddots & \vdots \\ g_{m1} & \cdots & g_{mk} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_m \end{bmatrix} \mathbf{x} = \mathbf{G}\mathbf{x}. \tag{2.2}$$

From a receiver's perspective, coded packets can be seen as equations which when put together form a linear system that can be solved for the set of original packets as

$$\mathbf{x} = \mathbf{G}^{-1}\mathbf{y}. \tag{2.3}$$

From this last expression it is clear that a receiver is able to retrieve the original data $\mathbf{x}$ from the coded data $\mathbf{y}$ when the coefficient matrix $\mathbf{G}$ is invertible. Since there are $k$ unknowns (the original packets), this implies that the receiver must receive at least $k$ linearly independent combinations to decode the original information. In other words, it is sufficient to deliver any $k$ coded packets as long as they are linearly independent. This condition is much easier to satisfy than delivering every original packet at least once.

### 2.3.3 Relay

One key feature of network coding is recoding. Intermediate nodes in the network can generate new linear combinations from coded packets they receive, without decoding them. Let $\mathbf{y} = [y_1, \ldots, y_{k'}]$ be the set of coded packets received by some node, and $\mathbf{G}$ the corresponding set of global encoding vectors. As long as $k' \geq 2$, this node can generate new coded packets as

$$y' = \sum_{j=1}^{k'} v_j y_j. \tag{2.4}$$

with local encoding vector $\mathbf{v} = [v_1, \ldots, v_{k'}]$. An interesting property is that the new combination is also a combination of the original packets, as shown below

$$y' = \sum_{j=1}^{k'} v_j \sum_{i=1}^{k} g_{ij} x_i = \sum_{i=1}^{k} \left( \sum_{j=1}^{k'} v_j g_{ij} \right) x_i. \tag{2.5}$$

The corresponding global encoding vector $\mathbf{g}' = [g'_1, \ldots, g'_k]$ has the coefficients given by

$$\mathbf{g}' = \mathbf{G}\mathbf{v}. \tag{2.6}$$

Recoding is then as simple as applying a linear transformation to both the coded data and global encoding vector. It can thus be said that network codes are composable. In packet erasure networks, this feature allows nodes to adjust redundancy on a hop-by-hop basis with minimum delay. This contrasts with other FEC schemes, which operate end-to-end restricting the creation of redundant packets to the source, or in a hop-by-hop manner requiring blocks of packets to be decoded before encoding again, incurring in additional delay.

## 2.4 Knowledge Representation

The algebraic framework of network coding described in the previous section lends itself to an elegant representation of the knowledge of a node, when regarded as a vector space defined by the coding vectors it has collected so far, called the *knowledge space*.

**Definition 1** (Knowledge Space)**.** *The knowledge space of a node is the vector space spanned by the coding vectors received by that node, i.e., span*$(\mathbf{g}_1, \ldots, \mathbf{g}_m)$.

From this perspective, the number of linearly independent packets received defines the dimension of the knowledge space, which is linked to the number of degrees of freedom in that space. Therefore, the following relationship holds.

$$\#\ \text{DEGREES OF FREEDOM} \equiv \#\ \text{LIN. INDEPENDENT PACKETS}$$

For this reason, the term *degree of freedom* is used in network coding literature to refer to a dimension in the corresponding knowledge space. In order to deliver a new degree of freedom and increase the dimension of the knowledge space, a packet has to be linearly independent from all previously received packets, in which case it is called an *innovative packet*.

**Definition 2** (Innovative Packet)**.** *A packet is said to be innovative with respect to a given receiver, if it is linearly independent from all previously received packets.*

These concepts are widely used in network coding literature for good reason, as they allow to describe the state of receivers in terms of the unintelligible coded information, and therefore are also used throughout this work.

## 2.5 Code Design

While the framework for constructing linear network codes is well defined, its operation depends on a set of design parameters that can be enumerated as follows.

- *Block Size:* The block size, denoted as $k$, corresponds to the number of packets in the source block, and therefore dictates the maximum number of packets that can be coded together.
- *Field Size:* The field size, denoted as $q$, corresponds to the number of elements in the finite field over which coding operations are performed, and therefore dictates the number of different combinations that can be generated.
- *Density:* The code density, denoted as $h$, corresponds to the ratio of non-zero coefficients in a code, and therefore provides a measure on the percentage of packets within in a block that are effectively coded together.

The settings for these parameters depend on the application requirements and are constrained by practical limitations. In this section, we elaborate on the main compromises that need to be taken into account, and provide some guidelines on code design supported by the existing literature.

### 2.5.1 Practical Limitations

Instead of delivering information about a single packet, coded packets deliver fractions of information about multiple packets. In particular, coded packets deliver information about the packets participating in their formation. From an information-theoretic perspective, it is therefore desirable to generate coded packets that are combinations of *all* original packets, and to have as much of those combinations independent from each other as possible. Theoretically, throughput performance increases with the block size, field size and density. In practice however, there are several aspects that push these parameters down.

- *Overhead:* In order to decode, receivers must possess the coding coefficients associated with the coded data. A common approach is to append the coding vector to the coded packet. In this case, the overhead corresponds to the size of the coding vector which takes $k\log_2(q)$ bits for representation. For sparse codes, where most of the coefficients are zero, it might compensate to append only the non-zero coefficients paired with corresponding packet index. In this case case, the average size is $hk\log_2(qk)$, where $h$ is the code density. In either case, it is clear that the overhead increases with the block and field sizes.

- *Complexity:* The complexity of coding operations scales with the block size $k$. Generating one coded packet requires $\mathcal{O}(k)$ operations, and decoding a block of $k$ packets involves the inversion of a $k$-by-$k$ matrix which entails $\mathcal{O}(k^3)$ operations. It is then clear that coding cannot be performed over arbitrarily large blocks of data. If sparse codes are used then inversion can be done with lower complexity, but sparsity is hard to maintain as packets are recoded over a network. In general, complexity also increases with the field size. A workaround to reduce computation time is to use lookup tables for the multiplication and/or addition operations in the chosen field. Although this solution speeds up computation, the storage required by such tables also scales with the field size. The effects of complexity affect senders and receivers differently. On the encoding side, high complexity may cause the *encoding throughput* – the maximum rate at which coded packets can be generated – to hinder or even become a bottleneck on the communication performance. On the decoding side, complexity is felt a bit differently since packets are typically decoded in blocks. It is possible for receivers to store coded packets and decode them when possible without stalling the transmission. Still, complexity should allow decoding to be done in feasible time.

- *Delay:* In general, receivers need to collect a whole block of packets in order to decode the information. This induces a *decoding delay* that is proportional to the block size. By dividing data in small blocks, or designing proper code structures, it is possible to achieve lower decoding delay at the cost of reduced throughput.

### 2.5.2 Design Guidelines

#### 2.5.2.1 Block Size

If the data to be transmitted consists of a large number of packets, one solution is to split the original data into smaller blocks known as *generations*. This approach is discussed in greater detail in Section 2.6.

#### 2.5.2.2 Field Size

In the context of network coding, the choice of the field is not a trivial one. Since coding coefficients are picked from the chosen finite field, the size of the latter dictates the number of different combinations that can be created. Furthermore, when coefficients are picked at random as in RLNC, the probability of generating linearly independent packets increases with the field size. In practice however, there are also factors that push the field size down, as already discussed. The field size is thus an important design parameter whose choice depends on several factors. The computation capabilities of each device must be weighted and field sizes should be chosen in a way that the coding throughput does not hinder the performance. Interestingly, Lucani *et al.* [12] have shown that, if the coefficients are chosen at random, the average number of coded packets needed to decode $k$ original packets is upper bounded by $k+2$, for any field size. This means that the performance gap between different fields decreases as the block size $k$ increases. In practice, most solutions range between $GF(2)$ and $GF(2^8)$, with the latter already providing a good compromise between performance, complexity and overhead. Perhaps the biggest challenge however, is that no-size-fits-all, especially given the very different capabilities of modern devices. Recently, Lucani *et al.* proposed a new class of codes, termed Fulcrum network codes [13], which combine codes with different field sizes in an attempt to merge the best of two worlds.

#### 2.5.2.3 Density

The typical approach to network code construction involves including all packets in a given block into the composition of every single coded packet, i.e., all non-zero coefficients. This greedy approach provides the best throughput possible but poses other problems that have been discussed.

**Sparse Codes** One of the drawbacks of RLNC compared with other error correction codes is complexity. Fountain codes, such as LT [14] and Raptor[15], achieve lower complexity through the use of a sparse code structure, at the cost of considerable overhead on the number of packets required to decode. For network coding, an interesting solution has been presented in [16] which proposes adjusting the code density as transmission progresses. The scheme, named tunable sparse network coding, starts with a sparse code setting and increases the density as transmission progresses. The rationale is that in the beginning receivers have few packets and therefore even sparse packets are innovative with very high probability. As the receivers accumulate more and more packets the probability of a coded packet being innovative decreases. To compensate for that, the code density is increased.

**Structured Codes** Rather than just defining a ratio of packets to be included in a given combination, it is possible to go beyond and specify which packets should be included. This could be driven by different reasons such as allowing for earlier partial decoding or delivering specific packets missing by a set of receivers.

**Systematic Codes** In systematic network coding, transmission has two phases. In the first phase, all source packets are sent uncoded. Since receivers still have not received any packets, it is guaranteed that all packets will be innovative. In the second phase, redundant coded packets are sent to recover from eventual packet losses, using RLNC. This systematic mode can be seen as a special case of tunable sparse NC, with two extreme density settings.

## 2.6 Coding over Generations

Many applications require transmission of large amounts of data over a network. Consider a large file consisting of $M$ packets. From a theoretical perspective, the most efficient solution is to generate coded packets that are combinations of all $M$ packets. This would make it sufficient to deliver any $M$ linearly independent packets to complete the file transmission. In practice however, coding over a large number of packets becomes prohibitive in terms of complexity and overhead.

One solution to this problem is to divide the original data in smaller blocks called generations. Generations are subsets of original packets with linear combinations confined to packets within the same generation. The simplest approach to build generations is to divide the original data into $g = \lceil M/k \rceil$ disjoint blocks of size $k$. Each coded packet is then a linear combination of packets within some generation $g_i$ for $1 \leq i \leq g$. This partition changes the problem, no longer being sufficient to deliver any $M$ coded packets but rather to deliver $k$ coded packets for each of the $g$ generations.

In the absence of perfect information, it is necessary to decide from which generations to send coded packets. This problem has been first addressed in [17] who analyzed the random scheduling of disjoint generations, a tempting mode for distributed operation. The main problem is the uneven distribution of packets received per generation, with some generations receiving more packets than needed while others not receiving enough. In packet erasure networks, this can happen due to some generations experiencing more erasures than others. This motivated the concept of overlapping generations. Different overlapping strategies have been proposed such as grid schemes [18], head-to-toe [19], and random annex codes [20]. The idea behind all of them is to bind generations through a set of overlapping packets, such that generations that receive more packets can help to decode other generations.

## 2.7 Online Network Coding

Coding techniques are eximious at delivering information albeit in a non-intelligible form, and what matters in the end for applications is the original information that is hidden behind the code. In network coding, receivers need to receive a whole block in order to decode the information. The additional time until a node can access the original information is known as *decoding delay*. For some applications that just need data as whole, e.g., file transfer, this is not a problem. There are however important applications that can benefit from earlier decoding of packets. A good example is online streaming where packets need to be delivered in order, at a certain rate, and with a bounded delay.

### 2.7.1 Coding On-The-Fly

Coding over generations suggests that a sending node first needs to accumulate enough packets to compose a generation before encoding and transmitting. In terms of delay however, the best strategy is to transmit information as soon as possible. Hence, a better alternative is to generate and transmit coded packets from the packets that are available at a given time. In the same way, a receiver can perform Gauss-Jordan elimination operations as coded packets arrive, attempting to decode packets as soon as possible. This is known as *coding on-the-fly*. Naturally, it is not feasible to simply add packets to the code indefinitely. One way to cope with this is to keep the concept of generations, adding packets to the code only up to a certain point. The problem is that generations create a boundary over which packets cannot be mixed. In the absence of feedback from the receiver(s) that confirms the complete reception of a generation, this imposes a blind decision on whether to continue on the current generation or advance to the next one. Such decision has a strong impact in the context of delay sensitive applications such as streaming.

### 2.7.2 Sliding Window

The intrinsic limitations of the block-based operation motivated the development of a network coding variant known as *online network coding*, where coding is performed over a sliding window rather than fixed generations.

The first work on online network coding can be traced back to [21], where the authors presented powerful concepts to represent the knowledge of receivers. Upon delivering a coded packet, it is not possible to state that a particular original packet has been delivered. One simple solution is to mark an original packet as delivered when it has been decoded. That would mean storing and encoding all packets until they are decoded. As the authors of [21] have shown, that condition is not necessary for decoding. The basic concepts of the proposed approach are presented in the following.

Consider the arrival of packets at the source node, where the $i$-th packet that arrives is said to have index $i$. Then a *seen packet* is defined as follows.

**Definition 3** (Seen Packet[1])**.** *A receiver is said to have seen packet $p_i$ if it can compute a linear combination of the form $p_i + w$ where $w$ is a linear combination including only packets with indices greater than $i$.*

The concept of *seen packet* is a powerful one. If a packet $p_i$ has been seen by all receivers, then the source does not need to include it in further combinations. This property stems from the fact that even if receivers obtain only combinations of packets with index greater than $i$ they will eventually be able to compute $w$, and thus be able to decode $p_i$. Hence, although seen packets are not necessarily decoded, they can be considered delivered by the encoding node.

This interesting observation enables several things. First, it allows packets to be removed from the buffer even before they are decoded. Moreover, it also means that packets can not only be added to the code but also removed in an online fashion, with encoding operations behaving as a sliding window over the original stream of packets.

A relevant question is under what conditions is a new packet seen. The answer is linked to the algebraic interpretation of coding vectors. As demonstrated in [21], *the number of packets seen by a receiver is equal to the dimension of its knowledge space*. By definition, delivering an innovative packet increases the dimension of the knowledge space. It then follows that delivering an innovative packet causes a receiver to see a new packet. As such, the following equivalence holds.

$$\#\ \text{DEGREES OF FREEDOM} \equiv \#\ \text{LIN. INDEPENDENT PACKETS} \equiv \#\ \text{SEEN PACKETS}$$

From this, seen packets can be seen as another representation of the knowledge space of a node, that links dimensions of the knowledge space to original packets. As we have seen, this concept allows for better management of the transmission buffer and design of more dynamic codes.

## 2.8 Conclusion

In this chapter we introduced network coding and its most common forms of application. Some of these techniques will be the basic block to allow us addressing our thesis and fulfilling our claims. In particular, in Chapter 4, we use RLNC with disjoint generations for practical file transmissions over multiple wireless channels. We discuss design parameter aspects in the context of mobile devices. We propose a system architecture which includes a configurable generation scheduler and propose a specific scheduler that attempts to deliver each generation in order. In Chapter 5, we derive fundamental bounds for the number of coded packets that need to be transmitted to deliver a single block of packets with a given reliability over asymmetric erasure channels. In Chapter 6, we consider online coding,

[1] Originally stated in [21], this is not the only possible definition of seen packet. In Chapter 7 we use an alternative definition, proposed later by the same authors in [22], that has advantages in terms of keeping track of decoded packets. More generally, upon receiving an innovative packet, the receiver is able to see one new original packet included in the received combination.

with opportunistic transmissions over time-varying channels to balance queueing delay and energy consumption. Coding is used to ease the scheduling over different channels and serve packets faster in face of imperfect feedback. In Chapter 7, we analyze the decoding delay of individual packets in a streaming scenario with limited feedback.

# Chapter 3

# Multipath Communication over Heterogeneous Wireless Technologies

Different branches on multipath research have sprouted during the years. Our attention lies on a recent line of work that has been motivated by the coexistence of multiple wireless technologies, both at the network and device levels. This class of multipath problems is characterized by a strong component on wireless communications. But unlike other wireless multipath research branches, this one distinguishes itself by the presence of asymmetric link characteristics induced by the very own nature of each technology, and potential for link level multipath through simultaneous communication over multiple technologies. Such features make this set of problems unique in terms of challenges and potential.

Using multiple wireless communication interfaces as a complement to each other has drawn the attention of technological companies. Google for instance, has recently launched Project Fi, a system that dynamically switches between Wi-Fi or one of three different LTE networks in order to provide each user with the best possible connection at a given time and place. Although it just provides access through one interface at a time, it indicates that diversity may be one solution to improve user experience. In terms of simultaneous use of multiple interfaces, Samsung has launched an application named Download Booster that enables downloads using Wi-Fi and LTE together in its devices, whereas a technological start-up named Shoelace Wireless released a couple of similar solutions for Android platforms in general. These latter solutions showcase the throughput benefits of interface aggregation at the application layer. Still, communication over heterogeneous wireless interfaces is a rather recent topic lacking in-depth studies. Therefore, and despite the existence of these real-world demonstrations, there are still important challenges to overcome in order to efficiently coordinate multiple interfaces and seize their full potential.

Our work seeks to provide a comprehensive understanding on communications over multiple wireless links along with novel solutions to these problems. Typical solutions [23, 24] either (i) choose a single interface at a time, which limits the throughput of the system, (ii) split the traffic across interfaces, which introduces challenges for delivering packets in order and in a timely fashion, or (iii) repeat the same content over different interfaces to cope with packet losses, which is an inefficient and static mechanism to add redundancy. This work advocates that packet-level coding, e.g. network coding, is critical to achieve efficient and practical multipath scheduling mechanisms. To this contribute the ability of coding to (i) abstract from individual packets, eliminating the need to specify which packets are sent through each link, and to (ii) recover from packet errors, allowing to continuously deliver informative packets, even in the limited feedback scenarios typical in wireless. In the following, we shall review closely related works that share our view for coded multipath communications, how they address some of the main challenges, and how they relate with our work.

## 3.1 Multipath Transport Protocols

The central role played by the Internet continues to drive researchers to design compliant solutions with its underlying protocols and work on multipath transmission has been no exception. At the transport layer, TCP remains as the standard protocol but its original design lacks desirable features such as multihoming. This motivated other protocols to be proposed among which the Stream Control Transmission Protocol (SCTP) has gained particular relevance in the context of multipath transmission, precisely due to its native multihoming support. Although SCTP considers multihoming as a simple realibility mechanism where alternative paths are activated in case the primary one fails, effective data transmission over multiple paths has been subsequently enabled through extensions such as Concurrent Multipath Transfer over SCTP (CMT-SCTP) [25]. However, SCTP based protocols still face a major hindrance which lies in their incompatibility with existing systems. Alternatively, TCP extensions for multipath operation have been more recently proposed, composing what is known as Multipath TCP (MPTCP) [26]. Although both SCTP and TCP based multipath protocols are focus of research, TCP based solutions are more likely to become mainstream due to their easier integration with existing systems.

MPTCP is a rather recent extension that is still undergoing validation and improvement. An early evaluation of its performance has been carried out by Nguyen *et al.* [27], who experimentally measured the throughput performance of MPTCP in three different configurations: Ethernet with Ethernet, Ethernet with WiFi, and WiFi with 3G. One key observation is that while MPTCP over paths with similar characteristics clearly outperforms TCP, the same does not happen when the paths have very different characteristics, where MPTCP can actually perform worse than TCP. These insights reveal that intelligent scheduling mechanisms are crucial to efficiently utilize multiple interfaces. To do that, it is important to understand the reasons that can limit the performance.

A common requirement to transport layer protocols is reliable and ordered delivery of data. These services can severely affect performance and thus require effective mechanisms to support them. In particular, the ordered delivery requirement can lead to what is known as *head-of-line* blocking, which happens when a line of packets is stalled because the first packet is missing. When transmitting packets over a single path, this can occur if the head packet is lost. In multipath transmission, the problem is exacerbated as the head packet can also be delayed, due to the different characteristics of the paths over which packets are transmitted. Not only does this delay packets from being delivered to the upper layers but also limits throughput performance by holding the congestion window from advancing. Moreover, both lost and out-of-order packets can trigger false signs of congestion causing the window to be reduced and further reduce throughput.

One solution used in the literature that has shown great potential to mitigate these problems, and the one which we advocate for in this work, is to use network coding. The work by [28] marks the ingress of network coding into the conventional protocol stack. The proposed solution, designated as TCP/NC, requires minimal modifications to the stack by introducing a shim layer between the TCP and IP layers. The motivation behind TCP/NC is to mask losses from the TCP layer, providing significant performance benefits over lossy links such as wireless links. However, the feature that we would like to highlight here is that of having TCP acknowledge *degrees of freedom* rather than original packets. Indeed, this is the basis for most network coding based transport protocols, including multipath ones and the basic principle is illustrated in Figure 3.1. Although the receiver still needs to obtain enough *degrees of freedom* in order to decode and effectively deliver the data to the application, all coded packets are equally useful towards such objective and thus it does not matter over which path they arrive from. Hence, this simple yet powerful idea masks the aforementioned problems, making TCP based protocols more robust to losses and path heterogeneity.

Standing on this, we now review three proposals of multipath TCP using network coding whose overview is presented in Figure 3.2.

Li *et al.* [29] proposed a network coded assisted MPTCP protocol named NC-MPTCP (Figure 3.2a). The proposed protocol uses network coding over blocks of packets in one of the paths, typically the fastest one, so that coded packets arriving on that path can help to recover from packet losses and out-of-order arrivals. For each block of $K$ packets, the proposed scheduler calculates the number of coded packets $N_c$ and uncoded packets $N_u$ to send over the respective paths such that, (i) the expected number of packets delivered is greater than or equal to the block size, i.e., $\mathbf{E}[N_c] + \mathbf{E}[N_u] \geq K$, and (ii) the packets are distributed proportionally to the rates $R_c$ and $R_u$ of each path, i.e., $N_c R_c = N_u R_u$. Essentially, the first criterion addresses redundancy, whereas the second levels out the load on each path. The solution to this scheduling problem is given under the form of a closed-form expression for the case of two paths. Simulation results evidentiate the goodput gains of NC-MPTCP compared to MPTCP as the path asymmetries increase in terms of packet loss rate and round-trip time. Nonetheless, as the path asymmetries increase, the performance of NC-MPTCP converges to that of single-path TCP. We can thus conclude that, although NC-MPTCP prevents the slowest path from stalling the transmission, it does not coordinate transmissions well enough to fully utilize multiple asymmetric paths, thus leaving room for improvement.

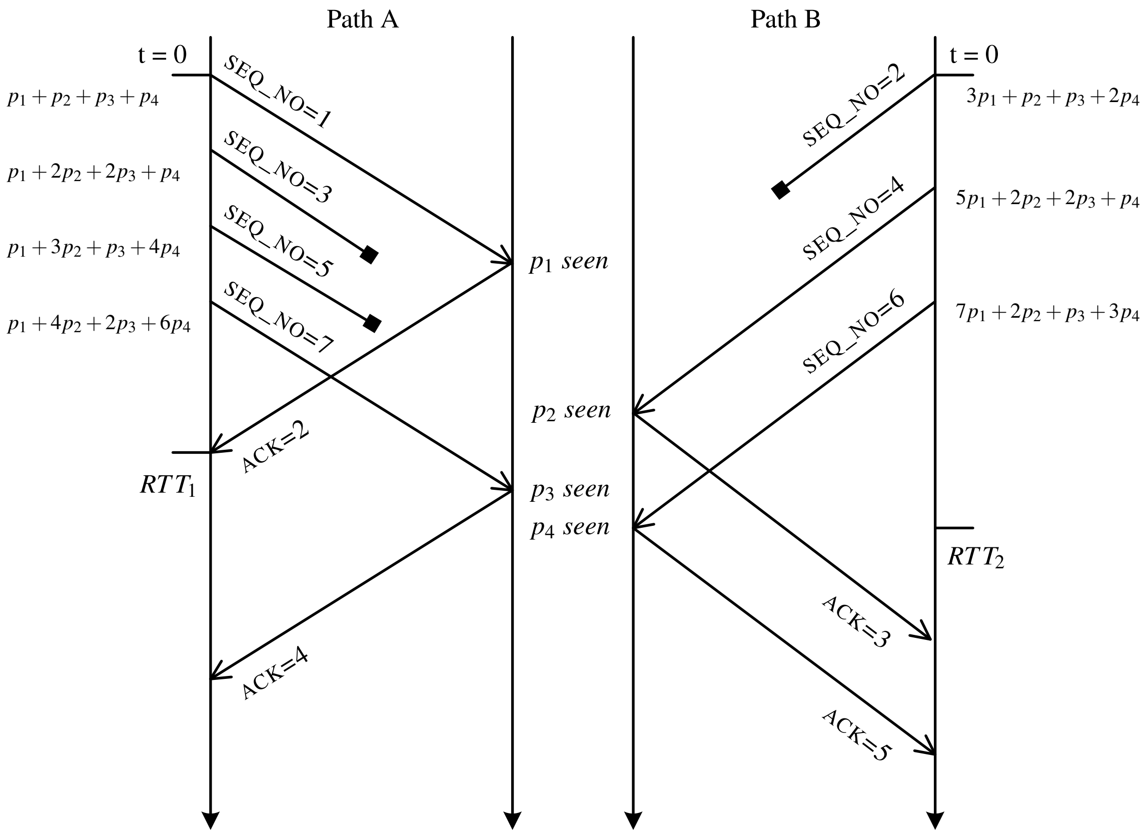
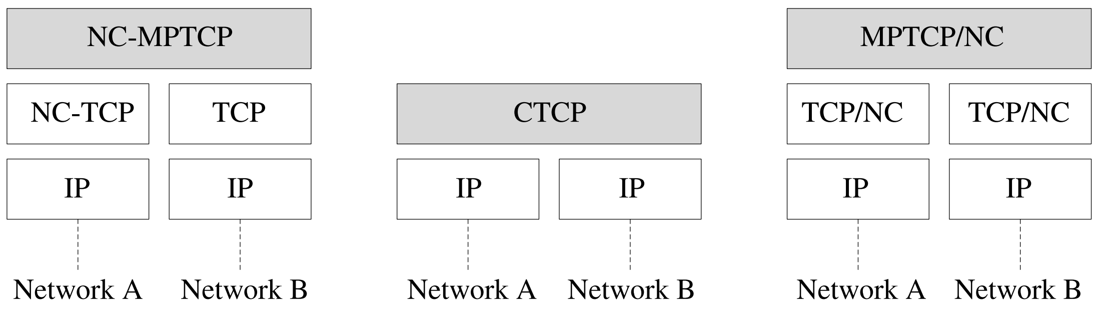


Figure 3.1: This temporal diagram illustrates the principle of operation of TCP/NC applied to a multipath scenario. The sequence numbers of the packets arriving to the receiver have gaps between them arising from losses and path asymmetries. Yet, it is quite interesting to observe that through the use of this coding technique, the receiver *sees* and acknowledges packets in order.

Kim *et al.* [30] proposed a coded variant of TCP, coined CTCP (Figure 3.2b), that is capable of multipath operation. Rather than building upon MPTCP as [29], the authors designed a new protocol meant to work on top of UDP. Once again, the authors employ coding over blocks of packets rather than a sliding window approach, arguing for bounded decoding delay and complexity. The proposed protocol features congestion control, network estimation, and multipath scheduling algorithms. The scheduler is executed before each packet transmission to decide from which block to transmit the packet. To that end, the scheduler iterates over the blocks in order and for each block it checks whether, in expectation, enough packets have been sent. If not, it transmits a packet from that block, otherwise it proceeds and checks the next block. Up to this point, the mechanism is rather straightforward, in that it transmits packets until the expected number of packets delivered is enough. But on top of this, the scheduler of CTCP adds an important feature. For the current block, i.e., the first unfinished block, the



(a) NC-MPTCP [29] (b) CTCP [30] (c) MPTCP/NC [31]

Figure 3.2: Overview of different coded multipath TCP solutions from a protocol stack perspective.

scheduler decides whether or not a path is suitable for sending a packet based on its RTT. This follows from the observation that, if $\mathrm{RTT}_1 \ll \mathrm{RTT}_2$, and a packet is transmitted over the second path and later another packet from the same block is transmitted over the first, then it may happen that the packet sent over the second path becomes redundant, lowering the throughput efficiency. Thus, the proposed scheduler attempts to detect these cases and, if the path is considered suitable, then a packet from the current block is transmitted, otherwise the path is used to transmit packets from the following blocks. Finally, the proposed protocol is evaluated on a real testbed with two WiFi interfaces at the client. The results show that CTCP clearly outperforms traditional TCP over single and multiple paths. However, the evaluation lacks results for heterogeneous interfaces or asymmetric path conditions. Moreover, the results reveal that when operating over multiple paths, the individual path performance degrades as the loss rates increase when compared to single path operation. As the authors themselves note, this decrease in performance is most likely due to the simple multipath scheduling algorithm employed, which allocates traffic without too much redundancy, once again suggesting that better scheduling mechanisms may be beneficial.

More recently, Cloud *et al.* [31] proposed MPTCP/NC (Figure 3.2c), a multipath protocol that adds a double layer of network coding to MPTCP. The lower layer works below each TCP connection in the same way as TCP/NC [28] and its purpose is to fight packet losses occurring in each individual flow, whereas the upper layer works on top of MPTCP to simplify the scheduling of packets across multiple paths. Differently from [29, 30], both coding layers operate over a sliding window rather than blocks. Moreover, coding is performed over all unacknowledged packets, thus ensuring that every packet will be innovative. On the one hand, this design circumvents the challenges associated with scheduling of blocks, i.e., sending the right amount of redundancy for each block and appropriately splitting the data across multiple paths. On the other hand, since the coding window can grow indefinitely before decoding can occur, it raises concerns in terms of delay and complexity. Another valuable contribution of this work is the experimental characterization of different radio access technologies (Iridium, WiFi and WiMax), in terms of their round-trip time and packet erasure probability distributions. This empirical data is used to generate the parameters fed to the analytical model, lending the theoretical results some realism. As expected, comparison of the theoretical throughput of MPTCP and MPTCP/NC, shows the better performance of the coded solution in terms of throughput.

The above works constitute recent efforts on multipath protocols motivated by the increasing number of multihomed devices. These works address common and important challenges that arise from the typical services provided by real-world protocols. As illustrated by these works, most of these challenges can be circumvented with network coding techniques. It is fair to point out that coding mostly masks some of the problems. By coding packets together, a decoding delay is introduced that corresponds to the time since coded packets arrive until they can be decoded and effectively used, which can be seen as a different kind of buffering delay. Still, it has great merit in boosting the performance of standard multipath protocols without major modifications to them.

## 3.2 Wireless Multihoming

A lot of the interest on multihoming arises in the context of heterogeneous networks and from the existence of multiple wireless interfaces available in modern devices. The works presented in the previous section incorporate some aspects of wireless communications, such as packet losses, but overlook other important ones, such as half-duplex communication, limited feedback information, and energy consumption. Even in terms of packets losses, wireless channels often exhibit bursty error patterns which are not captured by simple packet erasure channel models. Altogether, communicating over paths with asymmetric characteristics poses considerable challenges, which further accentuate when the considered paths include wireless links. Therefore, it is important to model these scenarios and propose multipath mechanisms suited for wireless communication. In this section, we review recent efforts that have been carried out in this direction, with focus on network coding enabled solutions.

Capela *et al.* [32] addressed the problem of minimizing the transmission delay of a flow of packets to a single multihomed user in the presence of contending traffic. The problem is approached from a queueing perspective with focus on the wireless component, where each access point is modeled as a M/M/1/K system. The service model for each server is based on the underlying wireless technology and error correction mechanisms considered. In this particular work, the authors provide coded and uncoded versions for the transmission models of WiFi and HSPA technologies. For the uncoded model, the standard retransmission mechanisms of each technology are considered, whereas for the coded model these retransmissions are disabled and packet-level coding is used to fight packet losses instead. The arrivals are modeled as a Poisson process, where the mean arrival rate on each server includes the contending traffic plus the fraction of the multihomed traffic traversing the associated path. The goal is to find the optimal allocation for the multihomed traffic such that the mean packet delay is minimized. To that end, the authors formulate a non-linear optimization problem, propose heuristic search techniques to look for optimal solutions, and evaluate them on a real testbed. As expected, the optimal solution typically involves sending traffic over multiple gateways, leading to lower mean packet transmission delay. Moreover, coded mechanisms are shown be more efficient than retransmission based ones, possibly leading to the same or better performance while using less access points. Overall, these results illustrate the benefits of multihoming, coding, and the synergy between the two, opening good prospects for the use of these two techniques. But there are also points that could benefit from further investigation and improvement. In terms of coding, a generation based coding is performed separately in each access point to fight packet losses over the respective link. In this way, the generations sent over each link are disjoint and thus do not help each other decode, leading to higher decoding delay. Also, such pure block based coding may not be the best solution in terms of per-packet delay and buffer occupancy. In terms of channel modeling, and despite the realism of the proposed models, these do not account for channel condition variations typical in mobility scenarios, which may be an important factor for load allocation and per-packet delay, nor does it take into account the effects of limited feedback.

Energy consumption is another relevant subject in wireless communications. Although network coding can itself lead to energy savings by reducing the number of transmissions needed to rebliably transmit data, employing opportunistic transmission strategies can provide further benefits by transmitting preferably under good channel conditions. This idea has been exploited by Pereira *et al.* [33] for transmitting a flow of packets with minimum channel utilization. The authors considered a setup where two nodes communicate with each other over multiple wireless links. Each link is modeled as a finite-state channel model with two states, *good* and *bad*, that seek to capture the time-varying nature of wireless channels. When a transmission opportunity is available, the source must decide for each link whether to send a packet or to wait for better channel conditions, at the cost of additional delay. The goal set forth in the this particular work is to find transmission policies that can serve the incoming flow of packets while keeping channel utilization to the bare minimum. By focusing on time-invariant policies that depend only on the channel state at a given time, the problem is formulated as a tractable linear programming problem and solved with exact methods. The results provided essentially show that dynamic schemes, that choose one interface or another depending on the current channel conditions, greatly outperform static ones in terms of channel utilization and energy consumption over low and medium load regimes. As the load approaches network capacity these gains diminish as both static and dynamic schemes must constantly transmit to satisfy the demand of incoming packets, and thus match in terms of energy performance. The core contribution of this work is indeed the optimization framework, which besides the transmission policies, outputs the value of its objective function that essentially provides a bound on the energy required to transmit a given flow of packets over multiple time-varying channels. Despite these contributions important to understand this class of problems, this work does lie on several unpractical premises. First of all, the work does not address delay performance which is typically important in real systems. According to the proposed formulation, packet delay can grow uncontrolled as long as all packets are served. Then, the formulation is based on the assumption that channel state information (CSI) is always available to make decisions, which is not feasible in real systems, as is not perfect feedback. These assumptions need to be relaxed in order to come closer to a practical solution.

## 3.3 Discussion

A recent body of research has focused on multipath communication enabled by wireless multihomed devices. Our analysis above discussed different approaches to this family of problems, focusing on coding based solutions.

Many recent solutions have been developed around MPTCP due to its practical appeal. In Chapter 4 we describe our take on a practical solution for file transmission over multiple wireless interfaces on mobile devices. From a design perspective, our work shares some traits with [30] as we also implement a new protocol meant to schedule blocks of coded packets over UDP. In our case, the primary goal was to demonstrate the viability of multipath solutions on mobile devices, in face of the limitations imposed. In particular, the limited computational power requires coding operations to be performed over smaller blocks, which hampers the scheduling process in face of uncertain path characteristics. For the same reason, scheduling algorithms cannot be made too complex at the risk of slowing down the transmission. Therefore, rather than stripping a block of packets across multiple paths like [30], our scheduling algorithm assigns different blocks to different paths, which is reasonable for the considered file transfer scenario. In return, we invest on an effective redundancy scheduling mechanism. Our scheduler tries to ensure that each block of packets is delivered with high probability to avoid that a block has to be rescheduled, thus increasing transmission efficiency. The result is an interface aggregation protocol that maintains a stable performance under heavy packet loss conditions.

The remainder of our work takes a more fundamental approach focused on the wireless component. While theoretical in nature, it addresses core challenges that underlie practical problems. Therefore, it is perhaps not surprising that some of the ideas found in the reviewed state-of-the-art protocols share some intuition with our own work. In Chapter 5 we investigate the scheduling of a block of packets over multiple asymmetric paths with half-duplex constraints, and propose transmission schemes to minimize the total transmission time. Our algorithm assigns traffic to each path such that, from a receiver's perspective, data reception from all paths finishes at the same time, therefore minimizing the total transmission time. Such intuition can also be found in NC-MPTCP [29], whose scheduler distributes the load among two paths according to the ratio between their data rates. While [29] develops expressions for two paths, ours supports any number of paths. Moreover, our scheduler takes into account the RTT of each path, in a similar fashion to the scheduler of CTCP [30], to avoid unnecessary packets to be transmitted over long latency paths. Then, while these and other schedulers fight packet losses by sending just the expected number of required packets, ours allows to adjust the amount of redundancy based on a reliability target. As our results show, such overprovisioning is essential to achieve near-optimal delay performance when there are costs associated with feedback reception and retransmission of missing packets.

In Chapter 6 we consider the transmission of a single flow of packets over multiple time-varying paths. In this setting, we consider an articulation of delay and energy performance under a theoretical framework that incorporates practical aspects, building upon existing works [32, 33]. The delay metric considered is similar to the one in [32], but our assumptions differ in several important aspects. First, we consider a time-varying channel model, a natural fit for wireless environments and one that has a big impact on packet delay performance. Secondly, we deal with imperfect feedback which poses increased challenges on the scheduling process. Finally, rather than considering delay as the only metric to optimize, we add energy consumption into the mixture, due to its matching relevance in wireless communication systems. This leads us into an opportunistic transmission problem similar to [33], where packet transmissions can be postponed to spare energy. Rather than blindly optimizing energy consumption like [33], we seek a compromise between both, while also relaxing the assumptions made in the referred work. In terms of practical application, our analysis could be useful to understand the delay behavior of real protocols such as MPTCP/NC [31], which use a similar coding approach, in the considered scenarios.

Finally, in Chapter 7 we perform an empirical analysis of packet decoding delay when using *online network codes* under limited feedback scenarios. As already discussed, coding operations add a layer of delay that corresponds to the time since a packet is received until it can decoded and becomes usable. While for certain applications it is sufficient to receive and use packets in batches, others benefit from receiving individual packets as early as possible. Therefore, it is important to understand the impact of such codes, and their structure, on delay performance, especially in limited information scenarios typical in wireless. This final chapter takes a shot at this problematic, closing the loop in terms of delay analysis.

Following this discussion, we now move on to the presentation of our work, which lies the foundations for the proposed thesis. Our take on multipath transmission does not focus on a specific protocol. Instead, we identify and tackle more fundamental problems, with focus on wireless links. With this, we expected that our insights and techniques to be valuable to a diverse number of situations where multiple wireless technologies are available.

# Chapter 4

# Coded Multipath on Android Devices

Mobile devices are one of the most iconic examples of multiple wireless technologies coexisting in a single device and, at the same time, embody the current communication paradigm which consists in establishing communication over a single interface. In order to showcase the benefits of aggregating multiple interfaces in a real use case, we developed an Android application for transferring files between two mobile devices over multiple wireless interfaces. These devices natively incorporate a set of wireless interfaces, e.g. WiFi, BT, 3G, NFC, that are more diverse than other platforms, hence the choice of platform is a natural one. However, mobile devices have reduced computation and memory resources thus creating a relevant set of challenges that need to be addressed.

Beyond interface aggregation, another goal is to showcase the benefits of network coding in a real application. While network coding has been shown to provide remarkable benefits in theory, it is necessary to translate those benefits into practice, hence its transition to the real world has also been a major topic of study [34]. In our case, we seek to exploit its potential to reduce the complexity of packet scheduling over multiple paths and provide an efficient communication over lossy wireless links.

Throughout this chapter, we present, discuss and evaluate our solution for the simultaneous use of multiple wireless interfaces together with network coding techniques to improve information throughput in a real scenario, before delving into more analytical frameworks.

**Problem Statement**

In this chapter, we seek to leverage multiple communication interfaces existing in mobile devices, together with network coding, to minimize the *file delay*.

**File Delay** (Figure 1.6): Time needed to transmit a large file, i.e. composed of many blocks of data, plus the time to receive the feedback acknowledging the end of the transmission.

**Contributions**

- *Implementation and Measurements with a practical use-case in Android Devices:* We present a system design and measurement results with Android mobile phones using various wireless interfaces and study protocol parameters to jointly improve throughput and delay.

**Publications**

- A. Moreira and D. E. Lucani, ”Coded Schemes for Asymmetric Wireless Interfaces: Theory and Practice,” *IEEE J. Sel. Areas Commun.*, vol. 33, no. 2, pp. 171–184, Feb. 2015.

## 4.1 Setup

Our setup consists of two LG Nexus 4 phones running Android 4.3, one acting as the sender and the other as the receiver. The devices have WiFi, Bluetooth, and 3G communication capabilities. The two devices can communicate simultaneously via a reliable Bluetooth channel and, either a WiFi channel with an access point in the middle or a 3G connection, as depicted in Figure 4.1.

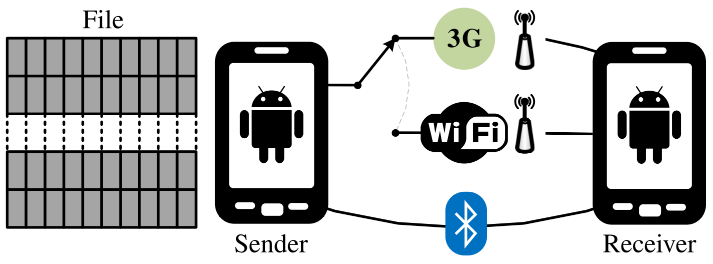


Figure 4.1: System setup for experimenting with our Android multipath file transfer application, trying to take advantage of the multiple interfaces available on mobile devices.

To better match the analysis developed throughout this thesis, we attempted to build a setup as close as possible to the models considered in our analysis. However, due to hardware, software and protocol constraints, some assumptions, e.g., constant transmission rates, cannot be guaranteed in practice. Below we enumerate the reasons that limited our setup.

- *WiFi Link:* For the WiFi link, we worked on top of UDP which provides an unreliable channel. However, the IEEE 802.11 standard includes the transmission of acknowledgments at the MAC layer, which masks much of the losses for the higher layers. Also, during our experiments, when moving the devices apart from each other we observed that link adaptation was being performed maintaining the losses low for the higher layers while decreasing the bit rate. In order to demonstrate the error correction potential of network coding, we generated synthetic packet losses by adding a rule to *iptables* to randomly drop incoming packets with some probability.

- *Bluetooth Link:* For the sake of simplicity, and because the Android framework only provides an API for the RFCOMM protocol, we worked on top of this protocol which also provides a reliable channel.

- *3G Link:* In this case we also worked on top of UDP. Since devices connected to a 3G network are in general hidden behind different sub-networks, NAT traversal techniques need to be applied in order to establish communications between them. In our case, we connected the devices to the same Virtual Private Network (VPN) using PPTP, thus providing a tunnel for the devices to communicate.

- *Simultaneous WiFi + 3G:* In Android, 3G data connections are disabled once WiFi is enabled. Despite being possible to recompile the Android OS in order to disable this behavior, we considered this option to be beyond the scope of our work. Thus,this combination of interfaces was not used.

- *Simultaneous WiFi + Bluetooth:* Most mobile devices come equipped with combined radio modules, where WiFi (2.4 GHz) and Bluetooth share the hardware. During our experiments, we found that simultaneous communication over WiFi (2.4 GHz) and Bluetooth was not possible at high speeds due to this hardware limitation since the module was either being used for WiFi or Bluetooth. Our solution was to switch WiFi operation to the 5 GHz band for which there is a separate hardware module.

- *Simultaneous 3G + Bluetooth:* For this combination, no particular limitation was found and therefore was considered in our experiments.

It is interesting to note that despite their potential for multipath communications, these devices are not prepared for that purpose as the simultaneous use of multiple interfaces has limitations that need to be circumvented.

## 4.2 Coding in Practice

In Chapter 2, we described the fundamentals of network coding and introduced its main design parameters namely the field size $q$ and the block size $k$. As explained back then, decoding has complexity $\mathcal{O}(k^3)$, hence the block size cannot be arbitrarily large. If the source data is too large, one solution is to split it into blocks of admissible size $k$ denoted in network coding terminology by generations, where $k$ becomes known as the generation size. Regarding the field size, the higher it is the more complex the coding operations in general. Moreover, to be able to decode, the destination node must know the coding coefficients used to generate each coded packet. A common approach is to build the coded packet by appending the encoding vector to the encoded data. The encoding vector contains the set of coefficients used to generate a coded packet and therefore requires $k\log_2(q)$ bits for representation. It is then clear that increasing both $k$ and $q$ increases packet overhead. In general, increasing the field size and generation size offers better network performance but also increases the complexity of coding operations and results in larger overheads due to coding coefficients representation. These performance issues are especially critical in the context of resource constrained devices such as mobile phones which have diminished processing power and limited energy. In the remainder of this section, we discuss suitable design parameters and some techniques to improve performance.

The right choice of the generation size is certainly dependent on the application. More generally though, it is important to be aware of acceptable ranges for the design parameters. In [35] a benchmark

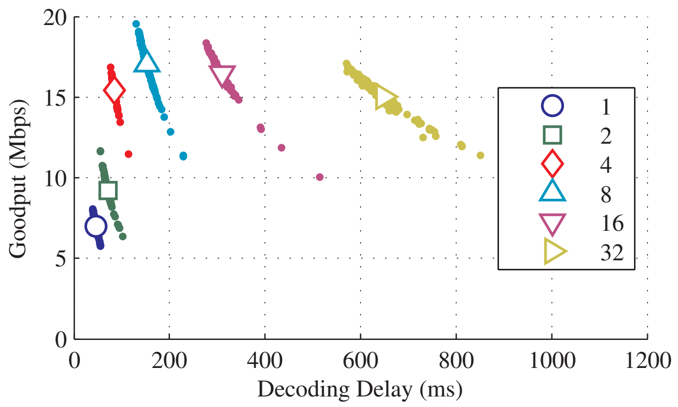


Figure 4.2: This plot shows the mean generation decoding delay against the mean goodput for different values of maximum active generations. Each data point represents a WiFi+BT transmission with 656 generations, generation size $k = 32$, field size $q = 2^8$, and symbol size 1300 bytes.

of coding operations on commercial mobile devices is presented. The tests are performed on different devices for several values of the field size and generation size. As an example, the average coding throughput for 64 packets per generation over $GF(2^8)$ running on a Samsung Galaxy S3 was $20.7 Mbps$. If we consider WiFi transmissions, this performance may not only weigh down on the overall throughput but can become a bottleneck if we recall that WiFi speeds can reach beyond this value in some cases. The presented results are for coding operations. In single-hop settings, performance can be easily improved by employing systematic coding. In systematic coding, each packet is sent once uncoded and only the following packets are coded. This has the advantage of speeding up coding operations, especially in case there are few losses since the number of coded packets required is small.

The design of coding systems is also influenced by the existence of transmission delays [34]. When the Round-Trip Time (RTT) is not negligible when compared to the transmission time of one generation, keeping only one generation active is not the best approach to achieve good throughput performance. One solution to improve network performance is to keep transmissions from multiple generations active. Due to the design of practical systems, letting the amount of data grow too much may also cause problems. An effect called bufferbloat occurs when the transmission link becomes a bottleneck causing an oversized buffer to be filled [36]. This results in turn in high latency and decreased throughput. In network coding, this may be triggered in different ways. For example, if there is no limit to the number of active generations or no control on the level of redundancy, then the amount of transmission data can be more than what the buffer can handle properly. Another drawback of keeping multiple generations active is the increase of the decoding delay. Although file transfer in general is not a delay critical task, there are some types of contents that may benefit from an ordered reception of the data. One example are images encoded using progressive JPEG, which can be decoded in multiple stages. Another example is the possibility of previewing stream data, such as video or audio, while it is being downloaded. In order to attenuate these effects, the number of active generations should be limited. To backup these statements, we present in Figure 4.2 experimental results that show the goodput and decoding delay for different maximum active generations.

## 4.3 Multipath Protocol

In this section, we present a multipath transmission protocol that combines the insights presented later in the analytical chapters of our work with practical considerations. The proposed protocol aims to be flexible rather than optimized for a specific setting in order to allow different experiments to be performed. Our goal is to present a high-level overview of the implemented solution whose architecture is depicted in Figure 4.3.

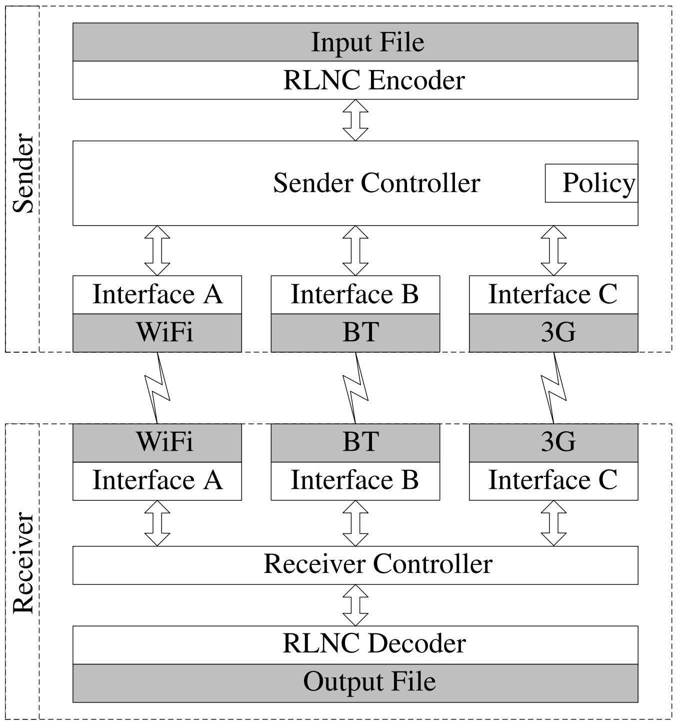


Figure 4.3: Architecture of the implemented protocol. A main feature of this architecture is the modularity which allows for an easy extension or modification of specific parts of the protocol.

- *Interface:* The *Interface* is a software abstraction that represents a channel for sending and receiving data. Each *Interface* component exposes a set of methods that hide implementation details and allow the protocol to be agnostic to the underlying channel, thus simplifying protocol design and enabling an easy integration of new communication interfaces. Sender and receiver communicate through one or more of these interfaces.

- *Sender Controller:* Coordination among interfaces in the sender is performed by the *Sender Controller*, which acts as an intermediary between the different channels and the data source. Its main purpose is to schedule transmissions across different channels and manage feedback information. There are several operations that are performed by the controller within this process as enumerated below.
  - *Error Rate Estimation:* Whenever a coded packet is sent through one of the interfaces, the controller updates the number of packets sent through that channel for the respective generation. When feedback is requested for a given generation, the number of packets sent is attached to the request packet and in turn the receiver appends the number of packets received to that information in the feedback packet, thus providing the sender with a sample of the error rate within the feedback packet.
  - *Generation Delivery Estimation:* The channel error rate estimation is crucial for another feature of our protocol which consists on estimating the probability of a generation being finished. To perform this task, the controller estimates the mean and variance of the number of packets delivered to the receiver based on the channel error estimate at each transmission, which it uses to calculate the probability of having delivered enough packets to complete a generation[1]. Based on this information it decides whether to send more packets or request feedback. Although feedback is used, guaranteeing beforehand that a generation is delivered with high probability, reduces the number of times that generations have to be rescheduled. This reduces the amount of control information that needs to be exchanged and other inefficiencies associated with the retransmission.
  - *Congestion Control:* Finally, the controller limits the number of active generations an interface may handle at the same time. If there are no more packets to be sent among the active generations then feedback must be received in order to proceed with the transmission. This is done in order to attenuate the effects of congestion, as described in Section 4.2.
- *Policy:* Within the set of all generations, the controller needs to decide which generations to send first. This decision depends on performance objectives such as to maximize throughput or to minimize decoding delay, and the transmission state at the moment of the decision. The transmission state consists of generations that have already started, generations that are only waiting for feedback and so on. Upon request by one of the interfaces, the controller makes its decision according to a transmission *Policy*, a set of rules that schedules the different parts of date to the different interfaces, and whose logic is decoupled from the remaining components to allow for easy modification. In Algorithm 1, we present the policy used in our experiments. The proposed policy primarily attempts to reduce the delivery delay of each generation by prioritizing

[1]This calculation is addressed in more detail in Chapter 5 where we investigate the transmission of a single block of packets.

generations that have already been started (Rule 1). However, it does not compromise throughput as it prioritizes data transmission (Rule 2) over feedback request repetition (Rule 3). Finally, if none of the previous rules matches, then an interface may transmit packets from generations assigned to other interfaces (Rule 4). One limitation of the proposed policy however, is that it tries to separate generations across interfaces (Rules 1-3) only assigning packets from one generation to multiple interfaces if there is no other option (Rule 4). This implies that the interfaces will not be collaborating all of the time. Although this strategy has no impact on the overall throughput it also does not seize all the delay benefits achievable per generation by splitting the transmission of each generation across the multiple channels. This decision was driven by the fact that this coordination is more complex for small $k$ in highly asymmetric and changing conditions, and does not provide as much flexibility for trading off throughput and delay.

- *Encoder:* The *Encoder* is the component that generates coded packets from the source data using RLNC. The encoder parameters are the maximum packet size, maximum generation size and field size. The maximum packet size and maximum generation size are used by the encoder to divide the file into generations. The partitioning scheme used by the encoder is implemented in the Kodo C++ library [37] and defined in [38]. The aforementioned algorithm computes a partitioning of the file so that the resulting blocks are as equally sized as possible. On the other end of the system, the *Decoder* decodes the received coded packets and reassembles the decoded generations into a file. The parameters set in the encoder are communicated to the decoder during the initialization phase of the communication.

- *Receiver Controller:* Finally, the *Receiver Controller* simply keeps the number of packets received for each generation and interface.

**Algorithm 1** Delay-Throughput Policy

```
Upon request by interface $l$, pick the first match:
    Rule 1: Transmit packets from earliest generation started by interface $l$ and for which there are packets to be sent, i.e., it is not waiting for feedback.
    Rule 2: Start a new generation, provided that the number of maximum active generations is not exceeded, and assign it to interface $l$.
    Rule 3: Request feedback for earliest generation assigned to interface $l$.
    Rule 4: Assign packets from any generation even if assigned to another interface.
```

Our design and policies integrate some of the theoretical results and insights presented throughout this work, shaping them to the practical limitations faced, thus combining theory and practice.

## 4.4 Experimental Results

We now present experimental results obtained using our setup. Network coding operations are performed by the Kodo [37] library. The settings used are generation size $k = 32$ and field size $q = 2^8$ and systematic coding on. The number of active generations per channel is limited to 8. The packet size is $W = h + w + v$ where $h = 5$ is the header size, $w = 1300$ is the data size and $v = 32$ is the size of the coding vector. All experiments are performed using the policy in Algorithm 1. Our results seek to show:

- The improved throughput and robustness brought by multipath communications
- The gains provided by network coding when there are errors in the channels

To exhibit the benefits of multipath communications, we setup a scenario where two users exchange a file between their mobile devices. We then ran two practical use cases that combine different interfaces. In the case of Figure 4.4, the users are connected to a 3G cellular network and are able to maintain a Bluetooth connection. For this case where the channels possess similar rates, the aggregated throughput is twice of what would be achieved if transmission would be accomplished through a single interface.

In the case of Figure 4.5 the users are still able to maintain a Bluetooth connection but instead of a 3G link, they are now connected through an WiFi access point. Here we simulated a common scenario where the users may be moving. At the beginning of the transmission, the users are close to an access point and therefore the quality of the WiFi channel is very good, making the throughput contribution of the Bluetooth channel relatively small. As the users move away together from the access point, the quality of the WiFi channel drops and that of Bluetooth remains stable, thus increasing its relative contribution. This scenario serves not only to show throughput benefits but also that multipath settings

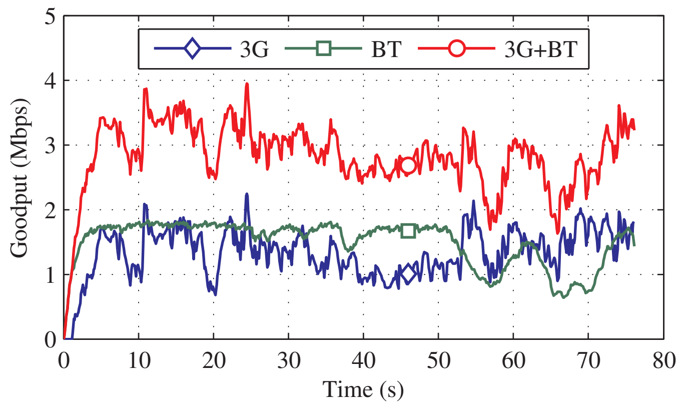


Figure 4.4: Multipath transmission of a 27 MB file over 3G and BT. This example shows that when the individual speeds of each channel are similar, the throughput gain achieved by bundling $L$ channels can be $L$-fold.

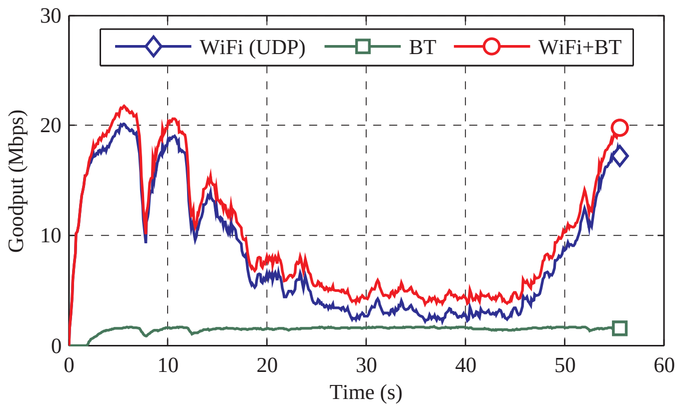


Figure 4.5: Multipath transmission of a 70 MB file over WiFi (UDP) and BT. The two end devices are always close to each other but move away from the WiFi access point. Robustness increases as the connection would be maintained even if one of the channels would break.

can provide robustness by solving the single point of failure problem in terms of communications channels.

To showcase the benefits of network coding in the presence of losses, we compare the performance of our Coded/UDP protocol against an Uncoded/TCP protocol over different packet loss rates on the WiFi channel. The Uncoded/TCP protocol uses our multipath architecture to split generations across the TCP/WiFi and BT interfaces but disabling all error recovery mechanisms such as transmission of redundant packets and feedback requests. To simulate errors in a controlled manner, we inject packet losses on the WiFi channel by adding an entry in the *iptables* of the receiver device to randomly drop incoming packets with a fixed probability. The results for a single transmission are shown in Figure 4.6 for Coded/UDP and Figure 4.7 for Uncoded/TCP, where the packet erasure rate was fixed at 5%. The results clearly show that our coded protocol is able to outperform uncoded transmission over TCP, providing gains in throughput of nearly 4 times. To reinforce these results, we further evaluated both protocols over multiple runs and different packet erasure rates. The results are shown in Figure 4.8. For the selected range of error rates our Coded/UDP protocol is able to maintain a steady performance whereas Uncoded/TCP abruptly looses performance. These results confirm that the use of protocols that can naturally exploit multiple paths and manage losses is critical to deliver high performance.

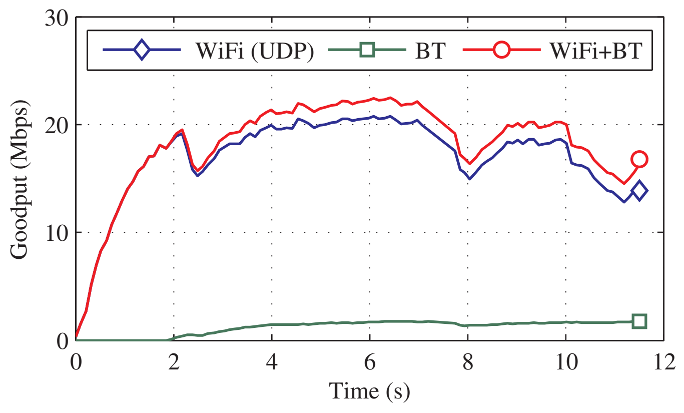


Figure 4.6: Multipath transmission of a 27 MB file over BT and WiFi (UDP) with 5% injected losses using coded transmissions. The performance remains comparable to the case with no injected losses.

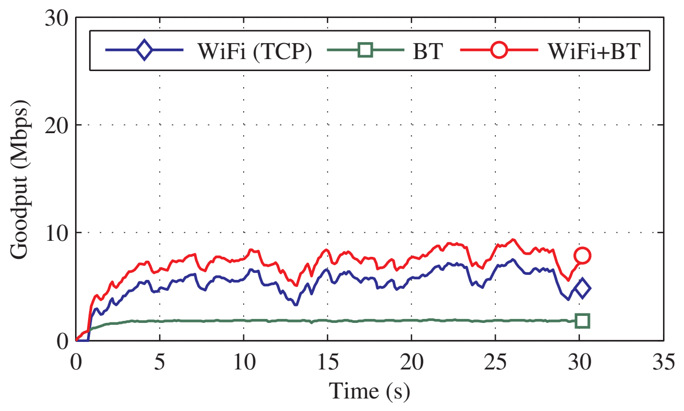


Figure 4.7: Multipath transmission of a 27 MB file over BT and WiFi (TCP) with 5% injected losses using uncoded transmissions. The performance is severely affected despite the small PER.

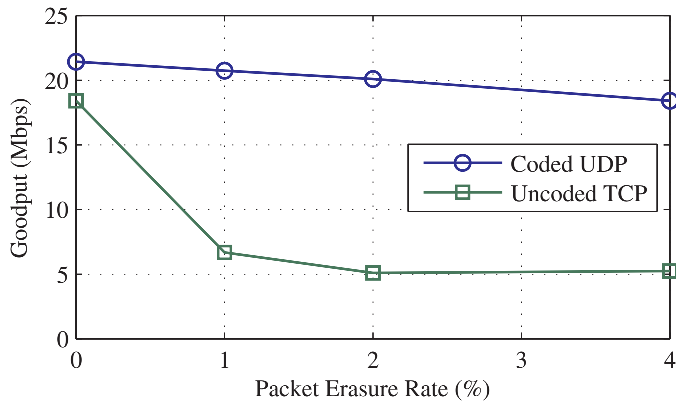


Figure 4.8: Performance comparison of the proposed network coding protocol against uncoded transmissions over TCP for different packet erasure rates injected on the WiFi channel. Each point corresponds to the mean of 10 transmissions of a 70 MB file. The results clearly show the remarkable gains achieved with network coding.

## 4.5 Conclusions

This chapter presented implementation and measurement results in commercial devices to better understand the problem of transmitting data over multiple asymmetric half-duplex channels. Our discussion also emphasized a variety of challenges to effectively trade-off delay and throughput. Nonetheless, our implementation was capable of aggregating the throughput of very heterogeneous wireless interfaces, as expected from a flow perspective, as well as providing a reliable protocol in the presence of losses and varying channel conditions.

Still, splitting the transmission of individual blocks across multiple asymmetric interfaces turned out to be a challenging problem, especially for the small block sizes considered. While this aspect did not have relevance in the considered application, it may have in others that benefit from a more timely delivery of individual blocks. Therefore, it is worthwhile to understand this problem, and come up with efficient solutions for delivering individual blocks across asymmetric interface in a more timely manner, which is the topic of the next chapter.

# Chapter 5

# Coded Multipath for Asymmetric Wireless Interfaces

A timely transmission of data packets to a destination over heterogeneous, half-duplex wireless channels presents unique challenges due to inherent asymmetry in their round-trip time, packet losses, and transmission rate. Coding across data packets and performing a judicious usage of each communication channel is at the heart of achieving a timely transmission, with important consequences to providing convergence services and efficient multi-homing. Still, a fundamental understanding of the benefits of coding over multiple wireless interfaces with half-duplex constraints has been missing. Specifically, the problem of how to schedule the transmission of a block of coded packets as well as which channel and when to stop transmitting to listen for feedback has not been studied.

This chapter analyzes the transmission of a block of packets over asymmetric channels. It derives fundamental limits and practical mechanisms that jointly optimize feedback and coding with respect to the time to complete the block transmission.

**Problem Statement**

In this chapter, we seek mechanisms to split the transmission of a block packets among multiple half-duplex channels with different characteristics in order to minimize the *block delay*.

**Block Delay** (Figure 1.6): Time needed to transmit a single block of packets and receive feedback acknowledging the end of the transmission.

**Contributions**

- *Concept of Capacity in Multiple Constrained Routes:* We propose a metric for measuring the capacity of a session using multiple transmission paths given a reliability constraint and maximum number of allowed transmissions per path. We derive closed-form bounds for this expression based on probability inequalities.

- *Mathematical Model:* We develop a framework based on a Markov model to characterize a wide array of coded and uncoded schemes. The model can analyze channels with different round-trip delays, packet losses, and raw transmission rate per packet.

- *Joint Optimization of Feedback and Coding:* We propose a variety of delay optimal and suboptimal schemes that choose how many transmissions to allocate in each half-duplex channel as well as when and which ones to stop for receiving feedback.

- *Numerical Analysis:* We present numerical results comparing a variety of uncoded scheduling techniques, our proposed coding mechanisms, and a lower bound on delay based on a system with full duplex channels. Our results show that our half-duplex mechanisms are within 1 dB of the lower bound over a wide range of operating conditions that reflect real-world scenarios. They also show that the uncoded approaches not only have a worse performance than our proposed solutions, but that the choice of uncoded technique is highly dependent on system conditions.

**Publications**

- A. Moreira and D. E. Lucani, “On coding for asymmetric wireless interfaces,” in *Proc. IEEE Int. Symp. Network Coding*, Cambridge, MA, June 2012, pp. 149–154.

- A. Moreira and D. E. Lucani, “Coded Schemes for Asymmetric Wireless Interfaces: Theory and Practice,” *IEEE J. Sel. Areas Commun.*, vol. 33, no. 2, pp. 171–184, Feb. 2015.

## 5.1 System Model

### 5.1.1 Network Model

We consider a unicast session between two nodes, where the source node (sender) has a block of $K$ packets in its buffer that must be sent to the destination node (receiver). The two nodes are connected by $L$ half-duplex links. We model each link $l$ as a packet erasure channel, with packet erasure probability $P_e^{(l)}$. We assume that each channel $l$ has a fixed transmission rate of $r^{(l)}$ [bps]. In our treatment of the problem, we adopt a time-slotted system, where time is divided into equally sized time-slots each with length of $T_{slot}$. Fixing the slot length $T_{slot}$ and the packet size $W$ in bytes, the transmission rate can be translated into an equivalent quantity $T_t^{(l)}$ (Equation 5.1) that represents the number of time slots that channel $l$ takes to transmit one packet.

$$T_t^{(l)} = \left\lceil \frac{8 \times W}{r^{(l)} \times T_{slot}} \right\rceil \text{ (slots)}, \tag{5.1}$$

This allows us to model channels with different transmission rates. Equation 5.1 implicitly assumes that all packets have the same size. Moreover, it uses the ceiling operator because we use the slot as our time unit and measure time in integer number of slots, yielding a conservative approximation. Although this induces an approximation error, the slot size can be made as small as needed to obtain an acceptable accuracy level. Besides the transmission time per packet, we also consider the channels to have a non-negligible propagation time $T_p^{(l)}$ that corresponds to the time a packet takes to travel from the source to the destination. A representative diagram is presented in Figure 5.1.

Feedback is used to track the progress of the transmission. The transmission is considered complete when feedback reports that no more packets are needed. Because channels are half-duplex the sender must stop at least one channel in order to receive feedback. We assume that the feedback packet size is small enough such that its transmission time and loss probability are both negligible. Nonetheless, we

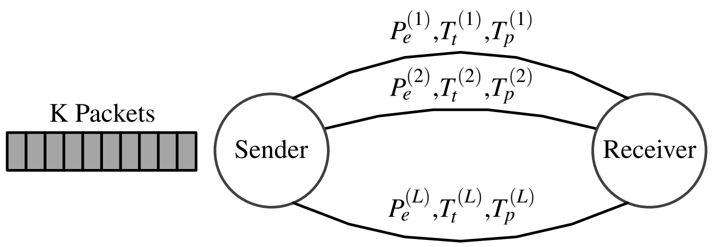

$P_e^{(1)}, T_t^{(1)}, T_p^{(1)}$
$P_e^{(2)}, T_t^{(2)}, T_p^{(2)}$
K Packets
Sender
Receiver
$P_e^{(L)}, T_t^{(L)}, T_p^{(L)}$

Figure 5.1: Network topology with two nodes where $K$ packets must be conveyed from the sender to the receiver. The two nodes are connected by $L$ links mainly characterized by three parameters, namely packet erasure rate $P_e$, packet transmission time $T_t$ and propagation time $T_p$. This topology can represent diverse scenarios from mobile phones communicating over multiple interfaces to multibeam satellite systems and multihomed devices in general.

assume that the packet still suffers the propagation delay relative to the channel through which it is transmitted. Thus, for a feedback packet we have $P_e^{(f)} = 0$, $T_t^{(f)} = 0$ and $T_p^{(f)} = T_p^{(l)}$, where $l$ is the channel used to send the feedback packet.

### 5.1.2 Coding

We assume that Random Linear Network Coding (RLNC) [11] is performed at the source node operating over the block of $K$ packets. In RLNC, the coefficients used to generate coded packets are chosen at random from the chosen finite field. Consequently, the probability of generating linearly dependent packets decreases as the field size increases. In our analysis, we shall consider a field size large enough such that coded packets are linearly independent with very high probability.

### 5.1.3 Transmission Model

In this chapter we focus on the time to complete the transmission of the block of $K$ packets. This comprises the time between the start of transmission and the completion feedback being received at the source node. In order to evaluate different transmission schemes in terms of the expected completion time, we propose a framework to model the transmission accurately. We attempt at providing upfront the expressions necessary to define the framework while maintaining it general enough to be applicable to different schemes.

We model the progress of the transmission as an absorbing Markov chain where each state represents the state of missing packets at the receiver, as illustrated in Figure 5.2. The considered the Markov

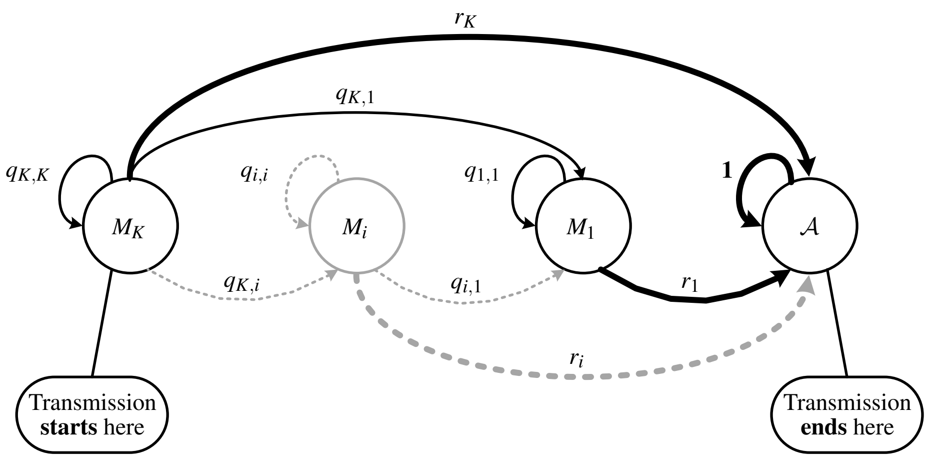


Figure 5.2: Absorbing Markov chain used to model the transmission process. Transmission starts in state $M_K$ and ends in the absorbing state $\mathcal{A}$. Thick lines highlight transitions from each state to the absorbing state.

chain has states $\mathcal{M} = \{M_1, \dots, M_K, \mathcal{A}\}$ that represent the number of innovative packets missing at the receiver. States $M_1, \dots, M_K$ are transient states, where $M_i$ stands for $i$ innovative packets missing, whereas $\mathcal{A}$ is an absorbing state that indicates that enough innovative packets have been received and thus transmission is finished.

For some state $M_i$, assume that each channel $l$ delivers $S^{(l)}$ packets according to some probability distribution $P_{S^{(l)}}$. Let us then define $S = \sum_l S^{(l)}$ as the total number of packets delivered. If the random variables $S^{(l)}$ are independent then $S$ is the sum of independent random variables and its probability distribution $P_S$ is simply the convolution of the individual distributions [39], as given by

$$P_S = P_{S^{(1)}} * P_{S^{(2)}} * \cdots * P_{S^{(L-1)}} * P_{S^{(L)}}. \tag{5.2}$$

Assuming that all packets delivered are innovative, the transition probabilities from some state $M_i$ are given by

$$q_{ij} = P_{M_i \to M_j} = P(S = i - j) \qquad , \forall i, j \in \{1, \dots, K\} \tag{5.3}$$

$$r_i = P_{M_i \to \mathcal{A}} = P(S \geq i) \qquad , \forall i \in \{1, \dots, K\}. \tag{5.4}$$

The transition probabilities are usually represented in the form of a transition probability matrix denoted here by $\mathbf{P}$. For the specified absorbing Markov chain with $K$ transient states and a single absorbing state, the transition probability matrix $\mathbf{P}$ can be written in the canonical form as

$$\mathbf{P} = \begin{bmatrix} \mathbf{Q} & \mathbf{R} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{5.5}$$

where $\mathbf{Q}$ is a $K$-by-$K$ matrix containing the transition probabilities $q_{ij}$ between transient states, $\mathbf{R}$ is a $K$-by-1 matrix containing the transition probabilities $r_i$ from the transient states to the absorbing state and $\mathbf{0}$ is a 1-by-$K$ zero matrix.

Besides the transition probabilities, the Markov chain is also characterized by an initial probability distribution. The initial probability distribution over transient states is represented by $\pi = [\pi_1 \cdots \pi_K]$, where $\pi_i$ denotes the probability of starting at state $M_i$. For all our purposes, we consider $\pi = [0\ 0 \cdots 0\ 1]$ to represent the fact that the receiver misses $K$ packets at the beginning of transmission.

Markov chain models allow the calculation of fundamental properties of its underlying process [39]. One that is of particular interest in this chapter is the expected number of steps until the absorbing state $\mathcal{A}$ is reached. If the steps are appropriately weighted by the corresponding number of slots one obtains the expected completion time of the transmission. Let $\mu_i$ be the mean number of slots until the absorbing state is reached when starting from transient state $M_i$. It is given by

$$\mu_i = c_i + \sum_{j=1}^{K} q_{ij} \mu_j, \tag{5.6}$$

where $c_i$ is the cost in slots associated with the transition from state $M_i$. Defining $\mu = \begin{bmatrix} \mu_1 & \cdots & \mu_K \end{bmatrix}^\top$ and $\mathbf{c} = \begin{bmatrix} c_1 & \cdots & c_K \end{bmatrix}^\top$, we can write Equation 5.6 in matrix form as

$$\mu = (\mathbf{I} - \mathbf{Q})^{-1}\mathbf{c}. \tag{5.7}$$

It is worth noting that $\mathbf{F} = \sum_{t=0}^{\infty} \mathbf{Q}^t = (\mathbf{I} - \mathbf{Q})^{-1}$ is the fundamental matrix associated with this Markov chain. Hence, $\mu = \mathbf{F}\mathbf{c}$. Finally, the expected time until absorption is simply

$$\Omega = \pi\mu. \tag{5.8}$$

From these expressions, it becomes clear that computing the transition probability matrix $\mathbf{Q}$ and the transition cost vector $\mathbf{c}$ is enough to characterize the expected completion time, since the initial state $\pi$ is already defined. The actual values for the transition probabilities and costs depend on the particular scheme being considered and thus shall be presented together with the respective scheme. The framework proposed here forms a base model that can be tailored to different schemes.

## 5.2 Fundamental Limits

The channel coding theorem [40] establishes that channel capacity is asymptotically achievable under the assumption of sufficiently large block length. For the transmission of limited blocks of information however, such result provides no guarantees on reliability of data transmission. Motivated by the problem of transmitting a finite block of $K$ packets, we derive a lower bound on the number of packets delivered with a given reliability for finite transmissions across multiple erasure channels. In the absence of feedback, such result provides valuable information concerning the state of the receiver and thus shall have an important role in our transmission schemes. Driven by this practical purpose, our approach relies on concentration inequalities to provide a simple bound that is reasonably tight and easy to compute.

Let $N^{(l)}$ represent the number of packets sent through channel $l$. The result of the transmission of the $j$-th packet sent through channel $l$ is given by the random variable $S_j^{(l)}$, whose value corresponds to the result of a Bernoulli experience with success probability $p^{(l)} = 1 - P_e^{(l)}$ and failure probability $q^{(l)} = P_e^{(l)}$. The total number of packets $N$ sent from the source is $N = \sum_{l=1}^{L} N^{(l)}$. The number of packets $S$ successfully received at the destination is also a random variable given by

$$S = \sum_{l=1}^{L} \sum_{j=1}^{N^{(l)}} S_j^{(l)}, \tag{5.9}$$

with mean and variance given by

$$\mu_S = \sum_{l=1}^{L} N^{(l)} p^{(l)}, \quad \sigma_S^2 = \sum_{l=1}^{L} N^{(l)} p^{(l)} q^{(l)}. \tag{5.10}$$

**Definition 4** (Reliability)**.** *We define the reliability objective $\varepsilon \in [0,1)$ as the minimum acceptable probability of delivering a given number of packets. To deliver at least $U$ packets while meeting the reliability objective it is sufficient to satisfy*

$$P(S \geq U) > \varepsilon. \tag{5.11}$$

Our work in [41] compares different probability inequalities for sums of independent bounded random variables [42] and concluded that Bernstein's inequality provides a good compromise between tightness and simplicity.

**Lemma 1** (Bernstein's Inequality)**.** *Let $Z_1, \ldots, Z_n$ be independent random variables with finite first and second moments, satisfying $Z_i \geq \mathbf{E}[Z_i] - b$. Let $Z = \sum_{i=1}^{n} Z_i$ with mean $\mu_Z = \mathbf{E}[Z]$ and variance $\sigma_Z^2 = Var(Z)$. For any $\delta \geq 0$,*

$$P(Z - \mu_Z \leq -\delta) \leq \exp\left(-\frac{\delta^2}{2\left(\sigma_Z^2 + \frac{1}{3}b\delta\right)}\right). \tag{5.12}$$

Leveraging the result presented in Lemma 1, we now present a key result of our work that provides a lower bound on the capacity of the network with reliability $\varepsilon$ when there is a limited number of transmissions in each route (Theorem 1).

**Theorem 1.** *Assume that the maximum number of transmissions $N^{(l)}$ per path $l$ and the reliability objective $\varepsilon$ are fixed. The number of packets that can be delivered with probability $\varepsilon$ is lower bounded by*

$$U = \mu_S - \frac{1}{3}\left(\beta b + \sqrt{(\beta b)^2 + 18\beta\sigma_S^2}\right), \tag{5.13}$$

*where $\beta = \ln(1/(1-\varepsilon))$ and $b = \max_l \left(p^{(l)}\right)$.*

*Proof.* The random variables $S_j^{(l)}$ correspond to the results of independent Bernoulli experiences. It follows that choosing $b = \max_l(p^{(l)})$ satisfies $S_j^{(l)} \geq 0 \geq p^{(l)} - b$ for all $l$ and $j$. It is also true that the first and second moments of a Bernoulli random variable are finite. Thus, the conditions of Lemma 1 are met. Applying Bernstein's inequality to $S$ we obtain

$$P(S - \mu_S < -\delta) \leq \exp\left(-\frac{\delta^2}{2\left(\sigma_S^2 + \frac{1}{3}b\delta\right)}\right). \tag{5.14}$$

Now let us simply rewrite Equation 5.11 as

$$P(S-\mu_S < U-\mu_S) \leq 1-\varepsilon. \tag{5.15}$$

By making $\delta = \mu_S - U$ the probabilities on the left-hand sides of Equation 5.14 and Equation 5.15 become the same. Equating the right-hand sides we obtain

$$\frac{\delta^2}{2\left(\sigma_S^2 + \frac{1}{3}b\delta\right)} = \ln\left(\frac{1}{1-\varepsilon}\right). \tag{5.16}$$

Solving the above equation for $U$ yields two solutions. As stated in Lemma 1, the inequality is only valid for $\delta \geq 0$. Since our formulation considers $\delta = \mu_S - U$ such condition only holds for $U \leq \mu_S$. By observing that $\beta b \leq \sqrt{(\beta b)^2 + 18\beta\sigma_S^2}$ one can conclude that $U \leq \mu_S$ only if the negative square root solution is considered, yielding the closed form solution inEquation 5.13. The result presented in Equation 5.13 also makes sense from a physical perspective since for $\varepsilon$ sufficiently close to one, the bound yields $\mu_S = U + \frac{2}{3}\beta b$. This means that the average number of packets delivered should be much higher than $U$ in order to deliver $U$ packets with a probability close to one, as expected. □

**Remark 1.** *Letting $N^{(l)} = T$ and $T \to \infty$ and assuming simultaneous transmissions, the result in Theorem 1 converges to $U/T = \sum_{l=1}^{L} p^{(l)}$, which constitutes the capacity (packets per slot) of a system with L communication channels with the same data rate. The value of $\varepsilon$ can be made arbitrarily close to one as $T \to \infty$. Thus, our bound is asymptotically tight.*

The evolution of $U/T$ is exemplified in Figure 5.3 for an increasing number of transmissions over two channels with asymmetric packet error probabilities.

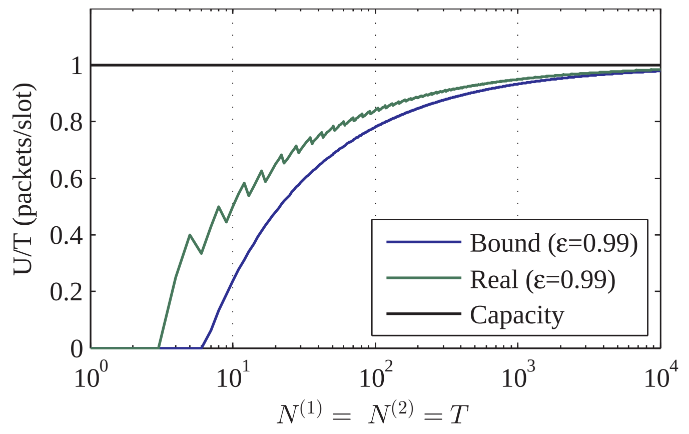


Figure 5.3: Comparison between our bound $U$ and the real $\varepsilon$-short-term capacity (calculated numerically) for an increasing number of time slots, when each channel transmits one packet per slot. The error probabilities in this case were set $P_e^{(1)} = 40\%$ and $P_e^{(2)} = 60\%$.

**Lemma 2** (Random Linear Network Coding)**.** *Assuming that $K$ packets are coded using random linear network coding with a large field size, i.e., they are linearly independent, it is sufficient to recover at least $K$ coded packets to decode the whole block. Under this assumption, $P(S \geq K) > \varepsilon$ implies decoding of the original block of $K$ packets with probability $\varepsilon$.*

In essence, the main result presented in Theorem 1 establishes a lower bound on the achievable rate of a network with limited packet transmissions per route. Under the current channel assumptions, random linear network coding achieves channel capacity asymptotically with the field size as it would deliver innovative packets in every transmission.

## 5.3 Time-Division Duplex Schemes

The scheduling of packets poses interesting challenges in the presence of delayed feedback. On the one hand, sending too few packets induces a cost since the time during which the source listens for feedback could have been spent sending useful coded data packets. On the other hand, if too many packets are sent and the channel is kept busy for too long the receiver may have received enough information but will be unable to send feedback therefore delaying the end of the transmission. This suggests that there is an optimal number of packets to send before stopping to listen for feedback in terms of the mean completion time. This problem was addressed in [43] for the case of a single channel.

The case of multiple channels poses extra challenges, as the above dilemma now coexists with the problem of allocating traffic to channels with asymmetric characteristics. Performing an exhaustive and unguided search for the optimal schedule composes an unpractical solution, as the solution space increases exponentially with the number of channels. A more viable solution could be to allocate data fractions to each channel, based either on their quality and/or timing characteristics, and then optimize the number of transmissions for each channel. Such a two-step optimization however, is unlikely to generate optimal results. Even if the initial allocation attempts to compensate for the different channel asymmetries, the same asymmetries will still induce different optimal numbers of transmissions for each channel. This translates into asymmetric block transmission times and poor delay performance. Therefore, joint optimization of redundancy level and packet scheduling across channels is critical for achieving good delay performance in the presence of asymmetric latencies and packet losses. Network coding facilitates this problem by simplifying the delivery of redundancy as well as the routing of packets.

Based on this, we propose two coding based schemes that use different strategies to complete the transmission. For each scheme, we provide a way to calculate the expected completion time based on the model in Section 5.1.3, provide an efficient algorithm for finding the optimal policy, and propose easy to compute heuristics that combine the bound obtained in Section 5.2 with time considerations to achieve near-optimal results.

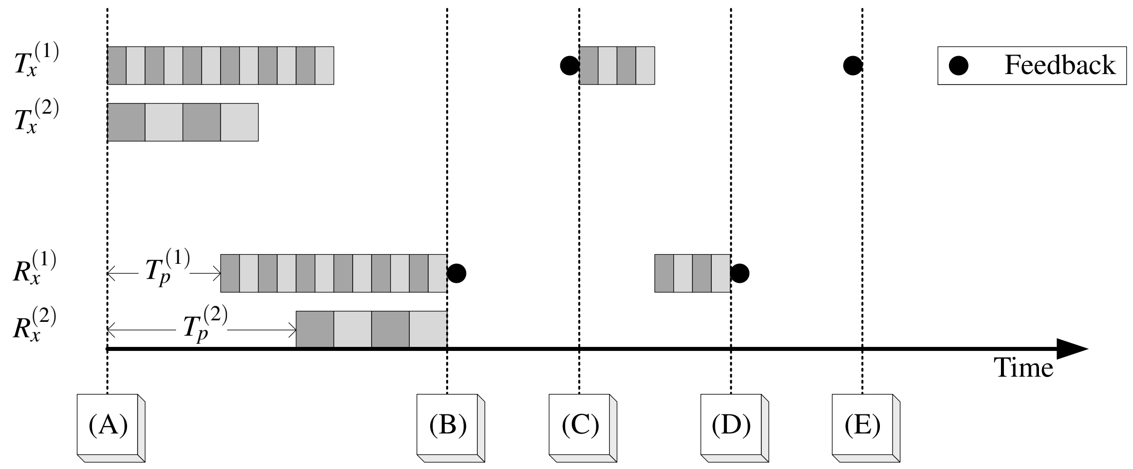


Figure 5.4: Schematic of the Coded Blocks scheme. (**A**) Due to its lowest propagation time, Channel 1 is chosen as the feedback channel, i.e., $f = 1$. Transmission then starts with $N_{CB}^{(1,K)}$ and $N_{CB}^{(2,K)}$ packets being transmitted over Channel 1 and Channel 2, respectively. (**B**) Data reception from all channels ends, at which point feedback is immediately sent back through Channel 1, acknowledging that $i$ packets are still missing. (**C**) Feedback reaches the sender, which schedules a new round of $N_{CB}^{(1,i)}$ and $N_{CB}^{(2,i)}$ packet transmissions for delivering the $i$ packets missing. In this case, because of the different propagation times and the number of packets that need to be transmitted, it does not compensate to send packets over Channel 2. (**D**) All packets in the second round have been received, and enough packets have been delivered. Therefore, feedback is sent back to acknowledge that transmission is complete. (**E**) Feedback is received by the sender at which point transmission is considered to be over.

### 5.3.1 Coded Blocks (CB)

In this scheme the sender transmits a scheduled number of packets $N_{CB}^{(l,i)}$ over each channel $l$ depending on the number of packets $i$ missing at the receiver at each transmission round. After sending the scheduled number of packets, the sender stops transmitting in all channels and waits for feedback. Feedback reports the number of packets still needed to complete the transmission and is sent back through the channel with lowest propagation time, i.e., $f = \arg\min_l(T_p^{(l)})$. Upon reception of feedback, and if there are packets missing, a new transmission round begins for the number of missing packets $i$ reported. Transmission finishes when the receiver announces that it has received enough innovative packets. The schedule of packets $\mathcal{N}_{CB} = \{N_{CB}^{(l,i)}\}$ for all channels $l$ and possible values of missing packets $i$ forms the transmission policy for this scheme. For a better understanding, refer to the example provided in Figure 5.4.

**Completion Time** Consider the Markov chain described in Section 5.1.3. Let the state represent the sender's knowledge regarding the number of missing packets at the receiver. Under this assumption, state transitions occur when feedback is received, corresponding to the time when the sender's knowledge is updated. The transition probabilities from a given state $M_i$ depend on the number of packets $N_{CB}^{(l,i)}$ sent in that state. Particularly, we have independent binomial distributions $P_{S^{(l)}} = Bi(N_{CB}^{(l,i)}, 1 - P_e^{(l)})$, based on which the joint distribution of packets delivered $P_S$ can be calculated according to Equation 5.2. Assuming that all packets are innovative, the transition probabilities from state $M_i$ can be calculated as in Equation 5.3. Calculating the transition probabilities between all transient states yields the transition probability matrix $\mathbf{Q}$. The transition cost $c_i$ associated with a transition from state $M_i$ corresponds to the number of slots needed to deliver all packets scheduled in that state and receive feedback. This value is given by $c_i = \max_l (T_{N^+}^{(l)}(N_{CB}^{(l,i)})) + T_p^{(f)}$, where $T_{N^+}^{(l)}(n) = nT_t^{(l)} + T_p^{(l)}$ if $n > 0$ and zero otherwise. With $\mathbf{Q}$ and $\mathbf{c}$, the mean completion time can be computed using Equation 5.8.

**Optimal Policy** The optimal policy $\mathcal{N}_{CBO} = \{N_{CBO}^{(l,i)}\}$ is the transmission policy that minimizes the expected completion time. The brute force approach for finding the optimal solution is to perform a grid search on the solution space $\mathbb{N}_0^{L \times K}$. However, that approach does not scale with the number of channels and block size. Fortunately, this task can be greatly simplified by exploiting the structure of the problem.

Since the number of packets missing can only decrease, we have transition probabilities $q_{ij} = 0$ for all $j > i$. This implies that the calculation of each $\mu_i$ does not depend on those $\mu_j$ for which $j > i$. This allows us to optimize each state independently, starting from $M_1$ until $M_K$.

The optimization within each state can also be simplified. Instead of performing an exhaustive search over all possible combinations of packets, we propose a mechanism to guide the search, as detailed in Algorithm 2. At the core of the algorithm, lies the observation that for minimizing the transmission time, the best solution is to send as many packets as possible over each channel within a given time window. Following that logic, the number of transmission slots is iteratively increased and the candidate solution is updated accordingly, as well as the best solution. If the candidate solution is worse than in the previous iteration then the channels that contributed to the new solution are removed from the set of channels $\mathcal{L}$ that improve the solution. The search stops when the set of channels that improve the solution becomes empty. This stopping condition is a heuristic that assumes that once adding packets in each channel degrades the performance, a further increase in transmitted packets will continue to do so. More conservative approaches can be implemented, e.g., stopping when the solution does not improve for multiple iterations.

**Heuristic Policy** The heuristic policy $\mathcal{N}_{CBH} = \{N_{CBH}^{(l,i)}\}$ is based on two fundamental observations. First, in order to maximize channel usage, channels should not wait for each other to finish, i.e., the respective transmissions as seen from the receiver should end at the same time. The logic is that, since feedback is considered only when all packets from all channels arrive to the receiver, having packets arriving from some channel but not from others it not optimal with respect to delay. If that is the case, more packets could have been sent in the other channels, increasing the probability of finishing the

**Algorithm 2** Optimal Policy

```
for all $i \in \{1,\dots,K\}$ do
    $slots \leftarrow 0$
    $N^{(1:L,i)} \leftarrow 0$
    $\mathcal{L} \leftarrow \{1,\dots,L\}$
    repeat
        $slots \leftarrow slots + 1$
        $\mathcal{R} \leftarrow \{l : \text{packet arrives from } l \text{ at } slots\}$
        $N^{(\mathcal{R},i)} \leftarrow N^{(\mathcal{R},i)} + 1$
        $candidateTime \leftarrow \text{ComputeTime}(N^{(1:L,1:i)})$
        if $candidateTime \le lastTime$ then $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{R}$
        else $\mathcal{L} \leftarrow \mathcal{L} \setminus \mathcal{R}$
        end if
        $lastTime \leftarrow candidateTime$
        if $candidateTime < bestTime$ then
            $bestTime \leftarrow candidateTime$
            $N_{CBO}^{(1:L,i)} \leftarrow N^{(1:L,i)}$
        end if
    until $(\mathcal{L} = \emptyset)$
    $N^{(1:L,i)} \leftarrow N_{CBO}^{(1:L,i)}$
end for
return $N_{CBO}^{(1:L,1:K)}$
```

transmission in the same time. Second, transmissions should be finished in a single round in order to bring feedback costs to a minimum and for that reason it may be advantageous to schedule enough redundant coded packets to guarantee decoding with high probability. Based on these insights, we propose an heuristic for calculating the number of packets $N_{CBH}^{(l,i)}$ to send over channel $l$ when $i$ packets are missing. The heart of the heuristic is the linear-quadratic system of equations in Equation 5.17, which brings the redundancy and timing considerations together.

$$\begin{cases} i = \mu_S - \frac{1}{3}\left(\beta b + \sqrt{(\beta b)^2 + 18\beta\sigma_S^2}\right), & \text{(Theorem 1)} \\ T_N^{(l_1)}(N_{CBH}^{(l_1,i)}) = \cdots = T_N^{(l_{|\mathcal{L}|})}(N_{CBH}^{(l_{|\mathcal{L}|},i)}), & l_j \in \mathcal{L} \end{cases} \tag{5.17}$$

The first equation is taken from Theorem 1 and draws the solution towards a target reliability, ensuring that $i$ packets are delivered with a minimum probability $\varepsilon$. Thus, the amount of redundancy can be tuned by changing the value of $\varepsilon$. The remaining $|\mathcal{L}|-1$ equations relate the transmission times $T_N^{(l)}(n) = nT_t^{(l)} + T_p^{(l)}$ of the channels in set $\mathcal{L}$. To keep the equations linear, $T_N^{(l)}(n)$ is assumed to hold for all $n$. However, its output for $n \le 0$ has no physical interpretation. For that reason, the transmission time for a given channel should only come in the equations if there are packets to be sent over that channel. Since the solution is not known beforehand and neither are the subset of channels that should

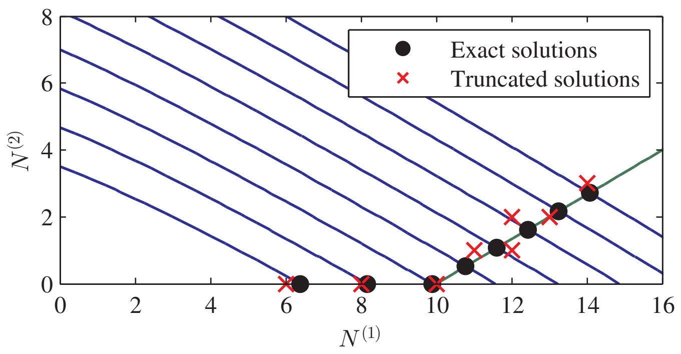


Figure 5.5: Graphical solution of the heuristic for the coded blocks scheme with $L = 2$ channels, block size $K = \{1,...,8\}$ and target reliability $\varepsilon = 95\%$. The channels parameters are $P_e^{(1)} = 25\%$, $P_e^{(2)} = 5\%$, $T_t^{(1)} = 2$, $T_t^{(2)} = 3$, $T_p^{(1)} = 10$, $T_p^{(2)} = 30$.

**Algorithm 3** Heuristic Policy

```
for all $i \in \{1,...,K\}$ do
    $\mathcal{L} = \{1,...,L\}$
    repeat
        validSolution $\leftarrow$ $true$
        $N_{CBH}^{(1:L,i)}$ = SolveSystem($\mathcal{L}$)
        for all $l \in \mathcal{L}$ do
            if $N_{CBH}^{(l,i)} \leq 0$ then
                $N_{CBH}^{(l,i)} \leftarrow 0$
                $\mathcal{L} \leftarrow \mathcal{L} \setminus l$
                validSolution $\leftarrow$ $false$
            end if
        end for
    until validSolution
end for
return $\lceil (N_{CBH}^{(1:L,1:K)} \rceil$
```

participate, our solution iteratively solves and modifies the system of equations until a valid solution is found, as described in Algorithm 3. At the start, $\mathcal{L}$ is initialized as the set of all channels. At each step, the algorithm generates and solves the system. Afterwards, it checks for all $l$ in $\mathcal{L}$ if $N_{CBH}^{(l,i)}$ is non-positive, in which case $N_{CBH}^{(l,i)}$ is set to zero and $l$ is removed from $\mathcal{L}$. The process is then repeated until all channels in $\mathcal{L}$ have a positive solution. Naturally, the solution is truncated to yield integer numbers of packets. The graphical solution for a specific set of parameters is shown in Figure 5.5.

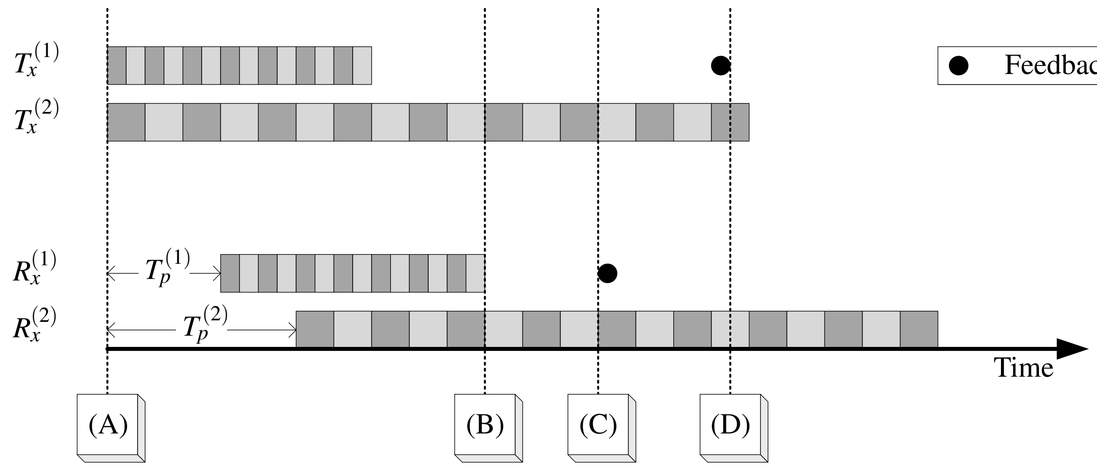


Figure 5.6: Schematic of the Coded Streams scheme. (**A**) Due to its lowest propagation time, Channel 1 is chosen as the feedback channel, i.e., $f = 1$. Transmission then starts with a given number of packets $N_{CS}^{(1)}$ being scheduled and transmitted over Channel 1. In contrast, a transmission of stream of packets is initiated over Channel 2. (**B**) All packets from Channel 1 have reached the receiver, point after which feedback can be sent, but only when enough packets have been received. Since the receiver still misses some packets, which are supplied by the stream of packets arriving from Channel 2. (**C**) At this point, the receiver has obtained enough packets, hence feedback is sent back through Channel 1 to acknowledge that transmission is complete. Unaware of this, the sender keeps transmitting packets over Channel 2. (**D**) Feedback is received by the sender and transmission is considered over at this point, albeit the receiver will still be receiving the packets in transit over Channel 2.

### 5.3.2 Coded Streams (CS)

In this scheme the sender transmits a scheduled number of packets $N_{CS}^{(f)}$ over the lowest latency channel, i.e., $f = \arg\min_l(T_p^{(l)})$. After sending the scheduled number of packets the sender stops transmitting on channel $f$ and uses it to listen for feedback. In the remaining channels, coded packets are continuously transmitted until the receiver has obtained enough packets. Once that happens and transmissions on channel $f$ are finished, a single feedback packet announcing the end of the transmission is sent back through channel $f$. The transmission ends when feedback is received at the sender. The schedule of packets $\mathcal{N}_{CS} = N_{CS}^{(f)}$ of the number of packets to send over channel $f$ forms the transmission policy for this scheme. For a better understanding, refer to the example provided in Figure 5.6.

**Completion Time** The calculation of the completion time for this scheme requires more elaborate techniques since the Markov chain defined in Section 5.1.3 cannot be used to model it as CB. That would consist of modeling the sender's knowledge of the receiver's state. Unfortunately, since CS only sends feedback at the end of transmission, there would be only one transition from state $M_K$ to state $\mathcal{A}$ with probability one but the cost of that transition would be precisely the expected completion time that needs to be calculated. The alternative is to use the Markov chain to model the receiver's state

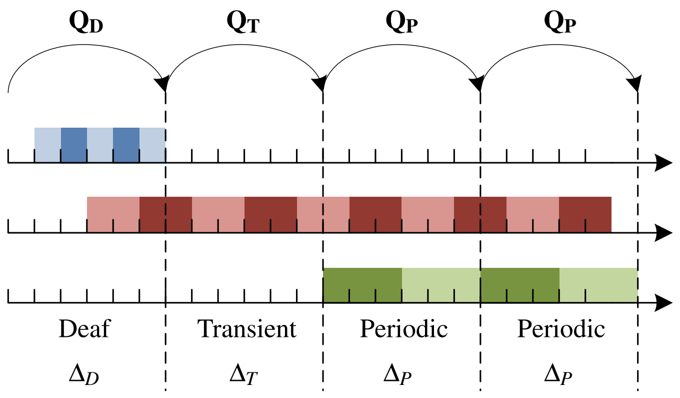


Figure 5.7: Packet transmissions using coded streams scheme as seen by the receiver. There are 3 channels with transmission times $T_t^{(1)} = 1$, $T_t^{(2)} = 2$, $T_t^{(3)} = 3$, and propagation times $T_p^{(1)} = 1$, $T_p^{(2)} = 3$, $T_p^{(3)} = 12$. The feedback channel $f = 1$ sends a scheduled number of packets before stopping.

Table 5.1: Coded stream scheme interval characterization

| | Number of Slots | Number of Packets |
|---|---|---|
| Deaf | $\Delta_D = N_{CS}^{(f)} T_t^{(f)} + T_p^{(f)}$ | $N_D^{(l)} = \max(0, \lfloor(\Delta_D - T_p^{(l)})/T_t^{(l)}\rfloor)$ |
| Transient | $\Delta_T = \max(0, \max_l(T_p^{(l)} - \Delta_D))$ | $N_T^{(l)} = \lfloor(\Delta_D + \Delta_T - T_p^{(l)})/T_t^{(l)}\rfloor - N_D^{(l)}$ |
| Periodic | $\Delta_P = lcm(T_t^{(l)} : l \neq f)$ | $N_P^{(l)} = \Delta_P / T_t^{(l)} \quad \forall l \neq f$ |

directly, as seen from the receiver. The immediate approach would be to consider state transitions at every time slot. However, that would break the time-homogeneous Markov model since the transition probabilities vary from slot to slot. To work around that, one could build a periodic Markov chain following the ideas in [43]. For multiple asymmetric channels though, such a model would not scale well. Therefore, we propose a new approach to handle this type of transmissions that takes advantage of periodic patterns in the transmission.

To better understand our approach, consider the illustrative example of Figure 5.7. Although analytically infinite, the transmission can be segmented into multiple finite intervals, which we denote as *deaf*, *transient*, or *periodic*. The transmission starts with a deaf interval during which the receiver receives packets from channel $f$, only, and cannot send feedback. The transmission can then go through a transient phase until the receiver starts receiving through all channels (except $f$), where transmitting feedback is possible. Afterwards, the transmission enters a periodic phase, in which the receiver receives a periodic pattern of packets through all links except $f$. Each of these periods is well characterized in terms of their duration and number of packets delivered, as summarized in Table 5.1.

Since the number of packets arriving within all intervals is well defined, the transition probability matrices $\mathbf{Q_D}$, $\mathbf{Q_T}$ and $\mathbf{Q_P}$ can be calculated for state transitions across the respective intervals. However, these matrices provide no information about what happens in each slot between interval boundaries. In fact, a transmission can end in any time slot after $\Delta_D$, which in general does not coincide with a state transition point. As a consequence, if we were to consider the $\Delta$s as the transition costs, the results from Section 5.1.3 for calculating the expected absorption time would not accurately translate the expected completion time. To get an accurate calculation we resort to the definition of expected value. Let us denote by $X_t$ the state of the Markov chain at some time $t$. Consider a time interval $\tau = \{1, \ldots, \Delta\}$ and the random variable $\Gamma(\Delta)$ defined in Equation 5.18.

$$\Gamma(\Delta) \triangleq \min(\Delta, t : X_t = \mathcal{A} \wedge X_{t-1} \neq \mathcal{A}). \tag{5.18}$$

This variable can be interpreted as the time until absorption limited to the interval duration $\Delta$. Its expected value conditioned on initial state $M_i$ is denoted here by $c_i(\Delta) \triangleq \mathbf{E}[\Gamma(\Delta)|X_0 = M_i]$ and calculated as in Equation 5.19.

$$\begin{aligned} c_i(\Delta) &= \sum_{t=1}^{\infty} \min(\Delta, t) \cdot P_{\mathcal{A}|i}(t) && \text{(5.19a)} \\ &= \sum_{t=1}^{\Delta-1} t \cdot P_{\mathcal{A}|i}(t) + \Delta \cdot \sum_{t=\Delta}^{\infty} P_{\mathcal{A}|i}(t) && \text{(5.19b)} \\ &= \sum_{t=1}^{\Delta-1} t \cdot P_{\mathcal{A}|i}(t) + \Delta \cdot \left(1 - \sum_{t=1}^{\Delta-1} P_{\mathcal{A}|i}(t)\right) && \text{(5.19c)} \\ &= \sum_{t=1}^{\Delta-1} (t-\Delta) \cdot P_{\mathcal{A}|i}(t) + \Delta, && \text{(5.19d)} \end{aligned}$$

For compactness we defined $P_{\mathcal{A}|i}(t)$ as in Equation 5.20, which can be interpreted as as the probability of being absorbed exactly at time $t$ given that the initial state was $M_i$.

$$\begin{aligned} P_{\mathcal{A}|i}(t) &\triangleq P(X_t = \mathcal{A}, X_{t-1} \neq \mathcal{A} | X_0 = M_i) \\ &= P(X_t = \mathcal{A}|X_0 = M_i) - P(X_{t-1} = \mathcal{A}|X_0 = M_i). \end{aligned} \tag{5.20}$$

Note that the sum in Equation 5.19d does not need to be evaluated for every slot $t$ until $\Delta - 1$ because $P_{\mathcal{A}|i}(t)$ is zero if no packets arrive at time $t$. Applying these expressions to the transient and periodic phases, we obtain the vectors $\mathbf{c_T} = \left[c_1(\Delta_T) \; \ldots \; c_K(\Delta_T)\right]^\top$ and $\mathbf{c_P} = \left[c_1(\Delta_P) \; \ldots \; c_K(\Delta_P)\right]^\top$. Note that for the deaf interval the duration is deterministic because feedback cannot be transmitted. Therefore, the expected completion time is lower bounded by $\Delta_D$. It is possible to show that the mean times until absorption $\mu$ can be calculated as in Equation 5.21.

$$\begin{aligned} \mu &= \Delta_D + \mathbf{Q_D c_T} + \sum_{j=0}^{\infty} \mathbf{Q_D Q_T Q_P}^j \mathbf{c_P} && \text{(5.21a)} \\ &= \Delta_D + \mathbf{Q_D c_T} + \mathbf{Q_D Q_T} \left(\sum_{j=0}^{\infty} \mathbf{Q_P}^j\right) \mathbf{c_P}. && \text{(5.21b)} \end{aligned}$$

Recognizing the infinite sum as the fundamental matrix $\mathbf{F_P}$ associated with $\mathbf{Q_P}$, we can rewrite this last expression as

$$\mu = \Delta_D + \mathbf{Q_D c_T} + \mathbf{Q_D Q_T F_P c_P}. \tag{5.22}$$

Each summand in Equation 5.22 corresponds to the contribution of each type of interval to the total time. Furthermore, a careful look at the summands corresponding to the transient and periodic intervals reveals similarities with Equation 5.7. In fact, the expectations calculated in Equation 5.19 work as transition costs, hence the used notation. Finally, the expected completion time $\Omega$ is given by

$$\Omega = \pi\mu + T_p^{(f)}, \tag{5.23}$$

where $T_p^{(f)}$ is added because $\pi\mu$ constitutes the time to complete at the receiver and, therefore, the time for the feedback to reach the sender must be added.

**Optimal Policy (CSO)** Finding the optimal policy $\mathcal{N}_{CSO} = N_{CSO}^{(f)}$ for this scheme is a subset of the same problem of CB. Now, the search only needs to be performed for a single channel and it does not require an optimization over multiple states. Hence, the search is straightforward.

**Heuristic Policy (CSB)** For obtaining the heuristic policy $\mathcal{N}_{CSH} = N_{CSH}^{(f)}$ for this scheme, we calculate the heuristic as if it was for the coded blocks scheme and consider only the result for $N_{CBH}^{(f,K)}$ corresponding to the schedule for channel $f$ when $K$ packets are missing. By the time the feedback channel is done transmitting its packets, the total number of packets sent by all channels is enough to complete transmission with a target reliability.

## 5.4 Numerical Results

### 5.4.1 Comparison Schemes

This section defines alternative transmission policies that are used for comparison purposes. In particular we define one policy that sets the best possible performance and two other that do not use coding and follow traditional approaches based on retransmissions.

**Coded Full-Duplex (CFD)** The sender transmits coded packets across all channels and stops only when a feedback packet signaling the end of the transmission is received. In order to minimize the completion time, feedback is sent through the link with the lowest propagation time. The channels are assumed to be full-duplex so the source does not need to stop the transmission in order to receive feedback. Although this is not a TDD scheme, it provides us with a gold standard (lower bound) in terms of time to complete the transmission and is used as a reference for the other schemes.

**Uncoded Duplicate (UD)** The sender transmits each original packet in every channel. After each transmission round, i.e., transmission of every missing packet, the receiver acknowledges the new set of missing packets through the channel with lowest propagation time and the process is repeated until the receiver announces the reception of all packets.

**Uncoded Split (US)** The sender allocates each original packet to one of the channels. Every packet sent in channel $l$ is retransmitted $N_{US}^{(l)}$ times (Equation 5.24) depending on the packet erasure probability $P_e^{(l)}$ of the channel. Specifically, we define the variants ceiling (USC) and floor (USF).

$$N_{USC}^{(l)} = \left\lceil \frac{1}{1 - P_e^{(l)}} \right\rceil \quad \text{or} \quad N_{USF}^{(l)} = \left\lfloor \frac{1}{1 - P_e^{(l)}} \right\rfloor, \tag{5.24}$$

The distribution of packets across channels attempts to level out the transmission times in a similar fashion to what is done in the coded blocks scheme. In this sense, this uncoded scheme is more clever than the previous one since it includes the time parameters of the channels in the decision. After each transmission round, the receiver acknowledges which packets are missing through the channel with lowest propagation time and the process is repeated until the receiver announces the reception of all packets.

### 5.4.2 Coded vs Uncoded

In this section, we compare coded and uncoded schemes over several configurations. We restrict coded schemes to the optimal ones not only to simplify the presentation, but also to focus the analysis on the achievable gains.

We start by analyzing the impact of the propagation time on the expected time to complete the transmission with the different schemes. We start by considering increasingly symmetric propagation times. The results are shown in Figure 5.8 for equal propagation times ranging from 0 to 200 slots, and packet error rate of 80% in each channel. For small delays, the UD scheme performs reasonably well, whereas US schemes suffer from sending too much redundancy. After a point, the extra redundancy sent with US schemes starts to pay-off and its performance surpasses that of the UD scheme. Even so, the performance of all uncoded schemes degrades severely as the transmission delay starts to increase, as does the their gap in performance. In contrast, coded schemes can potentially achieve a performance very close to the full-duplex case, and not as dependent on system conditions. For small

In Figure 5.9, we consider that the propagation time of channel 1 is kept fixed at $T_p^{(1)} = 50$ slots while the propagation time of channel 2 is varied. The packet error rate is 20% for each channel. A major difference in this setting is that the feedback cost is limited by $T_p^{(1)}$. In the interval until $T_p^{(2)} = 50$, feedback is sent over channel 2 and thus the cost associated with it increases with $T_p^{(2)}$. After that point, feedback is always sent through channel 1, which has fixed propagation delay. This has most impact on schemes that have to go back and forth several times, which is more common in uncoded schemes.

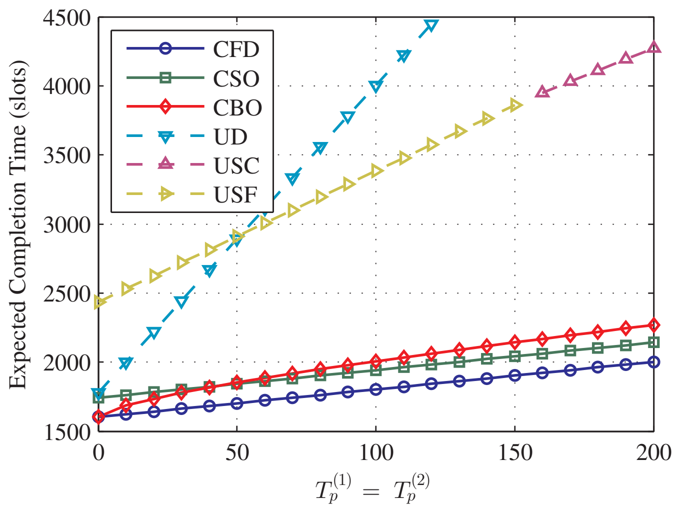


Figure 5.8: Comparison of coded and uncoded schemes over increasingly symmetric propagation delays. The system parameters are $K = 64$, $L = 2$, $P_e^{(1)} = 0.80$, $P_e^{(2)} = 0.80$, $T_t^{(1)} = 10$, $T_t^{(2)} = 10$.

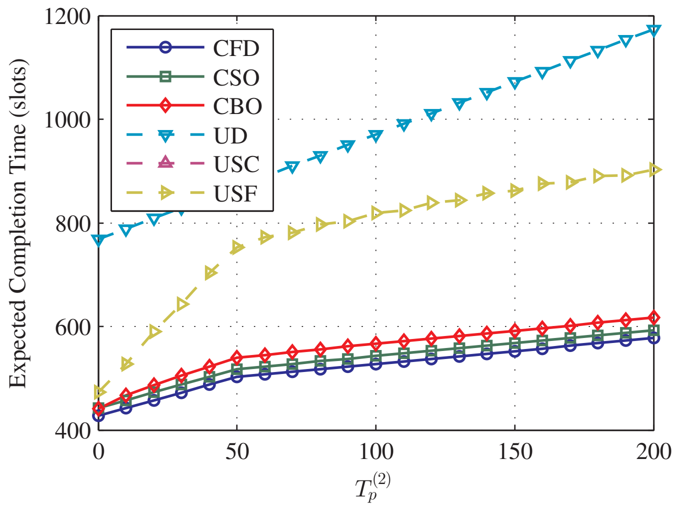


Figure 5.9: Performance comparison of coded and uncoded schemes over increasing asymmetry in propagation delays. The system parameters are $K = 64$, $L = 2$, $P_e^{(1)} = 0.20$, $P_e^{(2)} = 0.20$, $T_t^{(1)} = 10$, $T_t^{(2)} = 10$, $T_p^{(1)} = 50$.

For that reason, unlike the results in Figure 5.8, the performance of uncoded schemes stabilizes after $T_p^{(2)} = 50$. For coded schemes, the difference is not that noticeable as these are more capable to finish the transmission in a single round.

When feedback is not delayed, i.e., at least one channel has propagation time equal to zero, all schemes should perform optimally as perfect information is always available to the sender. Our results do not evidentiate that as their operation does not change for the special case of instantaneous feedback and do not make the best use of such information. The real challenges arise in the presence of both delays and errors. For those situations, these results show that coded transmissions are difficult to outperform as they offer near-optimal performance even with high latencies with simple logic, while each uncoded scheme provides (at best) a reasonable solution only in a very limited parameter region.

### 5.4.3 Coded Optimal vs Coded Heuristic

In this section, we analyze the performance of the heuristic policies. The purpose of this analysis is twofold. First, to evaluate the heuristics' performance compared to the respective optimal policies. Secondly, to study the impact of different values of the reliability parameter $\varepsilon$ in different settings. For each scheme, we test the heuristics with $\varepsilon = 0.00$, $\varepsilon = 0.50$ and $\varepsilon = 0.90$, which set lower bounds of 0%, 50%, and 90% on the probability of finishing the transmission, respectively.

We start the analysis with the CB scheme. This scheme is characterized by a silent time gap that corresponds to the time where the sender is waiting for feedback. During that time, all channels are idle. As a consequence, if the feedback delay is large and transmission is not finished in a single round, then the performance goes down. Depending on the parameters, it may compensate to send extra redundancy as suggested by the results in Figure 5.10. The results shown are for increasingly symmetric propagation times, which bring feedback delay up with them. When the delay is relatively small, sending just the mean number of packets ($\varepsilon = 0$) is the best choice. As the delay goes up, increasing the amount of redundancy yields better results, with $\varepsilon = 0.50$ performing better up to 125 slots, and $\varepsilon = 0.90$ from thereon.

We now analyze CS in a particular setting. This scheme is not as susceptible to silent gaps as CB and for that reason, this scheme works generally well even with heuristics in low reliability settings. There are nonetheless corner cases where the heuristic policies may benefit from extra redundancy. That happens when the propagation time of the feedback channel is much smaller compared to the other ones. In extreme cases, the feedback channel may finish transmitting its scheduled packets while packets from other channels are still on their way to the receiver, creating a silent gap. If transmission is not finished by the feedback channel, then transmission must wait for packets from other channels to arrive. To avoid that, it may be beneficial to send extra redundancy on the feedback channel. The results in Figure 5.11 support this reasoning. The results are for increasingly symmetric packet error rates and highly asymmetric propagation times. When the error rate is small, transmission through the feedback channel ends before packets from other channels reach the receiver. Hence, choosing to send a high amount of redundancy ($\varepsilon = 0.90$) yields the best results. As the error rates increase, so does the number of packets sent in the feedback channel which has the effect of narrowing the gap. Depending on the gap size, different settings of redundancy perform better than others as shown by the results.

We conclude this analysis noting that with any reasonable choice of the target reliability $\varepsilon$, the heuristics fall within 1 dB of the respective optimal policies.

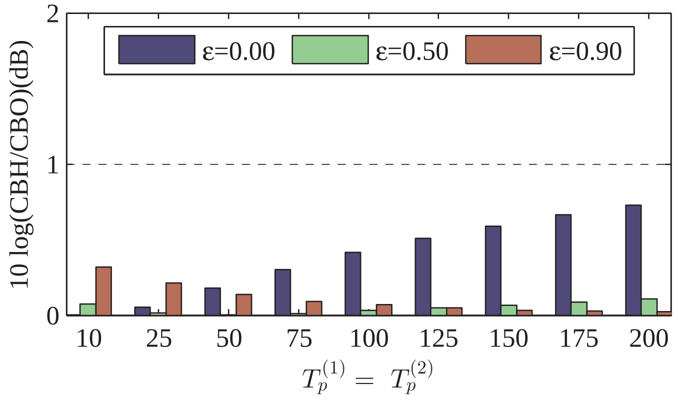


Figure 5.10: Comparison of the expected completion time between optimal and heuristic CB policies for different $\varepsilon$. The system parameters are $K = 64$, $L = 2$, $P_e^{(1)} = 0.20$, $P_e^{(2)} = 0.15$, $T_t^{(1)} = 10$, $T_t^{(2)} = 10$.

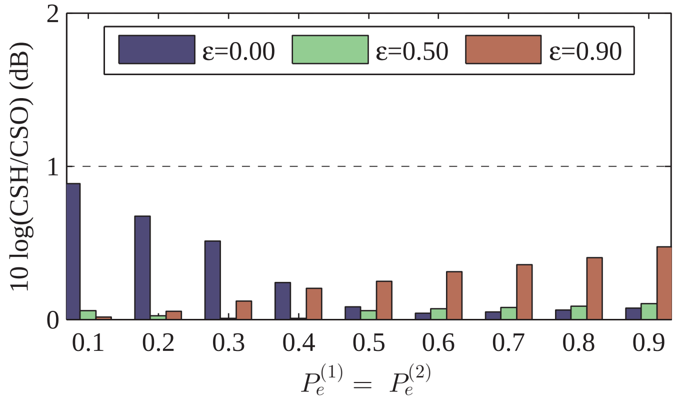


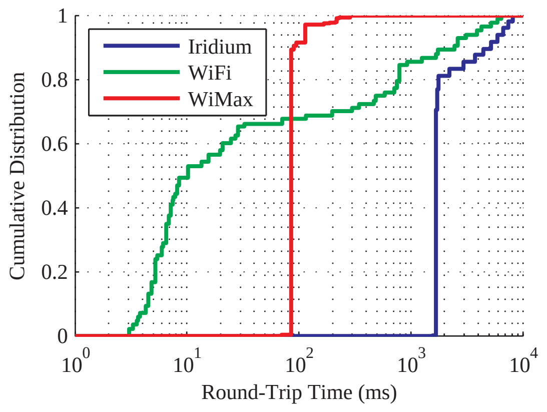
Figure 5.11: Comparison of the expected completion time between optimal and heuristic CS policies for different $\varepsilon$. The system parameters are $K = 64$, $L = 2$, $T_t^{(1)} = 4$, $T_t^{(2)} = 10$, $T_p^{(1)} = 50$, $T_p^{(2)} = 600$.

## 5.5 Practical Significance

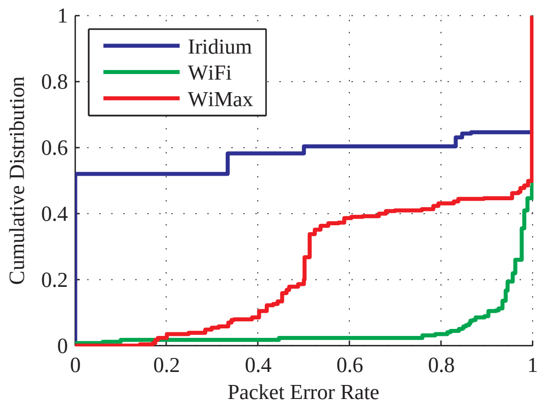
So far, our formulation remained generic, allowing it to be applicable to a wide number of situations where the considered challenges are present. Indeed, one can think of a number of different applications that would fit in the considered settings. Notable examples include communications in multi-beam satellite networks [44], underwater communication with multiple acoustic modems, and multihomed communication in heterogeneous networks.

In this section, we provide a realization of our framework in a mobile multihoming scenario. The settings we will use are based on experimental data obtained by Cloud *et al.* [31], whose results we partially reproduce in Figure 5.12 for convenience of the reader.

Inspired in the referred work, we consider the use of WiFi and WiMax interfaces, both running at 20 *Mbps*. Since the transmission rates are the same and the round-trip times are, in the considered



(a) Round-Trip Time Distribution



(b) Packet Error Rate Distribution

Figure 5.12: Measured Round-Trip Time (RTT) and Packet Error Rate (PER) for different wireless access technologies in a mobile scenario. The data was obtained by Cloud *et al.* [31].

example, larger than the transmission time of a packet, we can define $T_{slot}$ to be the packet transmission time, calculated in Equation 5.25 for a packet size $W = 1350\ bytes$.

$$T_{slot} = \frac{8 \times W}{r^{(l)}} = \frac{8 \times 1350}{20 \times 1024^2} \approx 0.515\ ms \tag{5.25}$$

The packet transmission time for both interfaces is then by definition equal to one slot.

$$T_t^{(WiFi)} = T_t^{(WiMax)} = 1\ slot \approx 0.515\ ms$$

The *round-trip time (RTT)*, which characterizes the delay in data transport, is cast as a propagation time in our model, where the propagation time of one link is assumed to be equal to half of its RTT. According to the data in Figure 5.12a, the RTT for WiMax is tightly concentrated around 80 $ms$, whereas the RTT for WiFi is much more spread, with 80% of the values lying in the range between 3 $ms$ and 900 $ms$.

$$T_p^{(\text{WiFi})} = \frac{RTT^{(\text{WiFi})}}{2T_{slot}} \in \frac{[3;900]\ ms}{2 \times 0.515\ ms} \approx \{3,\ldots,874\}\ slots$$
$$T_p^{(\text{WiMax})} = \frac{RTT^{(\text{WiMax})}}{2T_{slot}} = \frac{80\ ms}{2 \times 0.515\ ms} \approx 78\ slots$$

Regarding the packer error rate, consider the data in Figure 5.12b. The PER of the WiMax interface concentrates around 50% and 100%, whereas the PER of WiFi concentrates around 95% and 100%. Observations of 100% PER correspond to loss of connectivity, cases that we disregard as we are interested in the case where there is connectivity over both interfaces. This is a reasonable assumption since, as shown in [31], the moving node is often connected to multiple networks. Therefore, we consider the following values for the PER of the considered interfaces.

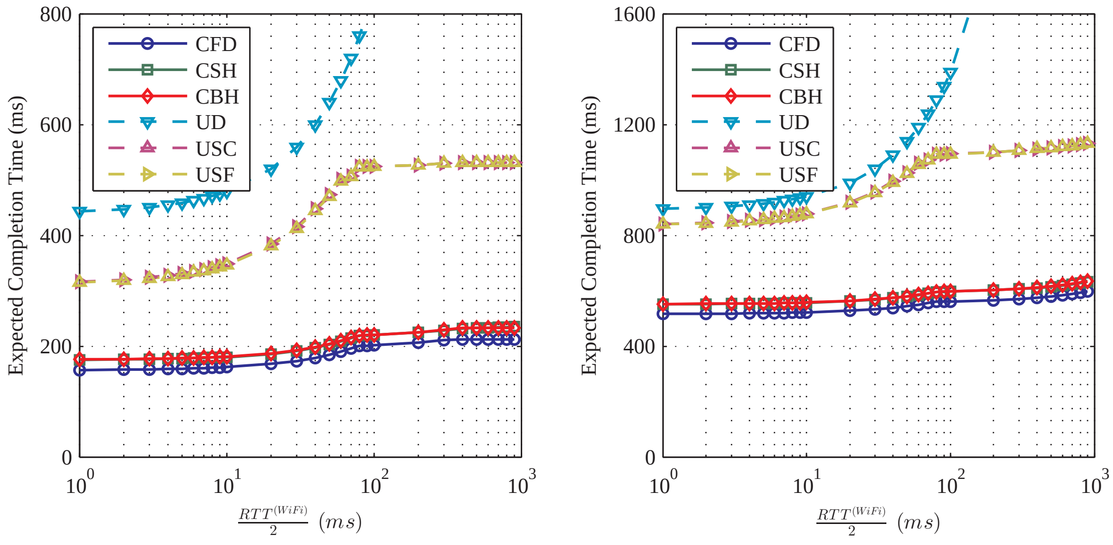
$$P_e^{(\text{WiFi})} = 95\% \quad \text{and} \quad P_e^{(\text{WiMax})} = 50\%$$

Based on this characterization, we can calculate the expected block transmission time using our framework and compare the performance of the different schemes. Given the higher dispersion of the WiFi RTT, we performed simulations over the observed range of variation, keeping the other parameters fixed as specified in Table 5.2. Moreover, being this example motivated by practice, we consider the heuristic versions of each coded schemes due to their simpler, and therefore more practical, construction. In particular, we consider a high redundancy setting obtained with $\varepsilon = 0.90$, due to the potential of high RTT values. The results are shown in Figure 5.13a for a block of $K = 128$ packets and in Figure 5.13b for a block of $K = 512$ packets.

As before, the performance of both coded schemes far exceeds that of uncoded ones, coming very close to the best achievable performance over the whole range of operation. Such closeness to the optimal CFD performance is particularly noteworthy given the use of heuristic schemes with a fixed redundancy settings, which says plenty about the efficiency and robustness of the schemes. In contrast,

Table 5.2: Channel model parameters for WiFi and WiMax based interfaces

| | WiFi | WiMax |
|---|---|---|
| $T_t$ | 1 *slot* | 1 *slot* |
| $T_p$ | $3,\ldots,874$ *slots* | 78 *slots* |
| $P_e$ | 95% | 50% |



(a) K=128 (b) K=512

Figure 5.13: Expected completion time for coded and uncoded schemes over varying WiFi RTT.

the performance of uncoded schemes is quite dependent on the operating conditions, and in particular on feedback associated costs. This is illustrated by the accentuated variation in performance until the $RTT^{\text{WiFi}} = 80\ ms$ point, range in which WiFi is used as the feedback channel. For higher block sizes, the impact of the channel latency decreases when compared to the data transmission time. Still, the differences in performance are quite significant.

Above all, this practical use case showed how our framework can be used to model actual communication interfaces, providing a validation of our ideas and mechanisms in real scenario.

## 5.6 Conclusions

This chapter provided theoretical analysis and practical schemes to better understand the problem of transmitting a block of data over asymmetric half-duplex channels. Our mathematical models provide a framework to analyze a wide variety of practical schemes with any number of communication channels with asymmetries in the round trip delay, packet losses, and raw transmission rate per packet, as demonstrated with the practical use case. Our numerical results show not only that our coded approaches consistently outperform uncoded scheduling strategies by several fold, but that the performance of uncoded schemes varies dramatically depending on channel conditions. The latter makes some strategies better than others in different regions, which can complicate the overall system design. Our approaches require a simple logic and deliver a performance that is close to the optimal solution if we had full duplex channels, i.e., no feedback cost.

# Chapter 6

# Coded Multipath for Time-Varying Channels

In the previous two chapters, we looked at problems related to block data transmission, where data is considered useful as a whole. In those cases, the performance metric of interest is the total time taken to transmit the whole block. There are however, other kinds of applications that require a more granular delivery of individual data packets.

It is well known that wireless channels are subject to fading effects that induce time-varying signal quality. Inevitably, even temporary drops in quality have a negative effect on communications, leading to poor delay and energy performance per packet. On the energy side, opportunistic schemes can use information about channel conditions to improve the energy efficiency of the transmissions, albeit at the expense of extra delay incurred while waiting for good channel conditions. On the delay side, and in consonance with our claims, we believe that the increased demands for low latency, highly reliable, and high data rate services require systems to explore alternative methods beyond a single active wireless communication link to compensate for fading and time-varying conditions, and provide a more timely transmission of individual packets. Channel diversity is key here since the likelihood of having at least one channel in good condition, and with it the conditions to provide a more timely response, increases with the number of uncorrelated channels.

In this chapter, we study the problem of using multiple time-varying channels with potentially heterogeneous conditions to reduce per-packet delay and energy consumption while providing a high throughput, even in imperfect feedback scenarios. To that end, we propose schemes that combine opportunistic scheduling with network coding to (i) exploit link diversity to achieve better overall performance, and (ii) satisfy a wide range of energy and delay requirements, under realistic assumptions. We unify all of these aspects under a mathematical framework and validate our design through an extensive set of results.

## Problem Statement

In this chapter, we exploit the use of multiple communication channels to compensate for time-varying changes in the quality of individual channels, and therefore provide a high energy efficiency while minimizing the *queuing delay*.

**Queuing Delay** (Figure 1.6): Time since a packet enters the transmission buffer until it is seen by the receiver.

## Contributions

- *Decision Making Framework:* We pose the scheduling problem as a Partially Observable Markov Decision Process and characterize the proposed framework for links with different channel characteristics, transmission rates and energy costs, as well as imperfect feedback conditions and competing performance objectives.
- *Multi-Objective Optimization:* We design a tunable reward function that incorporates the relative importance between delay and energy, subject to an independent buffer stability component.
- *Opportunistic Coded Schemes:* We incorporate the generated policies into transmission schemes that combine online network coding and opportunistic scheduling. Ultimately, we propose heuristic schemes capable of coping with the uncertainty typical of real systems.
- *Extensive Analysis:* We validate the proposed solution through an extensive set of results that encompasses analysis of design parameters, system parameters and representative use cases. The results highlight the advantages of using multiple channels and the versatility of the proposed schemes.

## Publications

- A. Moreira and D. E. Lucani, “Opportunistic Online Network Coding for Multiple Wireless Channels: Send, Probe, or Wait,” *IEEE Transactions on Communications*, (Submitted).

## 6.1 Preliminaries

We first remind the general rules followed by our notation, as it is used extensively throughout this chapter. A random variable is represented as an uppercase letter. The set of values it can take is represented with the corresponding calligraphic letter. A realization of the random variable is represented with the corresponding lowercase letter. For instance, the set of possible values that $X$ can take is $\mathcal{X}$ and $x$ refers to a value taken by the random variable. For simplicity, we may write $P(x)$ as a shorthand for $P(X = x)$ when the random variable can be derived from the context. The Cartesian product of sets $\mathcal{X}$ and $\mathcal{Y}$ is denoted as $\mathcal{X} \times \mathcal{Y}$. The $n$-fold Cartesian product of a set $\mathcal{X}$ is denoted as $\mathcal{X}^n$. When $X$ and $Y$ are sets of random variables, and $Y \subseteq X$, we denote by $x_{\langle Y \rangle}$ as the assignment of the subset of values in $x$ to the respective variables in $Y$. Uppercase letters are also used to represent parameters, metrics and other variables in general. Since the framework is based on temporal models, some variables are instantiated for different values of time $t$. For a given variable $X$ we denote as $X_t$ its instance at time $t$. Moreover, we refer to the list of variables between $t_1$ and $t_2$ as $X_{t_1:t_2}$, which is empty if $t_1 > t_2$. Similarly, quantities corresponding to a particular channel are identified with the parenthesized channel index in superscript. For instance, $X^{(l)}$ is a quantity associated with channel $l$.

## 6.2 System Model

We consider the problem of relaying a stream of packets from a single source node to a single destination node over multiple wireless channels in a time-slotted system. Packets arrive to the source node from the application layer and are stored in a buffer waiting for transmission. A controller is responsible for scheduling transmissions by choosing the most appropriate moments and channels for transmitting packets. We consider a dedicated feedback channel that may be delayed and subject to packet losses. The system model is depicted in Figure 6.1 and the details of the system are presented in the following.

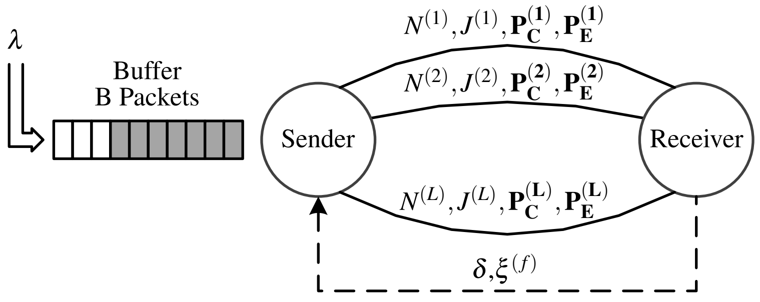

$\lambda$
Buffer
B Packets
Sender
Receiver
$N^{(1)}, J^{(1)}, \mathbf{P_C^{(1)}}, \mathbf{P_E^{(1)}}$
$N^{(2)}, J^{(2)}, \mathbf{P_C^{(2)}}, \mathbf{P_E^{(2)}}$
$N^{(L)}, J^{(L)}, \mathbf{P_C^{(L)}}, \mathbf{P_E^{(L)}}$
$\delta, \xi^{(f)}$

Figure 6.1: The system is composed by one sender and one receiver connected by $L$ time-varying channels plus one feedback channel. Packets arrive randomly to the sender at an average rate of $\lambda$ packets per slot and are stored in a buffer with capacity for $B$ packets.

### 6.2.1 Communication Model

We adopt a discrete-time communication model where time is divided in slots with duration $T_{slot}$. We further assume a late arrival queue model, where packet arrivals occur just before the end of the slot, after transmission of packets has been completed and feedback has been received. We assume that all events except for transmission have negligible duration and therefore are considered instantaneous. Transmission of packets starts at the beginning of each slot and lasts until the end of the slot. After transmission is finished and before the end of the slot the source listens for feedback. If feedback is received, the buffer is updated to create room for new packets, according to the *drop-when-seen* algorithm from [21], which removes packets from the buffer that have been reported as being *seen*. After that, new packet arrivals are handled. This implies that any packets arriving during a slot can only be serviced from the next slot onwards.

### 6.2.2 Coding

We assume that network coding is performed at the source node in an online fashion. One advantage of online network coding with respect to other codes, is that it does not require a fixed block of packets to operate. Instead, new packets can be included in the code as they arrive. This feature is useful in situations where the data to be transmitted is not available all at once. Also, packets can be removed from the code, and hence from the transmission buffer, as they are *seen* by the receiver, without needing to be decoded [21]. Therefore, a seen packet can be considered delivered without being necessarily decoded. We consider such online coding mechanism with combinations involving all packets in the buffer at a given time. Moreover, we assume that the field size is sufficiently large, such that every new coded packet is linearly independent from the previous ones with high probability. Under these assumptions, each packet delivered is innovative and allows the receiver to see one new packet.

### 6.2.3 Network Model

We consider a network consisting of a single source node (sender) and a single destination node (receiver) connected by $L$ independent wireless channels. Each channel $l$ represents an interface through which the source can transmit information at a rate of $N^{(l)}$ packets per slot with an energy cost of $J^{(l)}$ units per packet. We assume that the propagation delay is zero for every channel meaning that any packets successfully transmitted arrive instantly to the destination.

Given the wireless nature of the communication medium, we model the channels as Gilbert-Elliot (GE) channels. The Gilbert-Elliot channel is a particular case of a Finite State Markov Chain (FSMC) channel, a family of models that have been widely used to model wireless channels [45]. At any given time, an FSMC channel lies in one of a finite set of states. Transitions between states take place according to some stochastic process that is characterized by the probabilities of jumping from one state to the others. At each step the channel behaves as a packet erasure channel with an erasure probability that is dependent on the current underlying state. A major benefit of this model is that it provides a considerable level of abstraction that simplifies analysis of wireless systems while still capturing their intrinsic properties.

A Gilbert-Elliot channel can lie in one of two states, either *good* or *bad*. Let the state of channel $l$ at time $t$ be denoted by $C_t^{(l)}$ and the set of possible channel states be $\mathcal{C} = \{g, b\}$. The state transition probability function for channel $l$ is defined as

$$\mathbf{P}_{\mathbf{C}}^{(\mathbf{l})}(c', c) \triangleq P\left(C_{t+1}^{(l)} = c' \mid C_t^{(l)} = c\right). \tag{6.1}$$

For the considered two-state model, the transition probabilities for a single channel $l$ can be fully parameterized in terms of two probabilities $\theta^{(l,c)} = \mathbf{P}_{\mathbf{C}}^{(\mathbf{l})}(c, c)$, one for each state, defined as the coherence probabilities of channel $l$ in state $c$. Then, let the outcome of a single packet transmission over channel $l$ at time $t$ be denoted as $E_t^{(l)}$ and the set of possible outcomes as $\mathcal{E} = \{0, 1\}$. The outcome probability function for channel $l$ is defined as

$$\mathbf{P}_{\mathbf{E}}^{(\mathbf{l})}(e, c) \triangleq P\left(E_t^{(l)} = e \mid C_t^{(l)} = c\right). \tag{6.2}$$

For simplicity, we define $\xi^{(l,c)} = \mathbf{P}_{\mathbf{E}}^{(\mathbf{l})}(0, c)$ to represent the packet erasure probability of channel $l$ in state $c$. Here we assume that $\mathbf{P}_{\mathbf{C}}^{(\mathbf{l})}$ and $\mathbf{P}_{\mathbf{E}}^{(\mathbf{l})}$ are given for every channel $l$ and therefore each channel is completely characterized.

From these base parameters other metrics of interest can be derived. Particularly, if the Markov chain defined by $\mathbf{P}_{\mathbf{C}}^{(\mathbf{l})}$ has a single recurrent class which is also aperiodic, then it exists a limiting distribution $\pi^{(\mathbf{l})}$ over channel states, which is also a stationary distribution [39]. The individual entries $\pi^{(l,c)}$ can be interpreted as the limiting values for the probability of being in a particular state after a large number of steps, i.e., $\pi^{(l,c)} = \lim_{t\to\infty} P(C_t^{(l)} = c)$, which is independent of the initial state. Finally, the network capacity $K$ can be calculated as

$$K = \sum_{l=1}^{L} \sum_{c\in\mathcal{C}} \left(N^{(l)} \cdot \pi^{(l,c)} \cdot \left(1 - \xi^{(l,c)}\right)\right) \tag{6.3}$$

which establishes the maximum rate of packets that can be transmitted across the network in a sufficiently large time window.

### 6.2.4 Arrivals

We assume that packets arrive to the source node according to a Poisson process with a rate of $\lambda$ packets/slot. Naturally, the maximum long-term arrival rate $\lambda$ that the system is able to support properly is upper bounded by the network capacity $K$. Then, the arrival process may also be characterized in terms of the load imposed on the system, defined as $\bar{\lambda} = \lambda/K$.

### 6.2.5 Service

The controller decides how many coded packets to send through each channel at each time-slot. Therefore, the set of possible control actions for channel $l$ is simply $\mathcal{U}^{(l)} = \{0, \ldots, N^{(l)}\}$. The rules based on which the controller shall make its decisions are defined by transmission policies, which naturally depend on the performance requirements imposed. The calculation of optimal and heuristic transmission policies is treated in Section 6.3.4 for the case in which the system state is perfectly known and in Section 6.4.4 for the case when there is uncertainty about the system state.

### 6.2.6 Physical Queue

We assume that the source has a limited size buffer with capacity for $B$ packets. Upon arrival, new packets may be accepted if there is enough space in the buffer or discarded otherwise. Accepted packets are placed in the buffer forming a first-in-first-out (FIFO) queue. To make room for new packets, there are several possible strategies to manage which packets should be kept in the buffer. Here we consider the use of the *drop-when-seen* algorithm from [21], which is enabled by the considered online coding mechanism. With this strategy, the source drops a packet from the buffer when it gets confirmation that it was seen by the receiver, without needing the packet to be decoded. The described queue formed by the packets in the buffer is referred to as the *physical queue*.

### 6.2.7 Virtual Queue

In the absence of perfect information, there is a natural mismatch between the sender's knowledge and the true state of the receiver. In those circumstances, the sender will have more degrees of freedom in its buffer (physical queue) than the ones still missing from the receiver. To handle this knowledge gap, we borrow the notion of *virtual queue* from [21], where the virtual queue size represents the backlog in degrees of freedom between sender and receiver. Under perfect and instant feedback assumptions, the *drop-when-seen* algorithm ensures that the physical queue in the buffer tracks the virtual queue associated with the receiver state, and thus the physical queue size is minimized. With imperfect feedback, the physical queue lags the virtual queue as feedback is not always and readily available to update the sender's knowledge. Given the limited capacity of the buffer, the virtual queue size at any given time is at most $B$. Hence, the set of possible virtual queue states is $\mathcal{D} = \{0, \ldots, B\}$.

### 6.2.8 Feedback

Feedback from the receiver is delivered to the sender via a dedicated feedback channel with propagation delay $\delta$ and erasure probability $\xi^{(f)}$ both of which may not be negligible. We assume that feedback is sent by the receiver at the end of every time-slot, after any transmitted packets have arrived to the receiver. We consider two different types of acknowledgments, referred to as *differential* and *cumulative*.

Differential acknowledgments indicate how many packets were received over a channel in that slot. The set of values this acknowledgement can take for each channel $l$ is $\mathcal{U}^{(l)}$. The cumulative acknowledgment conveys the receiver's virtual queue state and hence takes values in $\mathcal{D}$. Each feedback packet contains the differential acknowledgment for each of the $L$ channels and the cumulative acknowledgment regarding the virtual queue state. Therefore, the set of possible feedback values is $\mathcal{F} = \mathcal{U}^{(1)} \times \cdots \times \mathcal{U}^{(L)} \times \mathcal{D}$.

These two types of acknowledgment provide complementary information. Cumulative acknowledgment is needed to compensate for lost feedback packets as it consistently provides full information regarding the virtual queue, whereas differential acknowledgments provide detail about individual transmissions by reporting the number of packets received through each channel.

## 6.3 Policies with Complete State Information

We now present a framework to determine optimal transmission policies under the assumption that the source node knows everything about the system at all the time. This knowledge encompasses both channel state and receiver state, hence feedback is not considered here. Although unreasonable from a practical perspective, this framework shall provide an optimal policy that will stand as a baseline of comparison for more realistic scenarios.

We leverage the Markov nature of the underlying model and pose the problem as a Markov Decision Process (MDP). This mathematical formulation considers a decision making agent that must choose the most appropriate action to take at each time step, considering the system state, its dynamics and different performance requirements. The mechanics of such decision making process can be formally described as follows. At each time step $t$, the system lies in some state $S_t$ belonging to the set of system states $\mathcal{S}$. Based on the current state, the agent performs action $A_t$ chosen from the set of possible actions $\mathcal{A}$. Depending on the current state and action performed, the agent receives a reward according to the reward function $R$, and the system evolves to a new state $S_{t+1}$ according to the transition probability function $\mathbf{P_S}$. Hence, an MDP can be characterized as a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathbf{P_S}, R)$ composed by a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a transition probability function $\mathbf{P_S} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \mapsto [0,1]$ and a reward function $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. The solution to the problem is a policy $\rho^* : \mathcal{S} \mapsto \mathcal{A}$ that returns the optimal action as a function of the state at each time. The relationships between the different variables can be represented using a Probabilistic Graph Model (PGM), shown in Figure 6.2.

### 6.3.1 States and Actions

The system state is composed by all channel states and the virtual queue state. Therefore, the set of system states is defined as $\mathcal{S} = \mathcal{D} \times \mathcal{C}^L$. At each slot, the controller must decide how many packets to send over each channel. Hence, the set of possible actions is the combination of all possible individual actions $\mathcal{A} = \mathcal{U}^{(1)} \times \cdots \times \mathcal{U}^{(L)}$.

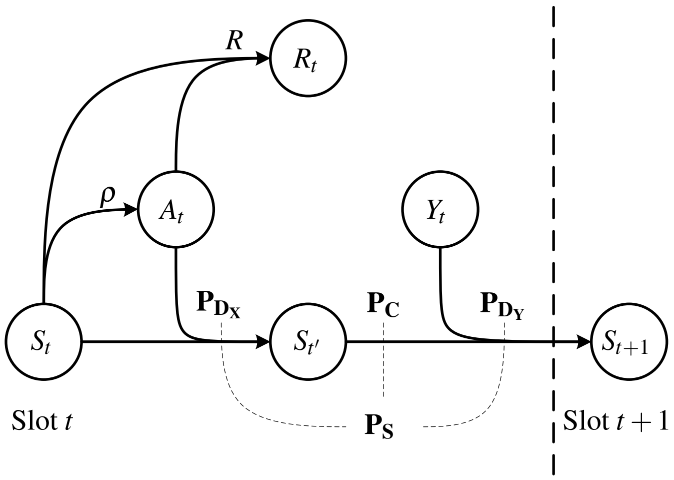


Figure 6.2: The MDP represented as a Dynamic Decision Network (DDN). Nodes represent variables and edges represent conditional dependencies between variables. The state $S_{t'}$ is introduced to simplify the formalisms involved, allowing to decompose the state transition probabilities $\mathbf{P_S}$ in several simpler components, namely queue state changes due to packet deliveries $\mathbf{P_{D_X}}$ and packet arrivals $\mathbf{P_{D_Y}}$, and channel state changes $\mathbf{P_C}$. Moreover, in the partially observable case, $S_{t'}$ will be the state reported by the receiver, corresponding to the state before the slot ends, where departures have taken place but the channel state changes and packet arrivals have not yet occurred.

### 6.3.2 Transition Model

The transition probability function $\mathbf{P}_{\mathcal{S}}(s',s,a)$ governs the evolution of the system state according to the actions that are performed in each state of the system. It defines the probability of jumping to state $s'$ if action $a$ is taken while in state $s$, which can be formally written as

$$\mathbf{P_S}(s',s,a) \triangleq P\left(S_{t+1} = s' \mid S_t = s, A_t = a\right). \tag{6.4}$$

Calculating the transition probabilities is simplified by the independence between the involved stochastic processes, namely channel state and queue state transitions. Additionally, transitions between channel states do not depend on the action taken and are characterized through the transition functions $\mathbf{P_C^{(l)}}$ defined in Section 6.2.3. We thus focus on the calculation of the transition probabilities for the virtual queue.

Let $X_t^{(l)}$ be a random variable indicating the number of packets delivered to the receiver over channel $l$ at slot $t$. Its probability conditioned on state $s$ and action $a$ is given by

$$\begin{aligned}\mathbf{P_X^{(l)}}(x,s,a) &\triangleq P\left(X^{(l)} = x \mid S = s, A = a\right)\\ &= P_{Binomial}\left(x, u, 1-\xi^{(l,c)}\right)\end{aligned} \tag{6.5}$$

where $u = a_{\langle U^{(l)} \rangle}$ and $c = s_{\langle C^{(l)} \rangle}$. Then, let $X = \sum_{l=1}^{L} X^{(l)}$ represent the total number of packets delivered to the receiver in a single slot. Since $X$ is the sum of independent random variables, its conditioned probability distribution is given by the convolution of the individual distributions [39]. Formally,

$$\begin{aligned} \mathbf{P_X}(\cdot, s, a) &\triangleq P(X \mid S = s, A = a) \\ &= \mathbf{P_X^{(1)}}(\cdot, s, a) * \cdots * \mathbf{P_X^{(L)}}(\cdot, s, a) \end{aligned} \tag{6.6}$$

where $\mathbf{P_X^{(l)}}(\cdot, s, a)$ denotes the probability distribution of $X^{(l)}$ conditioned on state $s$ and action $a$. Now let $Y$ denote the number of arrivals in a given slot which according to the system model follows a Poisson distribution with an arrival rate of $\lambda$ packets per slot, i.e. $Y \sim Poisson(\lambda)$. Using these distributions, let us decompose virtual queue transitions into departures and arrivals. The queue transitions due to departures are given by

$$\begin{aligned} \mathbf{P_{D_X}}(d, s, a) &\triangleq P(D = d \mid S = s, A = a) \\ &= \begin{cases} P\left(X = s_{\langle D \rangle} - d \mid S = s, A = a\right) & \text{if } d \neq 0 \\ P\left(X \geq s_{\langle D \rangle} - d \mid S = s, A = a\right) & \text{if } d = 0 \end{cases} \end{aligned} \tag{6.7}$$

Similarly, the queue transitions due to arrivals are given by

$$\begin{aligned} \mathbf{P_{D_Y}}(d, s) &\triangleq P(D = d \mid S = s) \\ &= \begin{cases} P\left(Y = d - s_{\langle D \rangle}\right) & \text{if } d \neq B \\ P\left(Y \geq d - s_{\langle D \rangle}\right) & \text{if } d = B \end{cases} \end{aligned} \tag{6.8}$$

To combine these two components, one must note that the queue dynamics depend on the arrival model. As described in Section 6.2, a late arrival queue model is assumed, where packet arrivals occur after packet departures. Accordingly, the full queue transitions are given by

$$\begin{aligned} \mathbf{P_D}(d, s, a) &\triangleq P(D_{t+1} = d \mid S_t = s, A_t = a) \\ &= \sum_{s'} \mathbf{P_{D_Y}}(d, s') \mathbf{P_{D_X}}(s'_{\langle D \rangle}, s, a) \end{aligned} \tag{6.9}$$

where the two processes are coupled through an intermediate state $s'$ that corresponds to the point in the slot after departures and before arrivals.

Finally, the full state transition probability function $\mathbf{P}_{\mathcal{S}}$ defined in Equation 6.4 can be calculated as

$$\mathbf{P_S}(s', s, a) = \mathbf{P_D}\left(s'_{\langle D \rangle}, s, a\right) \prod_{l=1}^{L} \mathbf{P_C^{(l)}}\left(s'_{\langle C \rangle}, s_{\langle C \rangle}\right) \tag{6.10}$$

The calculation is simplified as the individual transition functions for the virtual queue and channel states are independent when conditioned on the current state $s$, and thus can be simply multiplied together.

### 6.3.3 Rewards

The choice of the most appropriate action requires a measure of the quality associated with different actions. This is provided by the reward function $R(s,a)$ which assigns an immediate reward for executing action $a$ in state $s$. Since actions are intrinsically driven by rewards, the design of the reward function is of paramount importance for the performance of the system.

In this work, we focus on two fundamental performance metrics that are delay and energy, and aim to provide policies that satisfy different articulations of these metrics. Still, any trade-offs should be achieved with no penalty in queue stability. With a limited buffer, instability does not lead to unbounded delays but rather to a large number of discarded packets. Hence, we take into account packet discard rate as a third metric. Unlike the other metrics however, this last one is the result of a more rigid constraint, and thus should be kept close to the minimum. We note that all the considered metrics actually represent costs, represented in this framework as negative rewards. The reward component of delay is defined as

$$R_D(s) = -\frac{s_{\langle D \rangle}}{K}, \tag{6.11}$$

where the virtual queue size associated with the current state is divided by the network capacity to provide an estimate of the waiting times experienced by a newly arriving packet. Although the immediate reward does not depend on the action, different actions lead to different future rewards, as dictated by the system dynamics. For energy, we consider the transmission energy associated with an action $a$. Hence, the energy component of the reward is given by

$$R_J(a) = -\sum_l J^{(l)} \cdot a_{\langle U^{(l)} \rangle}. \tag{6.12}$$

The packet discard rate is measured by the expected number of packets discarded when the state is $s$ and action $a$ is performed. Denoting the number of packets accepted into the system by $W$, this reward component can be calculated as

$$\begin{aligned} R_B(s,a) &= \mathbb{E}[W] - \mathbb{E}[Y] \\ &= \sum_{w=0}^{B} \sum_{s' \in \mathcal{S}} w \mathbf{P}_{\mathbf{D_Y}}(s'_{\langle D \rangle} + w, s') \mathbf{P}_{\mathbf{D_X}}(s'_{\langle D \rangle}, s, a) - \lambda \end{aligned} \tag{6.13}$$

Putting together the different components composes a multi-objective optimization problem. We adopt the weighted sum method to combine the different reward components into a single reward function that fits nicely into the MDP framework. The reward function is composed as

$$R(s,a) = \beta R_B(s,a) + \alpha R_D(s) + (1-\alpha)R_J(a). \tag{6.14}$$

The relevance of the different metrics is controlled through the design parameters $\alpha$ and $\beta$. The balance between delay and energy is controlled by varying $\alpha$ between 0 and 1, whereas the penalty assigned to discarded packets is controlled by $\beta$. Among the drawbacks of the weighted sum method is the lack of intuition provided by the weights. In general, a system designer needs to adjust these to achieve some desired trade-off between the competing objectives. In Section 6.5.4, we provide some intuition, by showing results for a wide range of these design parameters.

### 6.3.4 Optimal Policy

We consider sequential decision making over an infinite time horizon and seek to define the policy that determines the best course of action over time. For infinite horizon problems, the optimal policy $\rho^*$ typically does not depend on the time at which the decision is made, and therefore is said to be stationary [46]. A stationary policy can be defined as a function $\rho : \mathcal{S} \mapsto \mathcal{A}$ that returns an action $a_t = \rho(s_t)$ depending only on the state $s_t$ for every time $t$. The derivation of the optimal policy is driven by the principle of maximum expected utility (MEU) which states that a rational decision maker should pick actions that maximize the expected utility [47]. The utility can be regarded as the value associated with a sequence of states and actions. By basing decisions on utility, policies maximize long-term benefits rather than just immediate rewards. Among the different ways to define the utility, we adopt the average reward per slot, defined as

$$V = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} R(s_t, a_t). \tag{6.15}$$

According to this definition, the expected utility $V^\rho$ associated with policy $\rho$ is defined as

$$V^\rho \triangleq \lim_{T\to\infty} \frac{1}{T} \mathbb{E}\left[\sum_{t=0}^{\infty} R(S_t, \rho(S_t))\right] \tag{6.16}$$

Following the principle of maximum expected utility, the optimal policy $\rho^*$ is the one that satisfies $\rho^* = \arg\max_\rho V^\rho$. Now, let us denote the optimal average reward as $g^* = V^{\rho^*}$, and define a new quantity $h^*(s)$ that can be regarded as the optimal differential reward when starting in state $s$. These two components satisfy the Bellman equation [46]

$$g^* + h^*(s) = \max_{a\in\mathcal{A}} \left\{ R(s,a) + \sum_{s'\in\mathcal{S}} \mathbf{P_S}(s',s,a) h^*(s') \right\}, \quad \forall s \in \mathcal{S} \tag{6.17}$$

whose solution is an optimal stationary policy. Among different methods, the Bellman equation can be solved using the value iteration algorithm, which returns the optimal policy $\rho^*$ and the corresponding $g^*$ and $h^*$. Before concluding, let us define the Q-function as

$$Q(s,a) \triangleq R(s,a) + \sum_{s' \in \mathcal{S}} \mathbf{P_S}(s',s,a)h^*(s') \tag{6.18}$$

which can be regarded as the utility of some state-action pair, and shall be the base for the heuristics defined further ahead.

## 6.4 Acting with Partial Information

Information about the system state is a major asset in any decision making process, generally leading to better decisions. However, immediate and error free information is hard or even impossible to obtain, thus raising extra challenges. Consequently, we evolve our formulation into a more realistic setting where the sender does not know the system state. Instead, the sender is intended to estimate the state using its own local information and external information provided by feedback.

We model this new problem as a Partially Observable Markov Decision Process (POMDP). Under this new formulation, a set of *hidden* variables (state) must be estimated based on a different set of *observable* variables (observations). The relationships between hidden and observable variables can be represented with a Probabilistic Graphical Model shown in Figure 6.3.

Formally, a POMDP can be characterized as a 6-tuple $(\mathcal{O}, \mathcal{S}, \mathcal{A}, \mathbf{P_O}, \mathbf{P}_\mathcal{S}, R)$ where two new elements are added with respect to the MDP, namely the set of observations $\mathcal{O}$ and the observation probability function $\mathbf{P_O} : \mathcal{O} \times \mathcal{S} \times \mathcal{A} \mapsto [0,1]$.

### 6.4.1 Observations

In the considered system, observations are provided in the form of feedback. For an observation $O_t$ corresponding to slot $t$, we have that $O_t \in \mathcal{F}$ if feedback was retrieved successfully or $O_t = \varnothing$ if feedback was lost. Hence, the set of possible observations is $\mathcal{O} = \mathcal{F} \cup \varnothing$, where the null element $\varnothing$ is used to explicitly denote that feedback was lost.

### 6.4.2 Observation Model

The remaining piece left to characterize is the observation probability function, defined as

$$\mathbf{P_O}(o,s,a) \triangleq P(O_t = o \mid S_{t'} = s, A_t = a). \tag{6.19}$$

To simplify the calculation, we decompose the observation probability in terms of its components. As defined in Section 6.2.8, feedback conveys two kinds of information classified as *differential* and

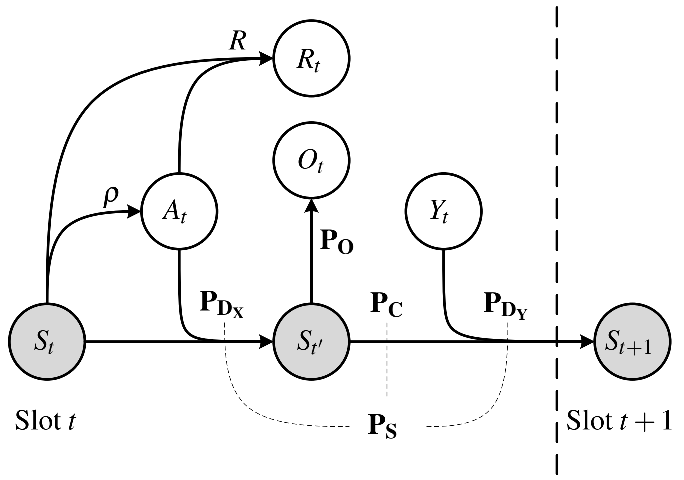


Figure 6.3: Probabilistic Graph Model of the POMDP process. Nodes represent variables and edges represent conditional dependencies between variables. Shaded nodes represent variables whose value is not known. Observations are made based on state $S_{t'}$, i.e., the receiver reports the state of its virtual queue before new packet arrivals.

*cumulative*. Using the fact that the different components are independent from each other when conditioned on specific variables, the full observation model can be assembled as

$$\mathbf{P_O}(o,s,a) = \begin{cases} 0 & ,o_{\langle D\rangle} \neq s_{\langle D\rangle} \\ \left(1-\xi^{(f)}\right)\prod_l \mathbf{P}_{\mathbf{X}}^{(\mathbf{l})}\left(o_{\langle X^{(l)}\rangle}, s, a\right) & ,o_{\langle D\rangle} = s_{\langle D\rangle} \\ \xi^{(f)} & ,o = \varnothing \end{cases} \tag{6.20}$$

The observation model allows to calculate the likelihood of each state given an observation. Note that the probability of receiving an empty observation is the same regardless of the state. This means that if feedback is lost, the likelihood of each state given the observation is the same for all states, and thus the observation model provides no information regarding the system state, as expected. On the other hand, when feedback is successful, the observed content makes some states more likely than others. The topic of knowledge update is addressed in detail in the following.

### 6.4.3 Knowledge Update

We now address the problem of tracking the system state from partial information. Under the proposed probabilistic framework, this problem can be seen as one of probabilistic inference, where the structure of the underlying PGM can be leveraged to efficiently calculate the desired probabilities. Still, the case of delayed and missing observations has not been as frequently considered. Although more complex, we consider such setting as it also makes the model more realistic. We recall that due to the characteristics of the feedback channel, acknowledgments are delayed by $\delta$ slots. As a result, at the beginning of slot $t$

only observations regarding transmissions up to time $t-\delta-1$ are available. Concerning lost feedback packets, we recall that these simply correspond to ambiguous null observations. We start our treatment by introducing some concepts that underlie the notion of knowledge.

**Definition 5** (Delayed Time)**.** *For the sake of notation simplicity, let $\tau = t-\delta$ denote time delayed by $\delta$ slots.*

**Definition 6** (Belief State)**.** *The belief state $z_t$ at time $t$ is defined as the probability distribution over the system state $S_t$ at time $t$ given the information available until then. Formally, it is defined as*

$$z_t \triangleq P(S_t \mid o_{1:\tau-1}, a_{1:t-1}, y_{1:t-1}, z_0), \tag{6.21}$$

*where $z_0$ is the initial belief state which holds the probability distribution over system states at the beginning of the process.*

**Definition 7** (Filtered Belief State)**.** *The filtered belief state $\bar{z}_t$ at time $t$ is defined as the probability distribution over the system state $S_\tau$ at time $\tau$ given the information available until then. Formally, it is defined as*

$$\bar{z}_t \triangleq \begin{cases} P(S_\tau \mid o_{1:\tau-1}, a_{1:\tau-1}, y_{1:\tau-1}, z_0), & t > \delta \\ z_0, & t \leq \delta. \end{cases} \tag{6.22}$$

*and represents the belief on the state at $\delta$ slots ago. Basically, it is the most recent belief that incorporates all possible past information up to that point.*

**Definition 8** (Information State)**.** *The information state $i_t$ at time $t$ comprises all information about the system until that time, and that is relevant for the selection of the optimal action. Formally, it is defined as*

$$i_t \triangleq (o_{1:\tau-1}, a_{1:t-1}, y_{1:t-1}, z_0). \tag{6.23}$$

A major problem with this representation of the information state is that it grows indefinitely with time, as it comprises the whole history of events. Fortunately, there is a more condensed way to represent all information relevant to the decision making process. Such representation is based on the fact that the filtered belief state $\bar{z}_t$ is a sufficient statistic for all information up to time $\tau$. Based on this, we can define the sufficient information state as follows.

**Definition 9** (Sufficient Information State)**.** *The sufficient information state is a compressed representation of the information state that holds all information sufficient to choose the optimal action. Formally, the sufficient representation of the information state $i_t$ at time $t$ is*

$$i_t = (a_{\tau:t-1}, y_{\tau:t-1}, \bar{z}_\tau). \tag{6.24}$$

*Represented in this form, the information state has fixed length, comprising a sequence of $\delta$ actions, a sequence of $\delta$ arrivals, and the belief at time $\tau$.*

**Remark 2** (Instantaneous Feedback)**.** *From Definition 9 it follows that in the case of instantaneous feedback, the belief state alone contains all information necessary for selecting the optimal action. This matches the results found in literature for non-delayed observations.*

Having established these concepts, we now focus on how to update the information state over time. Since retaining the last $\delta$ actions and $\delta$ arrivals is straightforward, the core task is to update the belief. According to Definition 6, the belief state relies on the initial belief and all events since the beginning of the process. As time increases, propagating the initial belief over time for each update becomes unfeasible. Fortunately, there is a more efficient solution that relies on the sufficient statistic property of the filtered belief state, as presented below.

**Lemma 3** (Filtered Belief Update)**.** *The filtered belief $\bar{z}_{t+1}$ can be obtained recursively from the previous filtered belief $\bar{z}_t$, controls $a_t$ and disturbances $y_t$ from thereon, as*

$$\bar{z}_{t+1} = \alpha \sum_{s_{\tau'} \in \mathcal{S}} P(o_\tau \mid s_{\tau'}, a_\tau) \sum_{s_\tau \in \mathcal{S}} P(S_{\tau+1} \mid s_{\tau'}, y_\tau) P(s_{\tau'} \mid s_\tau, a_\tau) \bar{z}_t \tag{6.25}$$

*where $\alpha$ can be regarded as a normalizing constant that makes the probability sum to one.*

When observations are delayed, the belief $z_t$ on the present state must be predicted, which can be done by propagating the latest filtered belief $\bar{z}_t$ according to the system dynamics, as presented below.

**Lemma 4** (Belief Prediction)**.** *The belief on the present state $z_t$ can be obtained from the filtered belief $\bar{z}_t$, controls $a_{\tau:t-1}$ and disturbances $y_{\tau:t-1}$ from thereon, as*

$$z_t = \sum_{s_t \in \mathcal{S}} \sum_{s_{t-1} \in \mathcal{S}} \cdots \sum_{s_{\tau+1} \in \mathcal{S}} \sum_{s_\tau \in \mathcal{S}} P(S_t \mid s_{t-1}, a_{t-1}, y_{t-1}) \times \cdots \times P(s_{\tau+1} \mid s_\tau, a_\tau, y_\tau) \bar{z}_t \tag{6.26}$$

*where the filtered belief $\bar{z}_t$ is propagated $\delta$ steps according to the conditioned state transition probabilities.*

The update equations expressed in Lemma 3 and Lemma 4 can be derived by applying Bayesian inference techniques to the PGM in Figure 6.3. The filtering step uses the new observation to refine the belief according to the observation model, whereas the prediction step propagates the belief $\delta$ steps ahead according to the transition model. These mechanisms allow to update the belief state with a computational complexity that is constant over time.

### 6.4.4 Heuristic Policies Under Incomplete Information

A fundamental property disclosed by the update procedure is that every new belief state depends only on the previous belief state and variables from thereon. It follows that the sequence of information states obeys to a transition probability function of the form

$$\mathbf{P_I}(i', i, a) \triangleq P(I_{t+1} = i' \mid I_t = i, A_t = a, Y_t = y). \tag{6.27}$$

Also, it is possible to define a function that assigns a reward to each information state and action pair as

$$R_I(i, a) \triangleq \mathbb{E}[R(s, a) \mid I = i, A = a] \tag{6.28}$$

Hence, the sequence of information states forms itself a controlled Markov process, where each new information state depends on the previous information state and the action performed. This insight allows us to define a Markov decision process over the information states, referred to as Information State MDP. In general, a discrete state space POMDP can be converted to an equivalent MDP in the corresponding information state space [46].

Based on this, one could consider using the algorithms available for MDPs to solve this new problem. However, the information state is a continuous variable taking on an infinite number of values. For this reason, the same algorithms are not applicable and several methods have been specifically developed to solve POMDP problems [48]. It is known however, that finding the optimal solution for a POMDP problem is in general hard [49].

In order to provide more applicable solutions, we focus on heuristic methods. Particularly, we present heuristics that rely on the optimal policy $\rho^*$ for the completely observable case, which can be pre-computed at system initialization. In the same way that $\rho^*$ specifies an action given the current state, the heuristics shall specify an action given the current belief state. Hence, the heuristic policies are functions $\rho_Z : \mathcal{Z} \mapsto \mathcal{A}$ that assign an action to each belief state according to some criteria. Below we propose and describe the considered set of heuristics.

#### 6.4.4.1 Maximum Expected Utility (Q-MDP)

This heuristic chooses the action based on its expected utility as given by the $Q$-function. It is defined as

$$\rho_{\mathrm{Q}}(z) \triangleq \arg\max_a \left( \sum_s z(s) \cdot Q(s, a) \right) \tag{6.29}$$

For a given state, the $Q$-function returns the utility of each action, assuming that optimal control will be performed from thereon. Interestingly, this heuristic would be optimal if the state would be known after the current step. Since that is not the case, this heuristic will be sub-optimal.

#### 6.4.4.2 Dual-Mode (DM)

The previous heuristic erroneously assumes that uncertainty will disappear after the current step, hence it does not perform actions for the sole purpose of gathering information. When the state is not known however, it might be worth to perform actions just to reduce the uncertainty about the system state, in turn leading to better decisions in the future. This creates an interplay between reward reaping and information gathering. A simple solution to this problem is to employ a dual-mode heuristic policy that switches between exploitation and exploration oriented policies depending on the level of uncertainty at each step.

In order to apply these ideas to our problem, let us first understand the main sources of uncertainty. As already described, feedback is assumed to be automatically sent by the receiver in every time slot, hence its reception does not depend on the action taken. However, channel state information may be ambiguous depending on the action. If the sender does not transmit over a channel, then the probability of observing zero packets delivered is always equal to one regardless of the channel state. Therefore, the sender should probe the channel by transmitting at least one packet in order to obtain channel state information, as would be expected. Still, it is necessary to decide when to perform the probing.

The idea of our DM heuristic is to force a transmission over channel $l$ when the difference between its channel state belief, denoted as $\phi_{\mathbf{t}}^{(\mathbf{l})}$, and its limiting distribution $\pi^{(\mathbf{l})}$ come below a given threshold $\eta^{(l)}$. The individual channel state belief $\phi_{\mathbf{t}}^{(\mathbf{l})}$ for a given channel $l$ can be obtained by marginalizing the complete belief $z_t$ over every other state. In the absence of external information, the channel state belief tends to the corresponding limiting distribution $\pi^{(\mathbf{l})}$, which corresponds to the least information that one can have about the channel state. To measure the difference between these two distributions, we use the Kullback-Leibler divergence, where the divergence between distributions $P$ and $Q$ is defined as

$$D_{\mathrm{KL}}(P \,||\, Q) = \sum_i P(i) \log_2 \frac{P(i)}{Q(i)} \text{ [bits]}. \tag{6.30}$$

Based on this, our DM policy is defined as

$$\rho_{\mathrm{DM}}^{(l)}(z) = \begin{cases} \rho_{\mathrm{Q}}^{(l)}(z) & , D_{\mathrm{KL}}(\phi_{\mathbf{t}}^{(\mathbf{l})} \,||\, \pi^{(\mathbf{l})}) > \eta^{(l)} \\ 1 & , D_{\mathrm{KL}}(\phi_{\mathbf{t}}^{(\mathbf{l})} \,||\, \pi^{(\mathbf{l})}) \leq \eta^{(l)} \end{cases} \tag{6.31}$$

For practical purposes, we assume that probing is allowed only when the expected number of missing packets, i.e., the expected virtual queue size, is at least one.

## 6.5 Simulation Results

We now evaluate, analyze and compare the performance of different transmission schemes over a wide range of conditions. We start by defining the base configuration of the system, whose parameters are then overridden as necessary for each experiment. Afterwards follows a descriptive list of the metrics used for evaluation as well as the transmission schemes considered. The remaining sections present and discuss the results.

### 6.5.1 Base System Configuration

The source node has a buffer with capacity for $B = 60$ packets and $L = 2$ interfaces available for communication. The two communication channels have equal transmission rates of $N^{(l)} = 1$ packets per slot and equal transmission costs of $J^{(l)} = 1$ energy units per packet. The packet erasure rates are $\xi^{(l,b)} = 0.90$ in the bad state and $\xi^{(l,g)} = 0.10$ in the good state, for both channels. The coherence probabilities are the same across channels and channel states and equal to $\theta^{(l,c)} = 0.98$. These channel parameters translate into a uniform channel state limiting distribution $\pi^{(\mathbf{l})} = \begin{bmatrix} 0.50 & 0.50 \end{bmatrix}$ for both channels, and a network capacity of $K = 1$ packets per slot. The feedback channel $f$ has no delay and is error free, i.e., $\delta = 0$ and $\xi^{(f)} = 0$. The packet arrival process is specified in terms of the load $\bar{\lambda}$ ranging from 5% to 95%. These values are summarized in Table 6.1 for ease of reference. The results are obtained via simulation, with each data point obtained by simulating the system over $10^6/\lambda$ slots.

### 6.5.2 Performance Metrics

Performance is evaluated in terms of the following three fundamental metrics.

- *Delay:* This metric corresponds to the average number of slots elapsed between a packet entering the buffer and it being *seen* by the receiver[1].
- *Energy:* This metric corresponds to the average number of energy units spent per slot, taken over all channels.
- *Discard Rate:* This metric corresponds to the percentage of packets that are discarded due to lack of buffer capacity.

[1]The delay metric considered in this chapter does not incorporate the decoding delay. Analysis of the decoding delay is a problem on its own, especially without perfect feedback assumptions. This problem is addressed in more detail in Chapter 7 for the specific case of in-order delivery of packets.

Table 6.1: Base system configuration.

| | Channel 1 | Channel 2 |
|---|---|---|
| $N^{(l)}$ | 1 | 1 |
| $J^{(l)}$ | 1 | 1 |
| $\xi^{(l,g)}$ | 0.10 | 0.10 |
| $\xi^{(l,b)}$ | 0.90 | 0.90 |
| $\mathbf{P}_{\mathbf{C}}^{(\mathbf{l})}(g,g)$ | 0.98 | 0.98 |
| $\mathbf{P}_{\mathbf{C}}^{(\mathbf{l})}(b,b)$ | 0.98 | 0.98 |
| $K$ | 1 | |
| $\pi$ | $\begin{bmatrix}0.25 & 0.25 & 0.25 & 0.25\end{bmatrix}$ | |

| | Feedback |
|---|---|
| $\delta$ | 0 |
| $\xi^{(f)}$ | 0 |

| | System |
|---|---|
| $B$ | 60 |
| $\bar{\lambda}$ | 5%–95% |

### 6.5.3 Transmission Schemes

Below follows the list of transmission schemes considered along with their description. The first three schemes assume complete state information, whereas the latter three work under the partially observable system paradigm. Also, the first two schemes mainly serve to establish lower and upper bounds on performance, thus providing a baseline of comparison for the other schemes.

- *MF – Minimum Flow:* This pseudo-scheme corresponds to the probabilistic policy obtained with the linear programming formulation in [33]. The term "pseudo" refers to the fact that the scheme is not effectively evaluated as a scheme since it does not conform to some of our assumptions. Instead, the referred formulation is used to yield a theoretical *lower bound* on the bare minimum energy required to service an incoming flow of packets, disregarding delay or buffer limitations.

- *FT – Full Transmission:* This scheme transmits as many packets as possible over each channel, regardless of the channel state, and given that the receiver is missing packets at the beginning of the slot. This scheme establishes a *lower bound* on the achievable *packet discard rate*, as well as an *upper bound* on the *energy* required to achieve it. It also poses a theoretical lower bound on *delay* under infinite buffer assumption. For finite buffer and imperfect feedback conditions, lower delay may be achieved as less packets are effectively serviced.

- *MDP – Markov Decision Process:* Transmissions scheduled with the MDP policy.

- *Q-MDP – Maximum Expected Value:* Transmissions scheduled with the Q-MDP policy.
- *DM – Dual Mode:* Transmissions scheduled with the DM policy.
- *DM-VP – Dual Mode with Void Probes:* This scheme operates in the same manner as the DM scheme with the difference that probing packets do not carry data, but still take time to be transmitted and are still acknowledged. Because of this, any differences in delay performance are achieved due to better information and not because data is opportunistically delivered. Similarly, probing packets are also assumed to consume no energy, to assess the energy performance benefits due to better information.

### 6.5.4 Navigating the Delay-Energy Region

We start by analyzing the effect of the two primary design parameters, $\alpha$ and $\beta$, on the performance. The main goal is to provide insight into the performance trade-off regions that are possible to achieve with the proposed framework over load levels ranging from 5% to 95%. For this analysis we consider a completely observable system and consider only the MDP policy.

Let us start by analyzing the performance as $\beta$ is varied and $\alpha$ is kept equal to zero. The results for the packet discard rate and energy are shown in Figure 6.4. For the considered range of $\beta$ values, the packet discard rate is very close to the minimum given by the FT scheme. For low and and intermediate loads, that can be achieved with an energy expenditure that is very close to the minimum given by the MF scheme. For high loads, some more energy must be expended to keep the number of discarded

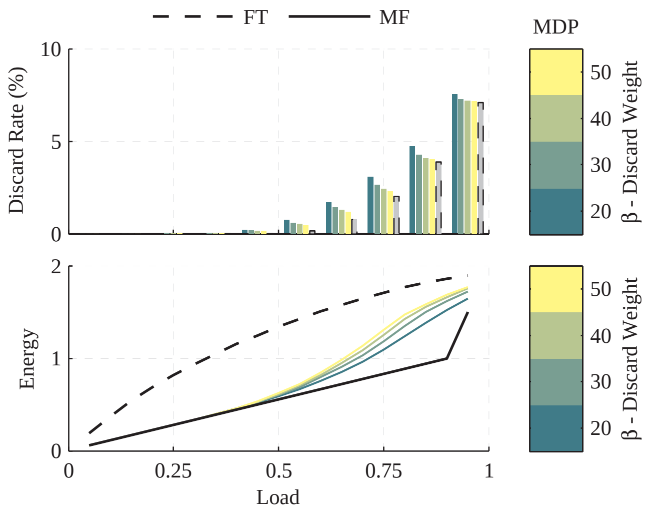


Figure 6.4: Performance trade-off between packet discard rate and energy using different values of $\beta$ for the MDP scheme. The base configuration has been used with load varying from 5% to 95%. Moreover, we set $\alpha = 0$ to keep delay out of consideration.

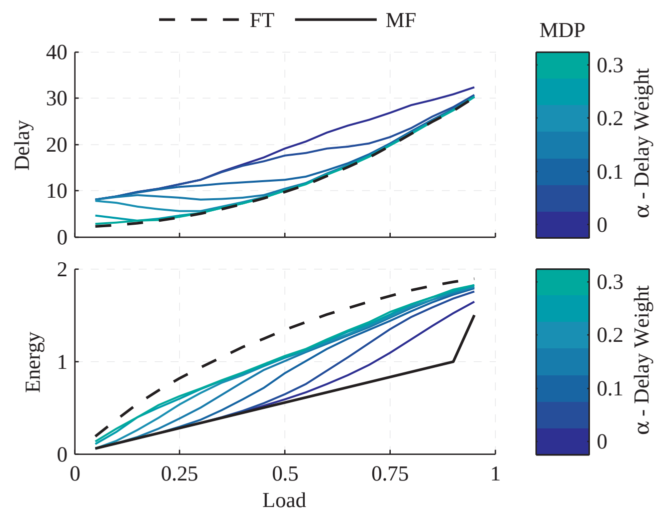


Figure 6.5: Performance trade-off between delay and energy using $\beta = 20$ and different values of $\alpha$ for the MDP scheme. The base configuration has been used with load varying from 5% to 95%.

packets low. When compared to the FT scheme, energy savings above 3x and 2x are achieved, for low and intermediate loads respectively, and even for high loads there is still some energy reduction. This shows that it is possible to save energy while keeping the rate of discarded packets close to the minimum. From hereon, we adopt $\beta = 20$ as it provides a low energy baseline while still withstanding the incoming packet flow.

With minimum service ensured, we now analyze how one can trade delay for energy by tuning the $\alpha$ knob. The results for delay and energy are shown in Figure 6.5. With $\alpha = 0$ we start back from the previous configuration, with the energy curve close to the minimum for most of the considered operation range. In this case, there is a noticeable price to pay in delay that is within 3x and 2x of the minimum for low and intermediate loads respectively. As $\alpha$ is increased, the delay curves go down whereas the energy curves go up, as expected. Remarkably, the delay curve for $\alpha = 0.30$ nearly overlaps that of FT with an energy reduction around 25% for low and intermediate loads compared to the same scheme. This energy reduction can be explained in part by the MDP scheme accounting for situations where the receiver is missing a single packet and not transmitting over one of the channels, unlike the FT scheme.

### 6.5.5 The Value of Information

In this section, we analyze the partially observable case and investigate how information impacts the performance of the system. For that, we compare the non-information-gathering Q-MDP heuristic, with the information-gathering DM-VP heuristic. The reason for considering DM-VM instead of DM, is that DM-VM does not send data on its probes, allowing to assess the isolated impact information on performance, rather than the combined effect of information gathering and data delivery accomplished by DM.

Although a better state estimation should yield a performance that is closest to the fully observable case, it comes at a cost. Not only feedback must be received but also channel state estimation requires packets to be transmitted for feedback to be informative. This very important fact poses a problem to energy conservative policies that refrain as much as possible from transmitting. Thus, achieving good performance with energy conservative policies is a challenging task.

Figure 6.6 for $\alpha = 0.05$ shows that Q-MDP has a worse delay performance than MDP for low loads (were the system observable). The reason for this is that the queue size and delay weight for low loads produce a utility that is not high enough to trigger transmissions in the face of uncertainty. That is, Q-MDP does not transmit unless it has enough confidence that the channel state is good, which may not be reached unless packets are transmitted. Q-MDP only re-starts transmissions, even in a bad state, when the queue size is large enough resulting in a high delay. A better solution is to actively search for information and this is where the dual-mode schemes come in. Looking at the results, the delay of the DM-VP scheme decreases as the divergence threshold, and in turn the probing frequency,

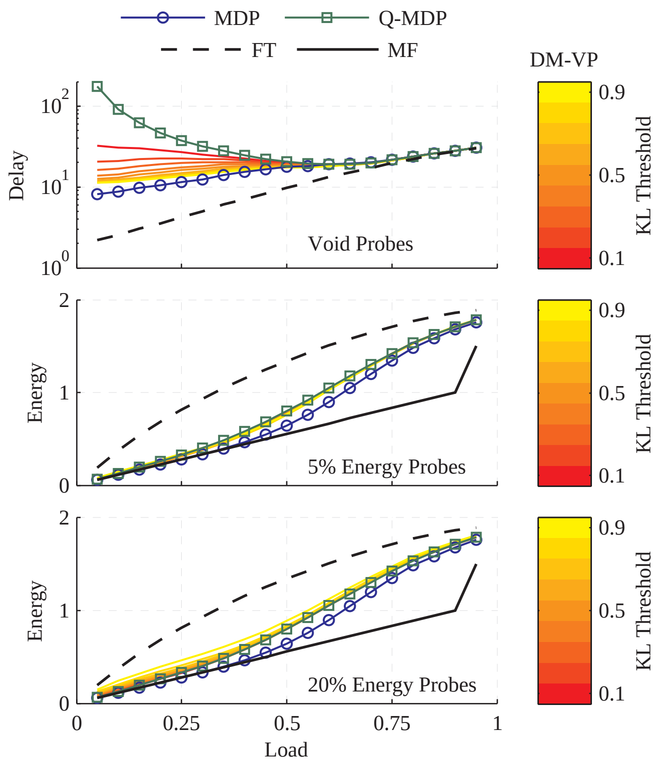


Figure 6.6: Performance comparison between different schemes for completely observable and partially observable systems using $\beta = 20$, $\alpha = 0.05$ and different symmetric values of $\eta^{(1)} = \eta^{(2)}$ for the DM-VP scheme. The base configuration has been used with load varying from 5% to 95%.

increases. Eventually, DM-VP has a delay performance that is close to the fully observable case (MDP) and this boost in performance is due to better state estimation. Since DM-VP does not deliver data when probing, it would be unfair to consider the same energy for probe and data packets. Rather than considering no energy consumption whatsoever, we show two different energy profiles, where probe packets consume 5% and 20% of the energy of a normal data packet. For 5% energy probes, the estimation benefits mostly outweigh the probing costs, resulting in energy gains when compared to the Q-MDP scheme. For 20% energy probes, the DM-VP scheme pays a price that increases with the probing frequency. Nonetheless, such price can be considered acceptable given the significant improvement in delay performance. This illustrates why optimal POMDP solutions are also information gathering and not just reward oriented.

To gain better insight, consider the time snapshots in Figure 6.7 representing the estimation of each channel, obtained with $\alpha = 0.05$, $\eta^{(l)} = 0.25$ and $\bar{\lambda} = 25\%$. The Y-Axes can be seen as the belief of a channel being in one of the two possible states. Since the MDP scheme has complete information, its profile corresponds to the real state. On the other hand, the belief associated with the limiting distribution $\pi^{(\mathbf{l})}$ corresponds to the level of least information. When estimation leans towards the good state, both Q-MDP and DM-VP transmit packets and thus are able to keep good estimates and quickly detect transitions to the bad state, as happened for channel one. On the other hand, when estimation leans towards the bad state, Q-MDP avoids transmissions, taking a long time to detect transitions to the good state or even completely missing them, as happened for channel two at $t = 200$. This shortcoming

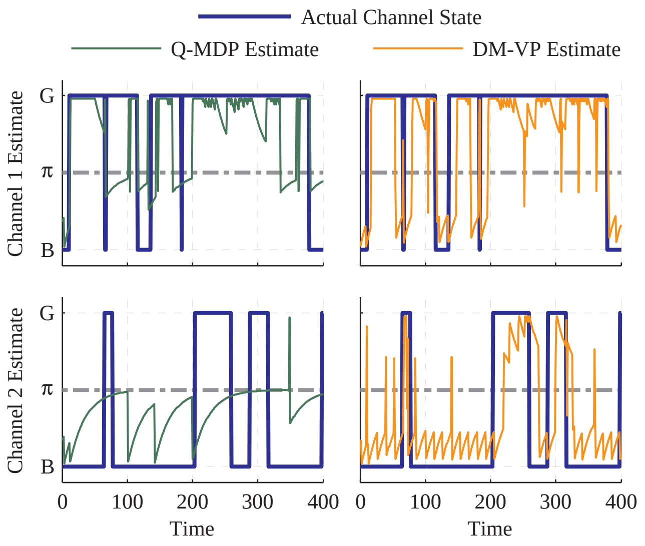


Figure 6.7: Snapshot of channel state estimates computed by different schemes. Each row corresponds to one channel, and each column to one heuristic scheme. The limiting distribution estimate $\pi^{(\mathbf{l})}$ indicates the level of least information. The base configuration has been used with load set to 25%. The design parameters were set as $\beta = 20$, $\alpha = 0.05$ and $\eta^{(1)} = \eta^{(2)} = 0.25$.

is overcome by DM-VP which probes the channel when the belief approaches the limiting distribution. As a result, even short transitions from the bad to the good state in channel two were detected by the dual-mode scheme. Nonetheless, Q-MDP is expected to perform well in settings that are not so energy conservative.

### 6.5.6 Responsiveness vs. Predictability

The varying nature of the channels considered lends interesting properties to the problem at hand. On the one hand, fast changing channels have the potential to provide better performance through increased responsiveness. On the other hand, fast changing channels are harder to predict making it difficult to take the appropriate actions.

Results for $\bar{\lambda} = 25\%$ and $\alpha = 0.05$ are shown in Figure 6.8 where the coherence probabilities (i.e. the probabilities $\theta^{(l,c)}$ of Gilbert-Elliot channel $l$ remaining in the same quality state $c$) are varied but kept symmetric across channels and channel states. This symmetry ensures that the limiting distribution of channel states $\pi^{(\mathbf{l})}$ and the capacity $K$ remain constant across the whole evaluation range, with the only variation being how fast each channel switches states. Figure 6.8 shows that both delay and energy of FT are higher for higher coherence probabilities, because they lead to more packets being accumulated during bad state intervals, causing delay and energy to increase. The performance of MDP has a trend similar to FT with an expected offset that reflects the different performance requirements.

The performance of DM-VP provides interesting insights as it marks the boundary between different components of the DM performance. In particular, the performance gap between Q-MDP and DM-VP can be attributed to improved state estimation, whereas the performance gap between DM-VP and DM can be attributed to extra data delivered opportunistically when probing. Since the primary goal of the DM scheme is to improve performance through better estimation, it is desirable to keep former gap large both in terms of delay and energy, with DM-VP setting the reference for the achievable performance with better information. The challenge is to choose thresholds to achieve this. Here, we adopt $\eta^{(l)} = 0.25$ for both channels, as it has shown to provide a good compromise for settings around our base configuration.

Concerning schemes without complete information, Figure 6.8 shows that Q-MDP has a significantly higher delay for most of the considered range of coherence probabilities as a consequence of the higher uncertainty, only decreasing for high coherence probabilities. The delay performance is greatly improved by DM which keeps delay close to MDP across the whole evaluation range. The results for DM-VP support that most of the difference between the two heuristic schemes is due to better state estimation, and that the effect of data carrying probes looses relevance as the coherence probabilities increase. In terms of energy, DM pays a price that is higher, the lower the coherence probabilities, due to the higher probing frequency.

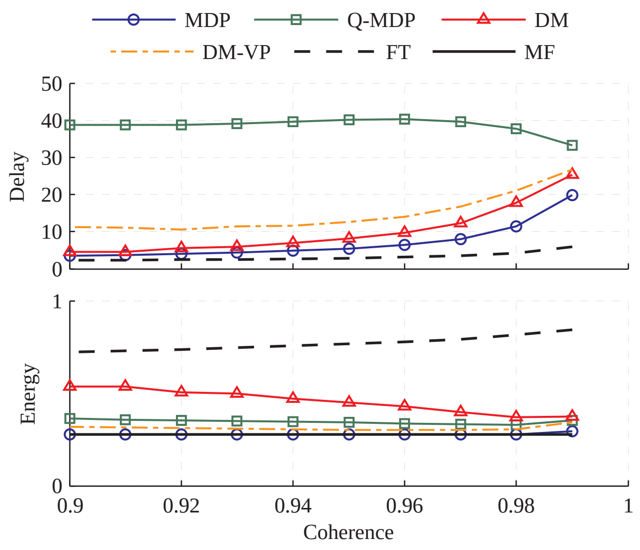


Figure 6.8: Impact of coherence time on the performance of the different schemes. The base configuration has been used with load set to 25% and symmetric coherence probabilities varying between 0.90 and 0.99. The design parameters were set as $\beta = 20$, $\alpha = 0.05$ and $\eta^{(1)} = \eta^{(2)} = 0.25$.

## 6.5.7 Imperfect Feedback

To cope with more realistic settings, the effects of imperfect feedback are now considered. Feedback quality is assessed in two dimensions, namely feedback loss rate and feedback delay.

### 6.5.7.1 Feedback Loss

The results for varying feedback loss rates are shown in Figure 6.9. Looking at energy, all heuristic schemes present an upward trend indicating more transmissions being made as the feedback loss rate goes up. Under uncertainty, the number of missing packets is overestimated, encouraging estimation based schemes (Q-MDP, DM and DM-VP) to send more packets. For the DM scheme this trend is much more accentuated due to the higher energy spent in probing. In terms of delay, Q-MDP exhibits decreasing delay until 80% feedback loss, after which point delay also increases. This can be explained by a decrease in transmission efficiency, due to poor channel state estimation, that outweighs the increased number of transmissions. On the other hand, DM delay always decreases albeit after 80% it is mostly due to excessive probing with data packets. In fact, the gaps to DM-VP show that information gathering looses relevance as feedback loss rate increases, as would be expected.

### 6.5.7.2 Feedback Delay

The results for varying feedback delay are shown in Figure 6.10. The performance trends are similar to the case with feedback losses. However, DM and Q-MDP follow the same delay trend, rather than

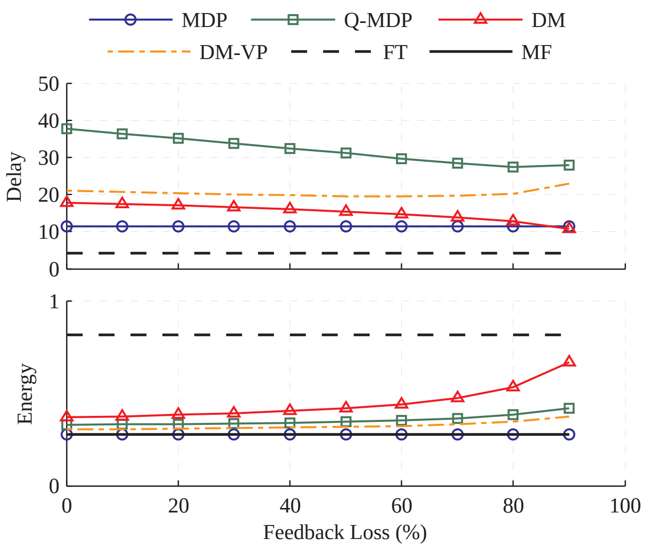


Figure 6.9: Impact of feedback losses on the performance of the different schemes. The base configuration has been used with load set to 25% and feedback loss rate $\xi^{(f)}$ varying between 0% and 90%. The design parameters were set as $\beta = 20$, $\alpha = 0.05$ and $\eta^{(1)} = \eta^{(2)} = 0.25$.

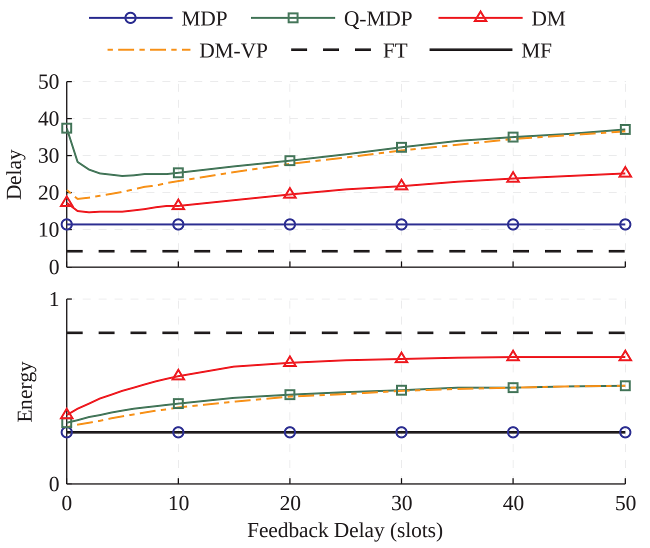


Figure 6.10: Impact of feedback delay on the performance of the different schemes. The base configuration has been used with load set to 25% and feedback loss rate $\delta$ varying between 0 and 50 slots. The design parameters were set as $\beta = 20$, $\alpha = 0.05$ and $\eta^{(1)} = \eta^{(2)} = 0.25$.

diverging. The explanation is that information gathering in the DM heuristic is triggered based on the filtered belief over which feedback delay has no influence. Hence, probing does not increase with feedback delay. Still, the delay curve of DM-VP approaches Q-MDP and steers away from DM as feedback delay increases, showing the reduced utility of information in the DM scheme.

Although both sources of imperfection have similar effects on performance, there is a higher sensitivity to delay in the sense that even a small delay can cause significant performance degradation. The rationale is that the information contained in feedback looses value during the time it takes to be delivered. In order for information to be useful, feedback delay must be significantly smaller than the coherence time of the channels.

#### 6.5.7.3 Packet Scheduling with Imperfect Feedback

Besides estimation, one major challenge under imperfect feedback assumptions is to decide which packets to transmit. This task is greatly simplified by the coding mechanism employed which ensures that novel packets are delivered in each transmission even with little or no feedback. This allows to reach high efficiency from an information theoretic perspective.

### 6.5.8 Use Cases

The presentation so far has steered around the base configuration to maintain the analysis simple while focusing on individual aspects of the system. Now, we consider potential use cases, where channels with different characteristics are combined to showcase the proposed methods in more representative scenarios.

#### 6.5.8.1 Two Slow Channels or One Fast Channel

This setup compares the performance of a two-channel system against a single-channel system. The two-channel system corresponds to the base configuration, whereas the single-channel has one channel with the same transition and error characteristics of the base configuration channels but with double transmission rate. In this way, both configurations possess the same network capacity. The single-channel system has an all-or-nothing nature with two extremes, providing either low or high throughput. The longer extreme bad state periods lead to larger queues, resulting in larger delays and less opportunities to spare transmissions in bad states, thus also increasing energy consumption. In contrast, the two-channel system provides a finer granularity with intermediate states where at least one of the channels is good, resulting in better performance in both metrics. To support this, consider the results in Figure 6.11. One result that stands out is the performance of FT, whose delay and energy are both lower for the two-channel case. In this case, delay gains over 2x can be attained for intermediate loads while still using less energy. When delay is traded for energy, the two-channel system also provides benefits as illustrated by the MDP and DM schemes for $\alpha = 0.05$. In this case, delay gains around 3x and 2x are achieved for low and intermediate loads, respectively. Overall, these results show the benefits of diversifying and relying on multiple slower channels rather than a single fast channel.

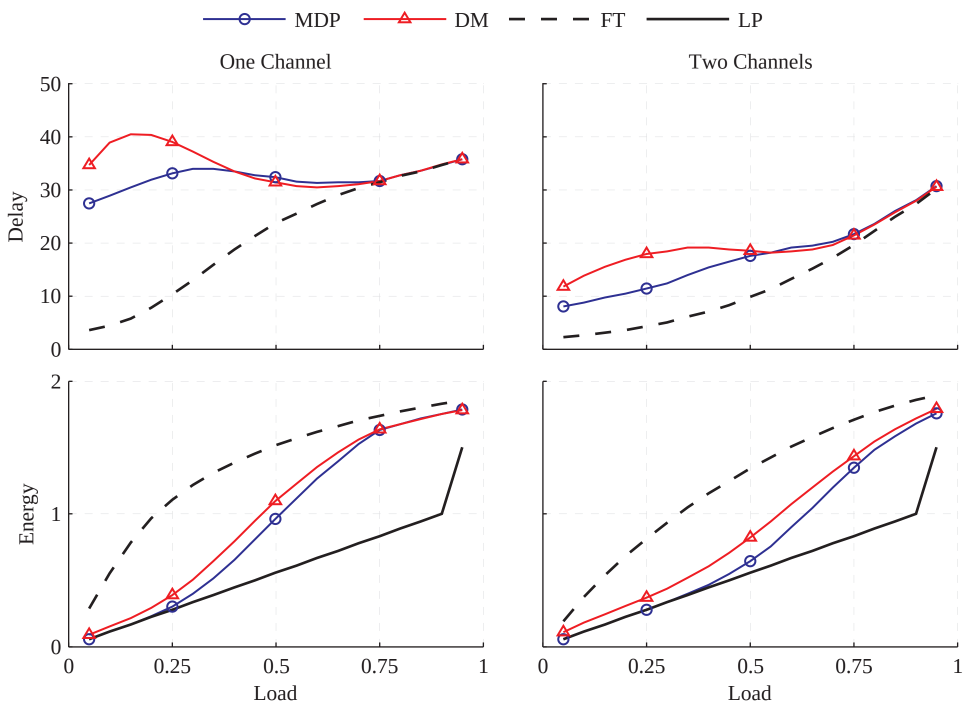


Figure 6.11: Performance comparison between a single channel system (left) and two channel system (right), both with capacity $K = 1$ packet per slot. The increased diversity provided by multiple channels enables better performance, which is seized by the proposed schemes. The design parameters were set as $\beta = 20$, $\alpha = 0.05$ and $\eta^{(1)} = \eta^{(2)} = 0.25$.

#### 6.5.8.2 Asymmetric Channels

This use case emulates two asymmetric links. The first link is characterized by coherence probabilities $\theta^{(1,b)} = 0.99$ and $\theta^{(1,g)} = 0.97$, error probabilities $\xi^{(1,b)} = 0.85$ and $\xi^{(1,g)} = 0.05$, transmission rate $N^{(1)} = 2$, and energy cost $J^{(1)} = 1$. The second link is characterized by coherence probabilities $\theta^{(2,b)} = 0.97$ and $\theta^{(2,g)} = 0.99$, error probabilities $\xi^{(2,b)} = 0.90$ and $\xi^{(2,g)} = 0.10$, transmission rate $N^{(2)} = 1$, and energy cost $J^{(2)} = 1$. Based on the transition probabilities, one can derive that the first channel spends 75% of the time in the bad state and 25% in the good state, and the second channel the other way around. To compensate, the first channel has twice the transmission rate and lower error probabilities, resulting in the channels having the same capacity. The results for the proposed setup are shown in Figure 6.12 for $\alpha = 0.10$. The solid line DM assumes perfect feedback whereas its dashed counterpart assumes $\xi^{(f)} = 0.50$ and $\delta = 5$. The performance gap between DM and MDP presents roughly the same magnitude across different loads, thus the relative overhead is higher for lower loads. For $\bar{\lambda} \leq 35\%$, the performance gap lies between 5%–50% (5%–20%) on delay and 30%–110% (55%–150%) on energy, with perfect (imperfect) feedback. For $35\% \leq \bar{\lambda} \leq 65\%$, the gap falls within 1%–10% (1%–10%) on delay and 7%–30% (11%–55%) on energy. For $\bar{\lambda} \geq 65\%$, the gap becomes even tighter, coming below 1% (10%) on delay and 7% (11%) on energy.

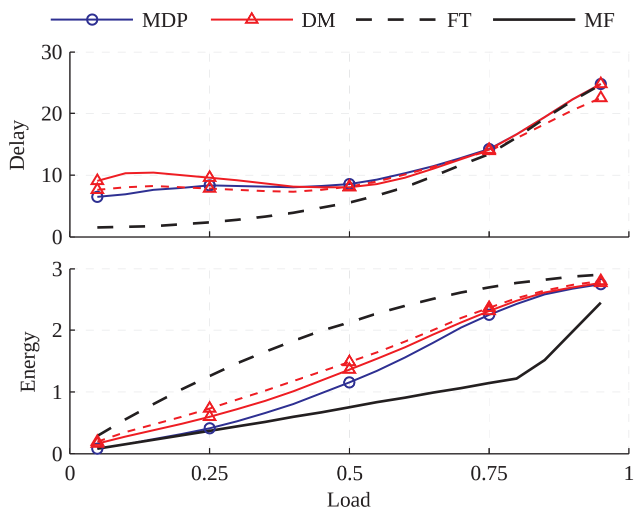


Figure 6.12: Operation with two channels, each with the same capacity but asymmetric coherence and error characteristics. The dashed DM curve corresponds to operation with an imperfect feedback channel characterized by $\xi^{(f)} = 0.50$ and $\delta = 5$. The design parameters were set as $\beta = 20$, $\alpha = 0.10$ and $\eta^{(1)} = \eta^{(2)} = 0.25$.

### 6.5.8.3 Heterogeneous Interfaces

This use case emulates a scenario where a high throughput link (e.g. WiFi) is backed up by a slower but more reliable link (e.g. 3G), capable of providing a steady QoS albeit with an added transmission cost. The first link is modeled as a Gilbert-Elliot channel with $N^{(1)} = 4$, $J^{(1)} = 1$, $\xi^{(1,b)} = 0.90$, $\xi^{(1,g)} = 0.10$ and $\theta^{(1,b)} = \theta^{(1,g)} = 0.98$. The second link is modeled as a packet erasure channel with $N^{(2)} = 1$, $J^{(2)} = 4$ and constant packet erasure rate $\xi^{(2)} = 0.10$. In summary, the first channel has a higher transmission rate and lower energy cost per packet but is more unreliable than the second, which has a low constant erasure rate. These parameters are usually determined from the characteristics of real channels and operation conditions, such as the moving speed, carrier frequency and expected SNR, as detailed in [50]. In this case, we simply defined values that are in line with other works in literature (see for instance [33], for a mapping of heterogeneous wireless interfaces to GE channel models).

The results for the proposed setup are shown in Figure 6.13 for $\alpha = 0.30$, where the dashed DM curve assumes $\xi^{(f)} = 0.50$ and $\delta = 5$. For $\bar{\lambda} \leq 35\%$, the delay performance of DM with perfect (imperfect) feedback is considerably degraded, spanning between 8%–470% (7%–700%) of the MDP case. Inspection of the MDP policy reveals that the slower/expensive link is used when the fast/cheap link is bad in order to keep delay low. In order to keep energy low however, DM requires a high level of confidence on the bad quality of the faster link in order to opt for the more expensive one, resulting in large delays. As the load increases, so does the importance of transmitting, hence DM uses the expensive channel more often and thus approaches MDP in terms of delay, while remaining close in terms of energy. For $35\% \leq \bar{\lambda} \leq 65\%$, DM with perfect (imperfect) feedback is within 1%–15% (1%–25%) on delay. For $\bar{\lambda} \geq 65\%$, the delay gap remains below 1% (15%). In terms of energy, the performance of DM with perfect (imperfect) feedback is roughly bound between 3%–20% (6%–30%) of the intended level, across the whole range of operation.

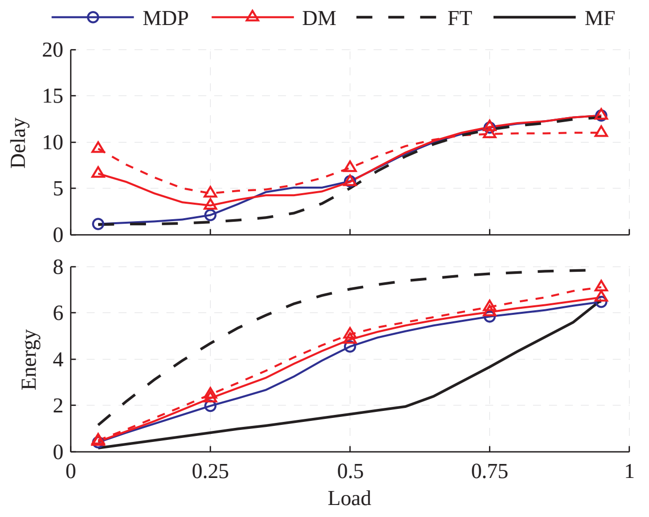


Figure 6.13: Operation with a high throughput link (e.g. WiFi), backed up by a slower but more reliable link (e.g. 3G) capable of providing a minimum level of service with an added transmission cost. The dashed DM curve corresponds to operation with an imperfect feedback channel characterized by $\xi^{(f)} = 0.50$ and $\delta = 5$. The design parameters were set as $\beta = 20$, $\alpha = 0.30$ and $\eta^{(1)} = \eta^{(2)} = 0.25$.

## 6.6 Conclusions

This chapter proposed mathematical models for designing and characterizing transmission policies over multiple, time-dependent wireless channels. We characterized the problem as an MDP for the case of perfect channel state knowledge and as a POMDP for the more realistic case of policies that require channel state estimation in the presence of delayed, lossy feedback. Our design merges opportunistic scheduling with online network coding to bring up schemes that can trade-off delay and energy according to the system's requirements.

Our numerical results show that the proposed design enables policies that provide different trade-offs in the performance space. Our analysis considers a wide range of parameters, including the effect of channel coherence, feedback delay, and feedback losses. It is shown that transmission over multiple, independent time-varying channels outperforms a single channel system with the same capacity in terms of delay. For a two channel system scheme, delay gains between two and three times were obtained while also using less energy. Moreover, the schemes proposed for partial observable scenarios are able to closely track the intended performance over different settings and operating regimes.

Future work shall consider scenarios with multiple transmitting nodes, both sources or relays, sharing the wireless channels. Note that our work's single transmitter case achieves energy conservation at the expense of underutilizing the wireless medium. However, the presence of multiple transmitting nodes would provide an opportunity to fill this void.

# Chapter 7

# Coded Multipath for Streaming

Throughout this dissertation, we have been advocating for the use of network coding to improve performance when communicating over multiple wireless links and, among other things, have shown its benefits at different scales of the delay metric. In the previous chapter, our analysis reached down to individual packet delays, considering the time taken since a packet enters the transmission buffer until it is *seen* and leaves the transmission buffer, which can be regarded as a queuing delay. Yet, there is one final piece missing which is the actual time until the original data is decoded and delivered to the intended application.

Network coding has been originally designed with bandwidth efficiency in mind. At the expense of this, data is delivered in an encoded manner and receivers must wait some time to be able to decode the information and access the original data. Naturally, this caveat raises some concerns on the application of network coding to delay sensitive applications, and real-time systems in particular, where data must be delivered within some deadline. Such as is the case of video streaming, an application that has been gaining popularity, either to access pre-stored video materials or live video feeds. The latter case is particularly interesting and challenging due to the need to keep a small delay between the stream reception and the actual live event. For instance, in remote monitoring scenarios, it is common to perform some kind of remote control based on a live video feed transmitted from the remote unit to a base station over a wireless medium. The viability of such applications naturally depends on the low latency of data transmission. But even in such unicast configurations, delivering packets in order can become a challenging task due to the random packet losses involved and scarce feedback information, which complicates packet error correction.

In this chapter, we address the barriers that exist between coding and real-time systems, closing the loop in terms of delay analysis of coded schemes over multiple wireless paths. To that end, we specifically consider ordered delivery of packets, a problem that underlies a wide range of important applications.

## Problem Statement

In this chapter, we seek transmission schemes that explore multiple communication channels and careful code constructions to minimize the *in-order delivery delay* of packets in limited feedback scenarios.

**In-Order Delivery Delay** (Figure 1.6): Time since a packet enters the transmission buffer until all packets preceding it and itself, are delivered to the application at the receiver.

## Contributions

- *Evaluation of Coded Multipath Schemes in Limited Feedback Scenarios:* We perform an empirical performance evaluation of coded multipath schemes in terms of the in-order delivery delay of packets. To that end, we propose a simple code construction that defines how to generate coded combinations in a controlled manner to achieve low decoding delays. Our results show that the considered schemes greatly outperform uncoded ones in limited feedback scenarios.
- *Worst-Case Delay Characterization:* Instead of the average delay metric considered in most works, we characterize the performance in terms of the probabilistic worst-case delay, aiming to better represent the stringent requirements found in real-time systems.
- *Multipath Architecture for Multicast Scenarios:* We propose a multicast architecture reliant on the use of multiple communication channels to each client to mitigate asymmetries among clients and improve delay performance while maintaining the use of bandwidth efficient codes.

## Publications

- A. Moreira, L. Almeida, and D. E. Lucani, ”Merging network coding with feedback management in multicast streaming,” *ACM SIGBED Rev.*, vol. 12, no. 3, pp. 49–52, June 2015

## 7.1 Related Work

The particular coding technique explored here is online network coding, a mode of operation that allows packets to be encoded as they arrive to the encoder. This cuts on the latency when compared with other codes that need to wait for a whole block of packets to start encoding, and thus makes it suitable for streaming applications. Moreover, feedback can be used to dynamically adjust which packets should be encoded to better satisfy specific performance requirements. In the seminal work on online network coding [21], the authors use feedback mainly to manage the buffer at the sender. In [51, 52, 28, 53], different authors study the decoding delay of online coding with feedback for the multiple receiver case. One common assumption in these works is the availability of perfect feedback, which is not feasible in wireless communications, and especially in multicast scenarios. A more practical solution to massive multicast streaming with online network coding has been proposed in [54]. However, the proposed coding mechanism only provides a best-effort solution with no formal analysis. Periodic feedback is considered in [55], in terms of the asymptotic behavior of throughput and decoding delay, and analyzed for a single client. The metric of choice considered is the in-order delivery delay, a suitable metric for streaming applications. Still, as most works, the analysis is based on average values. For more stringent requirements, such as those found in real-time applications, extreme values of the decoding delay should also be considered. In this chapter, we aim at characterizing the probabilistic worst-case delivery delay under periodic feedback settings.

## 7.2 System model

### 7.2.1 Network

We consider the setup depicted in Figure 7.1, where one source node is in charge of streaming a video to one receiver node through $L$ wireless interfaces. Each link $l$ between the source and receiver is modeled as a packet erasure channel with packet erasure probability $P_e^{(l)}$. Then, we consider that time is organized in slots. For simplicity, we assume that all $L$ interfaces work at the same rate, and each interface can transmit one packet per slot[1]. In total, the server can send $L$ packets per slot. Feedback can be periodically requested from the receiver only every $T$ slots. Hence, communication can be organized in *rounds* of $T$ slots, during which $M = T \cdot L$ packets are transmitted, and at the end of which feedback is received. Without loss of generality, we neglect the time taken by feedback requests and acknowledgments.

[1]The framework can be easily extended for different transmission rates

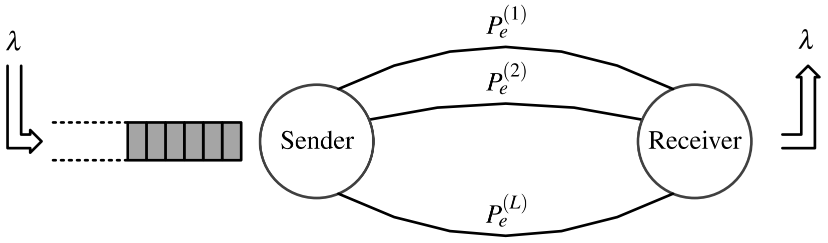


Figure 7.1: System Model for the considered unicast streaming problem, with one sender and one receiver connected by $L$ packet erasure channels.

### 7.2.2 Stream

We consider an application at the source node (sender) that produces packets at a rate of $\lambda$ packets/slot. In order to isolate our analysis from the effects of stochastic arrival processes, we assume that packets coming from the application pass through a traffic shaper that delivers the packets in a deterministic fashion to the transmission layer. At the destination node (receiver), an application consumes packets in an orderly fashion at an equal rate of $\lambda$ packets/slot. Hence, packets are delivered to the application in order. According to this, we define the *in-order delivery delay* as follows.

**Definition 10** (In-Order Delivery Delay). *The in-order delivery delay of a packet $p_k$ is defined as the time since it is available for transmission at the source until it and all previous packets $p_1$ to $p_k$ have been decoded and delivered to the application at the receiver.*

In order to achieve uninterrupted playback, the number of packets delivered to the receiver application must be greater than the number of packets consumed at all times. Due to the different types of delay inherent to data transmission and data decoding, meeting such condition requires buffering a certain number of packets before starting playback. The resulting playback delay dictates the maximum delay that each packet can incur without causing buffer underflow, and consequently playback interruption. To achieve uninterrupted playback, it is necessary to guarantee that the playback delay is higher than the individual delays of all packets. Naturally, the stochastic nature of the process does not allow to establish deterministic guarantees. Instead, we define the following probabilistic metric.

**Definition 11** (Probabilistic Worst-Case Delay (pWCD)). *The probabilistic worst-case delay is defined as the highest in-order delivery delay observed with some high probability $\varepsilon$.*

The above metric can be interpreted as the minimum playback delay necessary to ensure uninterrupted playback with some high probability $\varepsilon$, and will be the metric of choice for evaluating the performance in this chapter.

### 7.2.3 Coding

We assume that network coding is performed at the source in an online fashion. The online encoder takes packets as they become available in the buffer and generates coded packets that are linear combinations of these original packets, according to some code construction. The coded packets are then transmitted across the network. Upon reception, the receiver stores coded packets until decoding can occur, at which point the original packets are delivered to the application. In order to represent the knowledge of the receiver in terms of coded information we use the notion of *seen packet*, already introduced in Chapter 2. Here however, we consider a reverse definition that has been proposed in [22], and which holds a useful relationship with decoded packets.

**Definition 12** (Seen Packet)**.** *A receiver is said to have seen packet $p_k$ if it is able to compute a combination that only includes packet $p_k$ and packets $p_i$ with $i < k$, from all information it has received so far.*

As with the original definition, packets that have been seen by the receiver can be removed from the transmission buffer. On top of that, this particular definition of seen packet allows to easily establish a sufficient condition for a packet being decoded, which is presented in Lemma 5.

**Lemma 5** (Decodability)**.** *Under the terminology of Definition 12, if all packets $p_1, \dots, p_k$ up to some $k$ have been seen, then they have also been decoded.*

Although the above condition does not assert anything about isolated packet decodings, it does establish a condition for the decoding of all packets up to some packet $p_k$, which is the relevant event in the considered setup where packets are useful only when delivered in order.

## 7.3 Coding Scheme

Network coding schemes commonly combine all packets available, achieving optimal throughput at the cost of minimal decoding delay performance. In contrast, we explore schemes that combine packets in a more structured manner to reach decoding conditions sooner. Similarly to [55], we base our analysis on time-invariant schemes, i.e., schemes whose structure is the same for all communication rounds, which we extend to the multiple channel case as follows.

**Definition 13** (Time-Invariant Scheme)**.** *A time-invariant scheme can be described through a matrix*

$$x = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_{T-1}^{(1)} & x_T^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_1^{(L)} & x_2^{(L)} & \cdots & x_{T-1}^{(L)} & x_T^{(L)} \end{bmatrix} \tag{7.1}$$

*where $x_i^{(l)}$, for $1 \leq i \leq T$ and $1 \leq l \leq L$, are non-negative integers satisfying $x_i^{(l)} \leq M$. At the i-th slot of each round the source transmits a linear combination of the first $x_i^{(l)}$ unseen packets over interface l.*

These schemes incorporate the feedback obtained by considering the first $M$ unseen packets reported at the beginning of each round. As noted in [55], there is no advantage in combining more than $M$ unseen packets since at most $M$ innovative packets can be delivered, and therefore decoded, in each round. Apart from this, the individual elements of $x$ can be set in a number of different ways, dictating how many packets are combined in each transmission. Specifying these values composes the code design task. For the purpose of this chapter, which is to perform a preliminary and empirical analysis, we consider a simple yet flexible construction. Essentially, we consider a set of coding schemes characterized by a single parameter $\rho$, which specifies the rate at which packets are added for each transmission, such that the $k$-th packet in a round combines the first $U_k = \min\{\lceil \rho \cdot k \rceil, M\}$ unseen packets, i.e., $c_k = \sum_{i=0}^{U_k} p_i$. Below we provide some concrete examples for $T = 10$ slots and $L = 2$ channels.

**Example 1.**

$$x(\rho = 0.75) = \underbrace{\begin{bmatrix} 1 & 3 & 4 & 6 & 7 & 9 & 10 & 12 & 13 & 15 \\ 2 & 3 & 5 & 6 & 8 & 9 & 11 & 12 & 14 & 15 \end{bmatrix}}_{\textit{10 Slots}} \begin{array}{l} \textit{(Channel 1)} \\ \textit{(Channel 2)} \end{array}$$

**Example 2.**

$$x(\rho = 1.25) = \underbrace{\begin{bmatrix} 2 & 4 & 7 & 9 & 12 & 14 & 17 & 19 & 20 & 20 \\ 3 & 5 & 8 & 10 & 13 & 15 & 18 & 20 & 20 & 20 \end{bmatrix}}_{\textit{10 Slots}} \begin{array}{l} \textit{(Channel 1)} \\ \textit{(Channel 2)} \end{array}$$

Hence, by adjusting the parameter $\rho$ we can obtain codes that span different delay-throughput trade-offs. Lower values of $\rho$ result in more redundant combinations, lowering throughput but increasing the probability of earlier decoding events. Conversely, higher values of $\rho$ lead to higher throughput efficiency but larger decoding delay. In the limit, setting $\rho = M$ results in a greedy scheme that combines $M$ packets in every transmission.

We note that the scheme specifies the maximum number of packets that can be combined. The effective number of packets combined depends on the number of packets available at the transmission buffer. The effect of the coding scheme is expected to become more relevant as the load on the system increases and starts to approach capacity, leading to more packets accumulating in the transmission buffer, at which point the coding scheme takes effect to control the rate at which new information is sent.

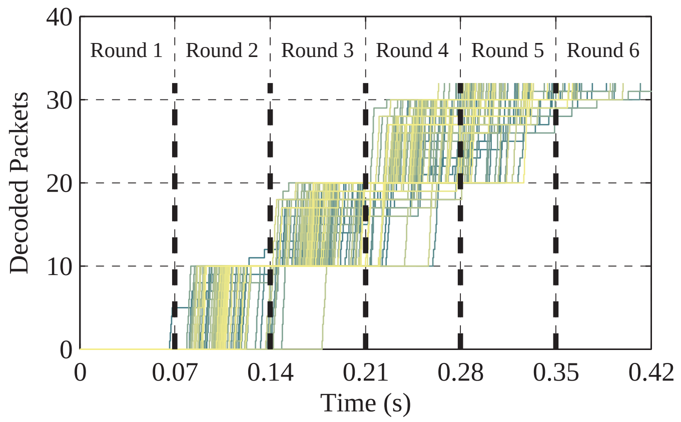


Figure 7.2: Experimental data showing the number of decoded packets over time and spanning multiple generations. The data corresponds to a single-path unicast setup composed by two Asus Eee laptops (server and client). In this experiment, a generation based approach was used with 32 packets per generation. The feedback period was set to $T = 10$ packets, corresponding to about 70 $ms$. Regarding the code, a greedy scheme was used in this example, hence all coded packets in a round consist of combinations of the first 10 unseen packets, which results in a clear burstiness of the decoding process.

## 7.4 Preliminary Experimental Results

In order to validate the proposed model and our understanding of the packet delivery process, we present some experimental results obtained during a brief experimental campaign. Our setup consisted of two Asus Eee laptops, each with a single 802.11g interface in a unicast setting. Rather than a live stream, data consisted on a video stored in the server. The configuration considered a period of $T = 10$ slots, and in this case also limited coding operations within generations of 32 packets. The number of decoded, and thus delivered, packets over time to the client application are shown in Figure 7.2, where the different generations have been overlapped for better visualization. The results illustrate quite well the organization of the communication in rounds and how small amounts of feedback help to unlock the coded data. The employed greedy scheme exhibits a bursty decoding behavior, reinforcing the idea that proper code designs should be used to achieve more regular delivery of data and consequently lower delivery delays, which is one of the motivations of this chapter. Nonetheless, this preliminary set of experimental results validates the considered model and confirms our understanding of the packet decoding and delivery process, motivating the rest of our analysis.

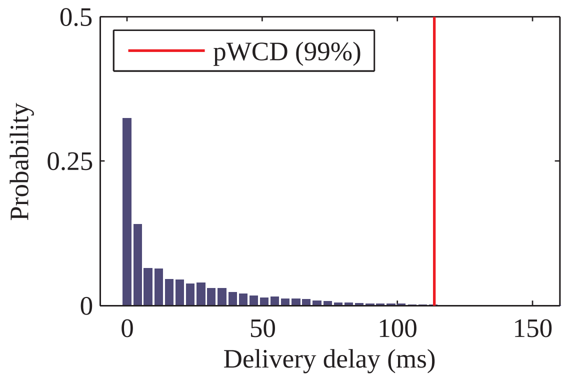
## 7.5 Numerical results

### 7.5.1 Coded vs. Uncoded

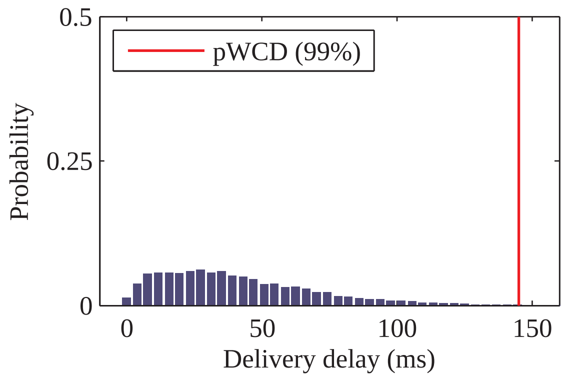
As we have shown in previous chapters, network coding is an efficient and flexible way to introduce redundancy that provides significant performance benefits in unicast scenarios with limited feedback information. Taking the results of Chapter 5 as an example, we saw that coded schemes greatly outperform uncoded ones in terms of the time to complete the transmission of a block of packets. The key explanation to this performance gap lies precisely in the *limited feedback* constraint which is inevitable in wireless systems. This limitation leads traditional uncoded transmission schemes into poor performance as there is no information about which packets have been lost, and therefore no clue on which packets to retransmit. In such a blind setting, one can either (i) transmit each packet multiple times, which is a rigid and inefficient way to add redundancy, or (ii) repeat just a few specific packets, which is mostly ineffective. Because of this, coded schemes are actually able to outperform uncoded ones in terms of in-order delivery of packets.

To see this we compare the performance of coded and uncoded schemes, in a setting with $L = 2$ interfaces, each running at 2 *Mbps* with 20% packet erasure rate, and feedback period of $T = 10$ slots. For the uncoded scheme, the first $M$ undelivered packets at the beginning of the round are each sent once uncoded, which corresponds to sending a different packet in each transmission. For the coded scheme, although we could fine tune $\rho$ to adapt to the arrival rate, we use $\rho = 1$ to make a fair comparison with the proposed uncoded scheme, since it corresponds to including one new packet in the code at each transmission.

The in-order delivery delay distributions for the coded and uncoded schemes are shown in Figure 7.3 for the considered network configuration and $\bar{\lambda} = 95\%$ arrival load. Clearly, the delay distribution for the coded scheme is more concentrated around small values, whereas that of the uncoded scheme is much more spread out. Accordingly, the pWCD of the uncoded scheme is considerably higher than that of the coded scheme, even for such a high load configuration. Furthermore, these results provide a detailed look on the delay distribution, illustrating the meaning of the pWCD, for $\varepsilon = 0.99$, which we will exclusively consider from hereon to characterize the performance.



(a) Coded ($\rho = 1$)



(b) Uncoded

Figure 7.3: Packet delivery delay distributions achieved with the coded and uncoded schemes for 95% arrival load. The results are for $L = 2$ network interfaces, each running at 2 *Mbps* and packet erasure rates of $P_e^{(1)} = 20\%$ and $P_e^{(2)} = 20\%$.

Now, we would like to analyze the performance of coded and uncoded approaches over different packet arrival rates, as well as analyzing the impact of $\rho$ in the performance of the proposed coded schemes. For that we define three instances of the coded scheme:

- *Unitary ($\rho = 1$):* This variant corresponds to adding one new packet to the code in each transmission, as previously considered.
- *Greedy ($\rho = M$):* This variant corresponds to a greedy scheme that combines all the first $M$ unseen packets in each transmission of a round.
- *Proportional ($\rho \propto \lambda$):* This variant adjusts the code rate $\rho$ to the arrival rate $\lambda$. Specifically, we set $\rho = c\lambda/L$, where the factor $c > 1$ serves to ensure that $\rho L > \lambda$, i.e., the number of packets to added to the code per slot is higher than the arrival rate, as required from a stability point of view. From a delay perspective, $c$ should be as close to 1 as possible, without causing instability. Based on empirical evidence obtained from our tests, we set $c = 1.1$.

Keeping the same channel configurations, we present in Figure 7.4 the results for the uncoded scheme, as well as all three instances of the coded scheme, for arrival loads ranging from 50% to 95% of the network capacity. Starting by the comparison among coded schemes, for low arrival rates, all coded schemes perform similarly as the combinations that each of them can perform is equally limited by the small number of packets typically present in the transmission buffer. As the arrival rate increases, the average number of packets in the transmission buffer increases with it, and the divergence in performance of the different coded schemes starts to show. As expected, the *greedy* scheme provides a very bad response as the arrival rate increases and approaches the network capacity, reinforcing that idea that such constructions are not suited for low-delay. On the other hand, even the simplest *unitary* scheme provides a much better performance, being only slightly improved by the *proportional* scheme.

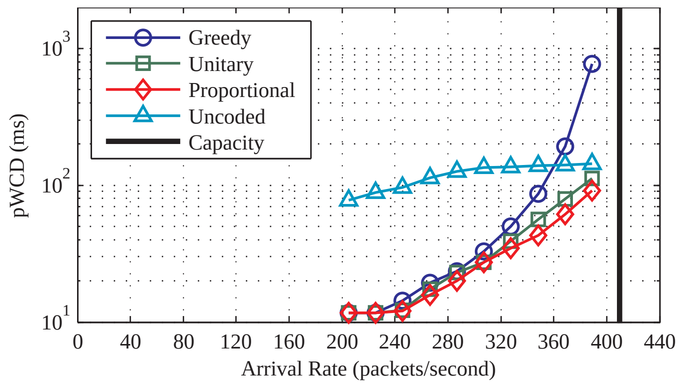


Figure 7.4: Performance of coded and uncoded schemes for different arrival rates. The results are for $L = 2$ interfaces, each running at 1 $Mbps$ and packet erasure rates of $P_e^{(1)} = 20\%$ and $P_e^{(2)} = 20\%$.

Switching to the comparison of coded and uncoded schemes, we can observe the superior performance of the coded schemes across the whole range of operation considered, with the exception of the greedy scheme which starts to perform worse for higher arrival rates. For a 50% load ($\lambda = 200pps$), the pWCD of the uncoded scheme is about 5 times that of the coded ones. As the arrival rate increases, the gain diminishes as there is less spare bandwidth to send redundant coded packets that would potentiate early decoding events. Nonetheless, the unitary and proportional schemes provide considerable gains even for a 95% load.

These results show that coded schemes with a simple design can greatly outperform uncoded schemes in terms of the in-order delivery delay, in limited feedback settings. Moreover, with the coded approach it is easy to adjust the throughput-delay trade-off (through $\rho$ in the case of our scheme) in way that is not possible with uncoded transmissions.

### 7.5.2 Single-Path vs. Multipath

Coding techniques have been shown to be able to deliver information with high throughput efficiency with little or no information whatsoever. But when packets need to be delivered in a timely manner, having better information can help deliver certain combinations faster and thus improve delay performance.

Intuitively, working with lower error rates increases the likelihood of delivering continuous sequences of packets without errors. To evaluate the extent to which this effect can affect performance, we compare our coded scheme in two different network configurations, one with a single link running at 6 $Mbps$ with 40% packet losses, and another one with two links each running at 2 $Mbps$ with 10% packet losses. It should be highly emphasized that both configurations provide the same capacity of 3.6 $Mbps$, and therefore any differences in performance are not due to differences in the effective throughput. The performance achieved by the coded scheme with $\rho = 1$ is shown in Figure 7.5 for both configurations, where a feedback period of $M = 20$ packet transmissions. The results reveal a significant performance gap between the two configurations that increases with the load, reaching up to 3 times in favor of the setting with lower packet loss rates. Indeed, it should highlighted that this difference is not necessarily due to the existence of multiple channels. With a single link it is also possible to articulate transmission rate and packet loss rate. However, depending on the operation point, it may be difficult to do so without reducing the total throughput. For instance, going from 40% to 10% error rate (a 4 times reduction), increases packet delivery rate only by 1.5 times. To maintain the same throughput, the transmission rate cannot be below 1.5 times the original one, which may be difficult to achieve, especially given the limited set of transmission rates to which network cards are subject to. In this context, multiple interfaces do offer the flexibility necessary to scale throughput while supporting low error rates, which have been shown here to be an important factor for good performance in limited feedback scenarios where packets need to be delivered in order.

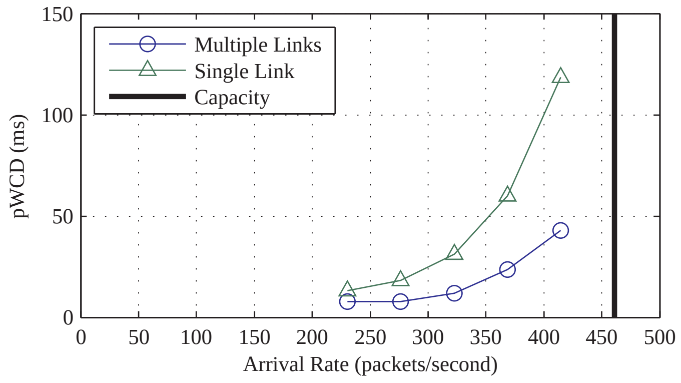


Figure 7.5: Performance of the same coded scheme in different network configurations with equal capacity. The single link configuration considers an interface running at 6 *Mbps* and $P_e = 40\%$. The multiple link configuration considers two interfaces, each running at 2 *Mbps* and with packet erasure rates of $P_e^{(1)} = 10\%$ and $P_e^{(2)} = 10\%$. In both cases, these settings result in a total throughput capacity of 3.6 *Mbps*.

## 7.6 The Multicast Case

Finally, we step out of our usual unicast setting and take a brief look on a multicast configuration to address another relevant application in the context of video streaming which is live broadcast. A particular example of this case is when many wireless clients try to access the same video stream in the same hot spot, e.g., associated to a sports or cultural event. This case, for bandwidth efficiency and scalability reasons, requires multicast transmissions over a wireless medium which typically uses unacknowledged packets at the link layer, and are thus unreliable. In heavy loaded conditions, packet losses increase and the overall quality of the stream reception degrades. Achieving a good broadcast streaming service requires solving challenging problems that arise from the strong asymmetries among clients. These asymmetries exist in two important ways. First, the links to each client present different error rates, which makes some clients lose more packets than others. Secondly, and even if the error rates were similar, clients lose different packets all of which need to be recovered.

Our goal is to perform a preliminary investigation on whether multiple uncorrelated channels can benefit performance of transmission to multiple clients. The intuition is that keeping connections over multiple channels to each client, where the quality of the channels are uncorrelated from each other, helps to average out the quality experienced by the different receivers, and consequently improve the joint performance, where we consider a packet to be delivered only when it is delivered to *all* clients.

### 7.6.1 Multicast Model

We consider the setup in Figure 7.6 which is essentially an extension of the unicast model for multiple receivers. Here a source node is in charge of multicasting a video to a group of $N$ receivers through

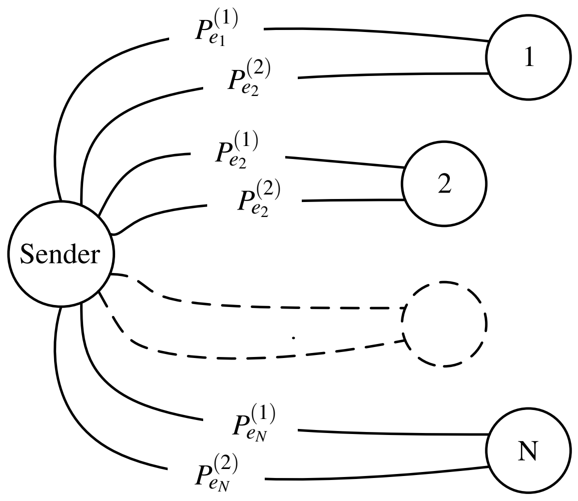


Figure 7.6: Multicast streaming system model

$L$ wireless interfaces, where the link between the source and each receiver $i$ over interface $l$ is each modeled as a packet erasure channel with packet erasure probability $P_{e_i}^{(l)}$. Similarly to the unicast case, we consider that feedback is requested periodically every $T$ slots. In this preliminary analysis, we assume that feedback is requested to all clients, albeit with feedback period much larger than the ones considered in the unicast case.

### 7.6.2 Numerical Results

To evaluate our ideas, we setup a simulation with $N = 100$ clients in two different network configurations. In one configuration we considered a single interface, i.e., one link to each client, running at 4 $Mbps$ with the losses to each client uniformly distributed between 10% and 40%. In the second configuration we considered two interfaces, i.e., two links to each client, each running at 2 $Mbps$, where the losses over each channel are independent and uniformly distributed between 10% and 40% across clients. Although, the two systems could seem equivalent in terms of average throughput, the division of capacity in two uncorrelated channels in the second configuration considerably changes channel quality distribution across clients. To see the real impact of this effect on performance, we show the delay performance results in Figure 7.7, where the feedback period was set to $M = 200$ packet transmissions and a greedy coding scheme was used for bandwidth efficiency. Interestingly, the simple fact of having multiple interfaces leads to gains above 1.5 times for high loads in the considered scenario.

Network coding is known to be an extremely bandwidth efficient technique in wireless multicast scenarios. However, trading throughput for delay in these settings is also known to be a much more complex problem. Hence, we believe that our proposed architecture with multiple interfaces makes up for a nice match with coding techniques, enabling better performance while keeping the bandwidth efficient codes.

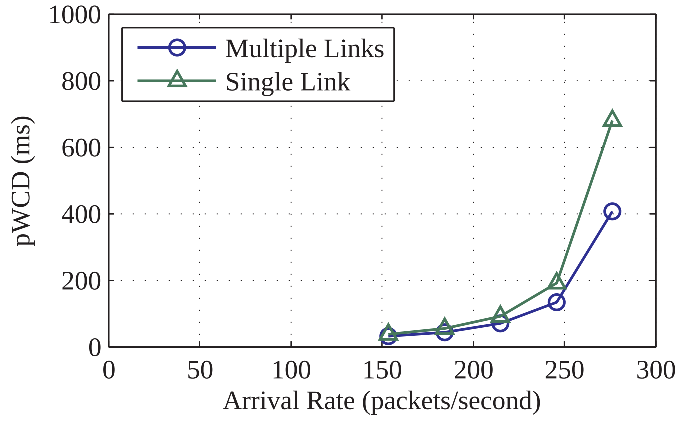


Figure 7.7: Performance of the coded scheme in different network configurations for a multicast scenario with $N = 100$ clients. The single link configuration considers an interface running at 4 $Mbps$ and $P_{e_i}^{(1)} \sim \text{Uniform}(10\%, 40\%)$. The multiple link configuration considers two interfaces, each running at 2 $Mbps$ and with packet erasure rates of $P_{e_i}^{(1)} \sim \text{Uniform}(10\%, 40\%)$ and $P_{e_i}^{(2)} \sim \text{Uniform}(10\%, 40\%)$. Each data point corresponds to the the average of 100 different runs, where the packet error probabilities to each client are randomly generated at the start of each run.

## 7.7 Conclusions

In this chapter, we considered coded multipath transmission techniques to reduce the in-order delivery delay in unicast and multicast settings with limited feedback. We have shown that coded schemes far surpass uncoded ones even in unicast settings, due to the ability of redundant coded packets to patch the holes created by packet erasures in an ordered sequence of packets. Also, we showed that in the absence of feedback, low rate interfaces provide a more predictable way to transmit information benefiting the ordered delivery of packets. In multicast scenarios, network coding has been well known to be an efficient and effective technique for wireless dissemination of information. However, designing low-delay codes in these settings without sacrificing much on throughput is a complex problem, in part due to the heterogeneous quality experienced by different clients. In this chapter, we suggested an architecture reliant on multiple communication channels that seeks to mitigate this problem by smoothing the quality differences experienced by clients, greatly improving delay performance even when using highly throughput efficient codes.

Despite the lack of a more strong formal analysis, this chapter provided empirical knowledge that strengthens our claims and opens several paths for future research. One interesting line for future research is to explore dynamic schemes. For example, the round size and the code structure could both be chosen dynamically depending on the erasure rate and packets missing by the clients at the beginning of each round. This would enable a more reactive behavior, quickly switching between high throughput and low delay settings as necessary, depending on the state of the decoding process. Naturally, it would also be relevant to consider more realistic settings. This includes considering larger feedback periods, incorporating the time needed to request and receive feedback, and performing adaptation under varying network conditions. Ultimately, it would be interesting to perform real-world experiments to validate these ideas.

# Chapter 8

# Conclusions

The number of coexisting wireless technologies has never been greater, enabling the transmission of data over multiple radio interfaces. But the realization of this promising idea lies dependent on the development of schemes that can efficiently harness the multitude of available resources. This work argued that coding techniques are the key to such schemes and investigated the benefits of coded communications over multiple wireless interfaces. Throughout a series of four chapters, we made several contributions and drew several conclusions that support our claims. Here we wrap the main ideas behind our thesis and its validation.

## 8.1 Thesis Validation

**Claim 1** Our first claim stated that data transmission over multiple wireless interfaces brings significant benefits to communication performance.

To show the validity of this claim, we identified two major benefits of using multiple interfaces that influence delay performance differently. The most upfront benefit comes from *bandwidth aggregation*. Aggregating the bandwidth of multiple channels inherently increases the overall capacity and consequently the maximum achievable transmission rate. This effect translates into higher average throughput and in turn shorter long-term transmission times. This aspect has been leveraged in Chapters 4 and 5, where block transmission delays were analyzed. In Chapter 4 it has been demonstrated that the throughput of different interfaces can effectively be aggregated in practice, reducing the time needed to transmit large files. Then, Chapter 5 has shown that coded schemes that adequately split packet transmissions among multiple interfaces are able to achieve transmission times close to the minimum. Naturally, this kind of benefit stems from using more radio frequency spectrum resources but it is still a relevant one as it provides a flexible way to adapt to different bandwidth requirements, without need for changes on the base technologies.

The second group of benefits stems from a more subtle, yet powerful aspect which is *diversity*. Signal quality diversity in particular, offers large benefits to certain applications in terms of delay. The main intuition is that having multiple channels with different, or simply uncorrelated, signal characteristics increases the chances of having at least one channel in a good state available, increasing the responsiveness of the system. In scenarios where packets arrive as a stream, this property is extremely relevant. This has been verified in Chapter 6, where individual packet queuing delays were analyzed in the context of stream transmissions. A remarkable result comes from comparing the performance of single-channel and two-channel systems with the same aggregate capacity. For a significant range of arrival loads, delay in the two-channel system is less than half of the single-channel system, while also requiring less transmissions and therefore less energy. In Chapter 6 we provided a quick insight on how multiple uncorrelated channels can be used to help to smooth the quality experienced by different clients in a multicast scenario, bringing benefits in terms of the global delay performance.

In summary, we can draw the following conclusions regarding the effect of these two aspects on the packet delay distribution. When dealing with large amounts of data, increased bandwidth translates into lower average delay whereas increased diversity translates into tighter concentration around the mean. When data arrives as a stream, bandwidth is important to support the incoming flow of packets, but from thereon diversity plays a much more important role as it is not as important to have more resources as it is to have the sufficient ones at the right time. In such cases, diversity helps by reducing not only the variance but the average delay itself.

**Claim 2** Our second claim stated that network coding is critical to deliver flexible, practical and efficient transmission mechanisms that exploit multiple communication interfaces.

To validate this claim we focus on two main benefits achieved with network coding in the context of our work, namely (i) throughput *efficiency* under limited feedback conditions allied with (ii) *simplified* scheduling across interfaces with asymmetric characteristics.

In Chapter 4, the error correcting capabilities of network coding have been demonstrated in practice. Our coded protocol operating over UDP has been shown to outperform uncoded transmissions over TCP in the presence of packet losses, taking most advantage of the provided bandwidth. In Chapter 5, the ability of coded schemes to complete the transmission of a block of packets in a single shot and with little overhead is fundamental to achieve near-optimal block transmission times. In Chapter 6, coding enables good delay performance in the presence of incomplete channel information, as well as delayed and lossy feedback. In Chapter 7, coding is a fundamental tool in improving the in-order delivery delay of packets under limited feedback conditions, as well as to cope with heterogeneous packet losses in multicast streaming, thus enabling a scalable solution.

But what is also remarkable is that all this efficiency is achieved with simple scheduling mechanisms across interfaces with different characteristics. In Chapter 4, the problem of scheduling packets was reduced to one of scheduling generations. In Chapter 5, it became clear that scheduling coded packets can be as simple as deciding how many packets to send in each interface since coded packets are, under the considered assumptions, equally informative. This contrasts with traditional transmissions, where to recover from errors it is necessary to define where and how many times to replicate each packet, with the added disadvantage that it is not as throughput efficient. In Chapter 6, similar ideas allowed us to establish a feasible decision making problem. Specifically, the set of possible actions in the considered decision theoretic framework is minimal, corresponding to the number of packets to transmit in each interface in each slot. In Chapter 7 it was shown that even a simple, yet careful, code construction is able to significantly reduce the delivery delay of individual packets when compared to uncoded schemes.

Overall, we conclude that network coding has been instrumental in our approach to multipath transmissions, providing an easy, efficient and flexible way to split and balance traffic among multiple interfaces.

## 8.2 Future Work

**Low-Level Multipath Protocol Implementation** When developing the multipath file transfer application presented in Chapter 4 we encountered a series of limitations, some of which resulted from confining implementation to the application layer. Moving implementation down the protocol stack would certainly provide more flexibility and enable a low-level management that would expose a single virtual link to upper layers.

**Experimental Evaluation** It would be quite valuable to implement and evaluate the proposed mechanisms in an experimental setup, especially in the context of a particular application. For the work in Chapter 5, we could use the developed Android application to perform some more experiments dedicated to the transmission of a single block of packets. Regarding the work in Chapter 7, we also developed a testbed for multicast video streaming, albeit few tests have been possible by the time of writing, and therefore it would be interesting to carry out more tests.

**Integration in Wider Settings** Our work investigated transmission across multiple paths over a single hop. After assessing its benefits in this elementary setting, it would be interesting to integrate the proposed ideas in other configurations. In Chapter 7 we already expanded our framework on a promising direction which is a multicast configuration. Other possible directions include considering multiple data sources, where a receiver can obtain data from multiple uncoordinated sources via different interfaces. The latter would be interesting for modeling recent edge caching systems based on coded distributed storage, another key technique being considered for 5G systems.

**Explore mmWave Communications** Recently, millimeter-wave (mmWave) communications have drawn a lot of attention as they are expected to play a major role in future networks. Set to operate in the 60 GHz unlicensed band, it presents unique characteristics. On the one hand, it has the potential to enable extremely high data rates given the right conditions. On the other hand, it is also highly susceptible to attenuation effects, offering poor performance under non-line-of-sight conditions. Therefore, even such a disruptive technology is not an universal solution but rather a complement to the existing ones, with very unique characteristics, and it would be interesting to investigate how they can work together.

# Bibliography

[1] J. G. Andrews, "Seven ways that HetNets are a cellular paradigm shift," *IEEE Communications Magazine*, vol. 51, no. 3, pp. 136–144, Mar. 2013.

[2] A. Goldsmith, *Wireless Communications*. Cambridge University Press, 2005.

[3] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[4] S.-Y. Li and R. Yeung, "Linear Network Coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, Feb. 2003.

[5] R. Koetter and M. Médard, "An Algebraic Approach to Network Coding," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 782–795, Oct. 2003.

[6] S. Jaggi, P. A. Chou, and K. Jain, "Low complexity algebraic multicast network codes," in *Proc. IEEE Int. Symp. Inf. Theory*. Yokohama, Japan: IEEE, June 2003, pp. 368–368.

[7] P. Sanders, S. Egner, and L. Tolhuizen, "Polynomial Time Algorithms for Network Information Flow," in *Proc. 15th ACM Symp. Parallel Algorithms Archit.* San Diego, CA, USA: ACM, June 2003, pp. 286–294.

[8] T. Ho, M. Médard, J. Shi, M. Effros, and D. R. Karger, "On Randomized Network Coding," in *Proc. Allert. Conf. Commun. Control. Comput.*, Monticello, IL, USA, Oct. 2003.

[9] P. A. Chou, Y. Wu, P. Chou, and K. Jain, "Practical Network Coding," in *Proc. Allert. Conf. Commun. Control. Comput.*, Monticello, IL, USA, Oct. 2003.

[10] S. Jaggi, P. Sanders, P. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial Time Algorithms for Multicast Network Code Construction," *IEEE Trans. Inf. Theory*, vol. 51, no. 6, pp. 1973–1982, June 2005.

[11] T. Ho, M. Médard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A Random Linear Network Coding Approach to Multicast," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[12] D. E. Lucani, M. Médard, and M. Stojanovic, "Random Linear Network Coding for Time-Division Duplexing: Field Size Considerations," in *2009 IEEE Glob. Telecommun. Conf. (GLOBECOM)*. Honolulu, HI, USA: IEEE, Nov. 2009, pp. 1–6.

[13] D. E. Lucani, M. V. Pedersen, J. Heide, and F. H. P. Fitzek, "Fulcrum Network Codes: A Code for Fluid Allocation of Complexity," *CoRR*, vol. abs/1404.6620, 2014.

[14] M. Luby, "LT codes," in *Proc. IEEE Symposium on Foundations of Computer Science*, Nov. 2002, pp. 271–280.

[15] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[16] S. Feizi, D. E. Lucani, C. W. Sørensen, A. Makhdoumi, and M. Médard, "Tunable sparse network coding for multicast networks," in *International Symposium on Network Coding (NetCod)*. Aalborg, Denmark: IEEE, June 2014, pp. 1–6.

[17] P. Maymounkov, N. J. Harvey, and D. S. Lun, "Methods for efficient network coding," in *Proc. 44th Annual Allerton Conference on Communication, Control, and Computing*, 2006, pp. 482–491.

[18] D. Silva, W. Zeng, and F. R. Kschischang, "Sparse network coding with overlapping classes," in *Workshop on Network Coding, Theory, and Applications*. Lausanne, Switzerland: IEEE, June 2009, pp. 74–79.

[19] A. Heidarzadeh and A. H. Banihashemi, "Overlapped Chunked network coding," in *IEEE Information Theory Workshop on Information Theory (ITW 2010, Cairo)*, Cairo, Egypt, Jan. 2010, pp. 1–5.

[20] Y. Li, E. Soljanin, and P. Spasojevic, "Effects of the Generation Size and Overlap on Throughput and Complexity in Randomized Linear Network Coding," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 1111–1123, Feb. 2011.

[21] J. K. Sundararajan, D. Shah, and M. Medard, "ARQ for network coding," in *IEEE International Symposium on Information Theory*, Toronto, Canada, July 2008, pp. 1651–1655.

[22] J. K. Sundararajan, P. Sadeghi, and M. Médard, "A feedback-based adaptive broadcast coding scheme for reducing in-order delivery delay," in *Proc. Workshop on Network Coding, Theory, and Applications*. IEEE, June 2009, pp. 1–6.

[23] A. L. Ramaboli, O. E. Falowo, and A. H. Chan, "Bandwidth aggregation in heterogeneous wireless networks: A survey of current approaches and issues," *J. Netw. Comput. Appl.*, vol. 35, no. 6, pp. 1674–1690, 2012.

[24] K. Habak, K. A. Harras, and M. Youssef, "Bandwidth aggregation techniques in heterogeneous multi-homed devices: A survey," *Computer Networks*, vol. 92, pp. 168–188, 2015.

[25] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths," *IEEE/ACM Transactions on Networking*, vol. 14, no. 5, pp. 951–964, 2006.

[26] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses," Tech. Rep., Jan. 2013, RFC 6824.

[27] S. C. Nguyen and T. M. T. Nguyen, “Evaluation of multipath TCP load sharing with coupled congestion control option in heterogeneous networks,” in *Global Information Infrastructure Symposium (GIIS)*. Da Nang, Vietnam: IEEE, Aug. 2011, pp. 1–5.

[28] J. K. Sundararajan, D. Shah, M. Médard, M. Mitzenmacher, and J. Barros, “Network Coding Meets TCP,” in *IEEE INFOCOM*, Rio de Janeiro, Brazil, Apr. 2009, pp. 280–288.

[29] M. Li, A. Lukyanenko, and Y. Cui, “Network coding based multipath TCP,” in *IEEE INFOCOM Workshops*, Orlando, FL, USA, Mar. 2012, pp. 25–30.

[30] M. Kim, A. ParandehGheibi, L. Urbina, and M. Médard, “CTCP: Coded TCP using Multiple Paths,” *CoRR*, vol. abs/1212.1929, 2012.

[31] J. Cloud, F. du Pin Calmon, W. Zeng, G. Pau, L. M. Zeger, and M. Médard, “Multi-path TCP with network coding for mobile devices in heterogeneous networks,” in *IEEE Vehicular Technology Conference (VTC Fall)*, Las Vegas, NV, USA, Sept. 2013, pp. 1–5.

[32] N. Capela and S. Sargento, “Multihoming and network coding: A new approach to optimize the network performance,” *Computer Networks*, vol. 75, pp. 18–36, 2014.

[33] C. Pereira, A. Aguiar, and D. E. Lucani, “Dynamic Load Allocation for Multi-Homing via Coded Packets,” in *Proc. IEEE Veh. Technol. Conf.* Dresden, Germany: IEEE, June 2013, pp. 1–5.

[34] M. V. Pedersen, D. E. Lucani, F. H. P. Fitzek, C. W. Sorensen, and A. S. Badr, “Network coding designs suited for the real world: What works, what doesn’t, what’s promising,” in *Proc. IEEE Inform. Theory Workshop*, Sevilla, Spain, Sept. 2013, pp. 1–5.

[35] A. Paramanathan, M. V. Pedersen, D. E. Lucani, F. H. P. Fitzek, and M. Katz, “Lean and mean: network coding for commercial devices,” *IEEE Wireless Commun.*, vol. 20, no. 5, pp. 54–61, Oct. 2013.

[36] J. Gettys, “Bufferbloat: Dark Buffers in the Internet,” *IEEE Internet Comput.*, vol. 15, no. 3, pp. 96–96, May 2011.

[37] M. V. Pedersen, J. Heide, and F. H. P. Fitzek, “Kodo: An Open and Research Oriented Network Coding Library,” in *NETWORKING 2011 Workshops*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, vol. 6827, pp. 145–152.

[38] M. Watson, M. Luby, and L. Vicisano, “Forward Error Correction (FEC) Building Block,” Tech. Rep., Aug. 2007, RFC 5052.

[39] D. P. Bertsekas and J. N. Tsitsiklis, *Introduction to Probability*, 2nd ed. Athena Scientific, 2008.

[40] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Wiley-Interscience, 2006.

[41] A. Moreira and D. E. Lucani, “On coding for asymmetric wireless interfaces,” in *Proc. IEEE Int. Symp. Network Coding*, Cambridge, MA, June 2012, pp. 149–154.

[42] W. Hoeffding, “Probability Inequalities for Sums of Bounded Random Variables,” *J. Amer. Statist. Assoc.*, vol. 58, no. 301, pp. 13–30, Mar. 1963.

[43] D. E. Lucani, M. Médard, and M. Stojanovic, "On Coding for Delay - Network Coding for Time-Division Duplexing," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2330–2348, Apr. 2012.

[44] F. Gomes, "Load balancing in multi-beam satellites," Porto, Portugal, 2012.

[45] P. Sadeghi, R. Kennedy, P. Rapajic, and R. Shams, "Finite-State Markov Modeling of Fading Channels - A Survey of Principles and Applications," *IEEE Signal Process. Mag.*, vol. 25, no. 5, pp. 57–80, Sept. 2008.

[46] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Athena Scientific, 2005.

[47] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Pearson, 2009.

[48] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, May 1998.

[49] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of Markov decision processes," *Mathematics of Operations Research*, vol. 12, no. 3, pp. 441–450, 1987.

[50] H. S. Wang and N. Moayeri, "Finite-state Markov channel-a useful model for radio communication channels," *IEEE Transactions on Vehicular Technology*, vol. 44, no. 1, pp. 163–171, 1995.

[51] L. Keller, E. Drinea, and C. Fragouli, "Online Broadcasting with Network Coding," in *Proc. Workshop on Network Coding, Theory and Applications*. IEEE, Jan. 2008, pp. 1–6.

[52] J. Barros, R. A. Costa, D. Munaretto, and J. Widmer, "Effective delay control in online network coding," in *Proc. IEEE INFOCOM*. IEEE, Apr. 2009, pp. 208–216.

[53] A. Fu, P. Sadeghi, and M. Médard, "Dynamic rate adaptation for improved throughput and delay in wireless network coded broadcast," *IEEE/ACM Transactions on Networking*, vol. 22, no. 6, pp. 1715–1728, 2014.

[54] D. Ferreira, R. A. Costa, and J. Barros, "Real-Time network coding for live streaming in hyper-dense WiFi spaces," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 773–781, Apr. 2014.

[55] G. Joshi, Y. Kochman, and G. W. Wornell, "The effect of block-wise feedback on the throughput-delay trade-off in streaming," in *IEEE INFOCOM Workshops*. IEEE, Apr. 2014, pp. 227–232.