UNIVERSITY OF CALIFORNIA,
IRVINE

Cross-Layer Optimization of Coded Wireless Networks

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Electrical and Computer Engineering

by

Hulya Seferoglu

2010

DEDICATION

To my parents

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT OF THE DISSERTATION

Cross-Layer Optimization of Coded Wireless Networks

By

Hulya Seferoglu

Doctor of Philosophy in Electrical and Computer Engineering

University of California, Irvine, 2010

Professor Athina Markopoulou, Chair

The network coding paradigm advocates that intermediate nodes should not only forward, but also process and combine packets, which has the potential to increase throughput and facilitate distributed operation of networks. This dissertation focuses on wireless networks, where network coding can be gracefully combined with and exploit the properties of the wireless networks. The goal is to design and evaluate algorithms and protocols, on top of given constructive network coding schemes, so as to fully exploit the network coding capabilities. The contributions of this dissertation are the joint optimization of (i) video streaming, (ii) rate control, and (iii) error correction, together with the underlying network coding mechanisms.

We first study video streaming over coded wireless networks. Our key insight is that, when the transmitted flows are video, network codes should be selected so as to maximize not only the network throughput but also the video quality. We propose video-aware opportunistic network coding schemes that take into account the importance and deadlines of video packets.

Second, we study rate control and scheduling. The key intuition is that network coding introduces network coded flows and new conflicts between nodes, which should be taken into account both in rate control and scheduling. We consider two types of traffic; video and TCP. In the case of video, its time-varying nature affects the underlying network coding opportunities. We observe that by delaying some scenes and by optimizing the rate allocation, we can create more network coding opportunities and thus improve video quality. In the case of TCP traffic, TCP flows do not fully exploit the network coding opportunities due to their bursty behavior and due to the fact that TCP is agnostic to network coding. In order to improve the performance of TCP flows over coded wireless networks, we propose a network-coding aware queue management scheme.

In the last part of this thesis, we combine inter- and intra-session network coding (I^2NC). Our scheme, I^2NC provides resilience to loss thanks to the error-correcting capabilities of intra-session network coding. Furthermore, it allows intermediate nodes to operate without the knowledge of the decoding buffers at their neighbors.

Chapter 1

Introduction

1.1 Motivation

Today's networks are based on the fundamental principle that the network forwards data, but the information itself is processed only at the end systems. Network coding is a research field that emerged over the past decade and breaks this fundamental assumption: it advocates that, in addition to forwarding packets, intermediate nodes should be allowed to also process and recombine several incoming packets into one or more outgoing packets [1, 2, 3, 4]. This showed that, in multicast networks where intermediate nodes do simple linear operations on incoming packets, one can achieve the min-cut capacity of the network to each receiver. The linearly combined packets can be utilized at the receivers to recover the original packets by solving a set of linear equations over a finite field. This breakthrough idea has spawned a significant effort [5, 6, 7, 8], initially in the information theory and computer science communities and more recently in the networking communities.

From a theoretical point of view, researchers have been studying the design of coding schemes and quantifying the benefits (in terms of throughput, delay, and robustness) as well as the cost of network coding, for various traffic scenarios and network topologies. From a practical point of view, researchers are exploring potential applications to practical networking problems at various layers of the protocol stack. For example, in the context of wireless mesh networks, one-hop opportunistic network coding has been shown to increase throughput by mixing packets from different flows into a single packet and by broadcasting over the wireless medium [9, 10, 11]. In the context of peer-to-peer content distribution, random network coding has been shown to facilitate distributed scheduling, reduce the download times and increase robustness to node failures [12, 13]. In the context of protocol design, intra-session network coding combined with retransmissions has been shown to successfully mask wireless losses from TCP congestion control [14]. There is also a growing body of work within the multimedia community that studies network coding techniques for multimedia and delay-sensitive traffic [15].

Despite the promise and significant amount of research activity generated over the last decade, network coding has not yet realized its full potential, especially in practical networked systems, because network coding introduces performance costs, novel security threats and it also introduces changes to the protocol stack. As a result, there is currently a gap between the theory of network coding and its practical applications. The goal of this thesis is to design and evaluate algorithms and protocols, on top of given constructive network coding schemes, so as to fully exploit the network coding capabilities and also address practical issues in coded wireless networks. More specifically, we are interested in understanding cross-layer issues such as video streaming, rate control, and error correction over coded wireless networks. This understanding is crucial for the deployment of network coding in practical networked systems.

1.2 Thesis Contributions

The contributions of this dissertation are the joint optimization of (i) video streaming (ii) rate control and (iii) error correction, together with the underlying wireless network coding mechanisms. More specifically, we make the following contributions:

- We study video streaming over coded wireless networks and we develop video-aware network coding algorithms [16], [17].
- We show the importance of network coding-awareness in rate control and scheduling over coded wireless networks. We optimize rate control protocols for video streaming and TCP [18], [19], [20], [21].
- We study the performance of network coding over lossy wireless networks and we develop loss-aware network coding algorithms along with optimal rate control protocols [22], [23].

In the next three subsections, we describe each contribution in more detail.

1.2.1 Video Streaming over Coded Wireless Networks

Developments in video compression and streaming, wireless networking, and cross-layer design, are continuously advancing the state-of-the art in wireless video [24]. Yet, providing high quality video over wireless networks is still a challenging problem due to limited bandwidth and time-varying nature of wireless links. Network coding has been shown to improve throughput by mixing packets from different flows and broadcasting the coded packets over the wireless medium [9, 10, 11]. With this advantage, network coding is a promising solution for video streaming over wireless networks. However, media traffic has some characteristics and requirements that introduce unique challenges and opportunities for network coding. In this study, we advocate the need for cross-layer design of video-aware network coding schemes that specifically take these features into account. Combining techniques from network coding and media streaming can make the best of both worlds.

We first consider the fact that different packets in the same multimedia stream may have different contributions to video quality. One important challenge with network coding is that it has been agnostic to the content of the packets that are coded together. In [17], [16], we consider video traffic transmitted over wireless networks with opportunistic one-hop network coding [10]. We design video-aware network coding schemes that take into account both the decodability of network codes by several receivers and the unequal importance of video packets, namely, distortion values and play-out deadlines. We demonstrate that these schemes improve the video quality without compromising the MAC throughput. In a sense, this work combines two orthogonal aspects of packet scheduling: (i) network coding to mix packets from different flows and increase throughput and (ii) radio-distortion optimized streaming of packets within the same stream to maximize video quality.

In addition to packet level difference, different multimedia streams may also have different level of importance according their traffic characteristics, sensitivity, or pricing. The challenge with network coding is the decision of determining which flows should be coded together when there are multiple media and/or data flows in the network. In this work, we assign importance to packets based, not only on their distortion value and playout deadline, but also on their traffic type and priority [17].

Finally, we consider strict delay requirement of media streaming and real-time communications, which poses both a challenge and an opportunity when network coding is used. On one hand, network coding increases delay due to additional encoding/decoding and possibly due to waiting at intermediate nodes for enough packets to arrive and be coded together. On the other hand, the increase in throughput can decrease the end-to-end delay. The design of scheduling and coding algorithms can trade-off throughput for delay so as to meet media requirements. In our work, [17], [16], we take the delay requirement into account, by incorporating it into the importance of a packet.

1.2.2 Rate Control and Scheduling over Coded Wireless Networks

Network coding has been shown to improve throughput by mixing packets from different flows and broadcasting the coded packets over the wireless medium [9, 10, 11]. However, it has non-trivial interactions with mechanisms in other layers of the protocol stack, namely: rate control, scheduling, and routing to fully exploit network coding benefit. Below, we present some key observations on the interaction of network coding and protocol stack.

Rate Control: Traditionally, the fundamental goal of transport protocols, such as Transmission Control Protocol (TCP), is to achieve as much bandwidth as possible while achieving some level of long-term rate fairness across competing flows. When network coding is employed in wireless networks, the achievable rate regions gets extended as compared to traditional routing. Interestingly, this has a non-trivial interaction with the rate requirements of applications at higher layers. The key observation is that at each node in the network which codes two flows, there should be packets from these two flows at each transmission instant to fully exploit network coding benefit. At the same time, transport protocols or higher layer flow control mechanisms should provide long term fairness. Therefore, transport protocols or in general terms, rate control mechanisms should be designed to be aware of the underlying network coding. In [18], through network utility optimization, we show that network coding awareness is crucial in rate control mechanisms in transport and flow control protocols.

Rate requirements of video applications vary over time due to time varying content of video. It is typical for video applications to have less motion - hence less rate requirements to satisfy a quality of service (QoS) - in some scenes, and to have high motion in consecutive scenes.

Thus, video streaming is an application that has their own, typically time-varying, rates that need to be adapted to match the rate region offered by the underlying coded wireless network. Conversely, the rates at the video/application layer affect the availability of network coding opportunities at the underlying network coding layer and thus the achievable region. In this work [19], we observe that by delaying some scenes and by optimizing the rate allocation over longer time intervals, we can create more network coding opportunities and thus achieve higher total utility.

TCP flows do not fully exploit the network coding opportunities due to their bursty behavior and due to the fact that TCP is agnostic to the underlying network coding. We observe that due bursty behavior of TCP, almost half of the time, there are no packets from the flows that would be network coded in the buffers of network coding nodes over wireless networks when buffer size is set to bandwidth-delay product. Due to this fact, TCP flows do not exploit full potential of network coding. In [21], [20], we formulated the problem as network utility maximization and we develop a distributed solution. Based on the structure of the optimal solution, we propose minimal modifications to congestion control and queue management mechanisms so as to make them network-coding aware. The results indicate that we are able to double TCP throughput over coded wireless networks.

Scheduling: Scheduling has two aspects in wireless networks: (i) which node and (ii) which flow should transmit at a given transmission instant. In current wireless networks, *e.g.,* 802.11, each nodes access medium according to CSMA/CA which basically provides fairness among the nodes trying to access the medium. After accessing the medium, each node transmits packets according to FIFO rule. However, when network coding is used, some flows are network coded and some are not. In our work [18], we show that network coding flows introduces more conflicts among the nodes trying to access the medium which should be considered in access mechanism. Furthermore, a FIFO scheme is not sufficient to exploit network coding benefits and provide fairness among the flows when network coding is used.

1.2.3 Performance of Network Coding over Lossy Wireless Networks

Dealing with loss in wireless networks is a hard enough problem, even without network coding. However, network coding among multiple flows amplifies the problem by exposing flows to loss not only on their own path, but also on the paths of flows that are coded together. Furthermore, side information to decode network codes is also prone to wireless link losses.

Error Correction for Network Coding: There is a wide spectrum of well-studied options for dealing with loss, *e.g.,* using redundancy (forward error correction) and/or retransmissions, locally (MAC) and/or end-to-end (transport layer). Local retransmissions increase end-to-end delay and jitter, which, if excessive, may cause TCP timeouts or hurt real-time multimedia. Furthermore, the best retransmission scheme for network coded packets varies with the channel loss probability and is hard to switch among re-transmission policies when the channel loss rate varies over time. Re-transmission also requires state synchronization to perform inter-session network coding, which is not reliable at all loss rates.

Network Coding for Error Correction: Although network coding adds extra challenges to error correction mechanism over lossy networks, network coding has some advantages: network coding has error correction capabilities similar to rateless codes, applied not only at the source but also at intermediate nodes. In the context of peer-to-peer content storage and distribution, random network coding has been shown to be more robust than traditional forward error correction against failures or departures of nodes [12], [13], [25]. The intuition is that, in case of a block being lost, network coding produces unique innovative blocks, while FEC-based schemes can replicate the same block (original or redundant). Similarly, over wireless networks, intra-session random network coding provides resilience to packet losses [14].

In this part of the thesis [23], we use error correction capabilities of network coding to improve network coding benefit over lossy wireless networks. Furthermore, our schemes are strong in the sense that they keep end-to-end delay low as compared to local and end-to-end re-transmissions, which improves TCP performance and is nice for multimedia applications. Although one could use various coding techniques, such as Reed-Solomon or Fountain codes, implementing error correction via intra-session network coding has several advantages. First, it has lower computational complexity than other error schemes. Second, in systems that already implement network coding among multiple flows, it is natural to incrementally add network coding for error correction functionality.

1.3 Overview and Organization

The rest of the dissertation is organized as follows. In Chapter 2, we provide the background necessary to understand basic network coding principles and the methodological tools used in this work. We also describe related work. Chapters 3, 4, and 5 are the core of the thesis. In Chapter 3, we present video-aware network coding. In Chapter 4, we first discuss the necessity of rate control and scheduling over coded wireless networks. Second, we consider video streaming and TCP over coded wireless networks, we study their rate requirements and their interaction with network coding, and we propose updates in the cross layer design. In Chapter 5, we study the performance of network coding over lossy wireless networks and we propose the joint use of intra- and inter-session network coding to improve network coding over such networks. Finally, we conclude in Chapter 6 with a summary of our contributions along with our final remarks.

Chapter 2

Background & Related Work

2.1 Network Coding

The network coding paradigm has emerged from the pioneering work in [1, 2], which showed that, in multicast networks where intermediate nodes do simple linear operations on incoming packets, one can achieve the min-cut throughput of the network to each receiver. The linearly combined packets can be utilized at the receivers to recover the original packets by solving a set of linear equations over a finite field. To better illustrate this point, let us discuss the following example.

Example 1 The example shown in Fig. 2.1 illustrates multicasting with network coding over wired butterfly topology. Each link capacity is equal to one packet per time slot. In this topology, source S transmits a flow to receivers R_1 and R_2 over the nodes B, C, I_1, and I_2. Source S transmits two packets a and b over links $A - B$ and $A - C$. These packets are forwarded to I_1 over the links $B - I_1$ and $C - I_1$. The link $I_1 - I_2$ is a bottleneck in this topology according to traditional routing in which packets a and b are transmitted in two time slots. With network coding, packets a and b are linearly combined (over a finite

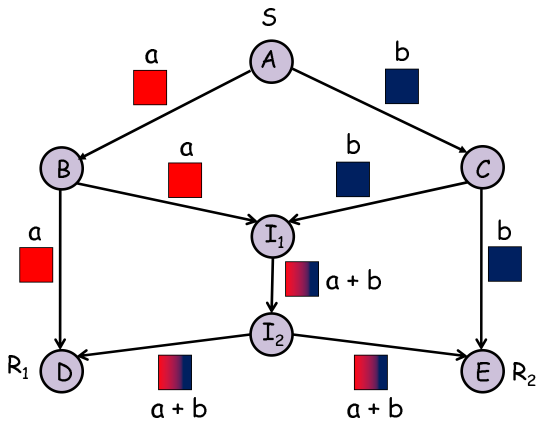


Figure 2.1: Butterfly topology.

field) and transmitted. Also, packet a is transmitted over $B - D$ and packet b is transmitted over $C - E$. As a result, receivers R_1 and R_2 receive two packets at each time slot. This is expected according to network coding theorem which shows that one can achieve the min-cut capacity of the network (which is two in this example) when intermediate nodes do simple linear operations on incoming packets. □

The breakthrough idea of network coding inspired significant effort in several directions [26, 27, 28], including practical application of network coding, studying topologies beyond multicast, such as unicast [29, 9, 30] and broadcast scenarios. The broadcast nature of the wireless medium offers an opportunity for exploiting the throughput benefits of network coding [31, 32]. These ideas are applied in the networking community in the context of wireless mesh network [10, 11]. COPE [10] implemented a pseudo-broadcast mechanism for 802.11 together with opportunistic listening and a coding layer between IP and MAC that is used to detect coding opportunities and pack packets from different flows into a single transmission, thus increasing network throughput. To better illustrate the network coding advantage over wireless networks with unicast flows (which we call inter-session network coding), let us discuss the following example.

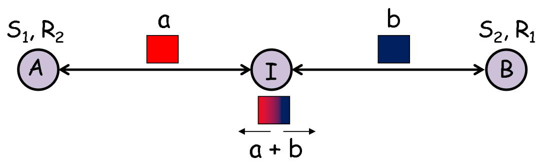


Figure 2.2: Alice-and-Bob topology.

Example 2 The example shown in Fig. 2.2 illustrates Alice-and-Bob topology. We assume that link capacities are one packets per time slot. In this topology, source S_1 transmits a flow to receiver R_1 and source S_2 transmits a flow to receiver R_2, over the intermediate node I. A and B transmit their packets a and b, in two time slots, and node I receives them. Furthermore, B knows b and A knows a, because they already transmitted these packets. In the next time slot, I broadcasts the network coded packet, $a \bigoplus b$ to A and B. Since B and A knows b and a, they can decode their packets a and b, respectively. Since I can transmit $a \bigoplus b$ in one time slot, instead of a, b in two time-slots, network coding reduces four transmission to three, and improves throughput by 33.3%. $\square$

The network coding benefit over wireless networks with unicast flows has generated a lot of interest in the research community. One-hop network coding in [10] is extended to include new coded wireless systems such as BFLY [33], pairwise network coding [34], tiling approaches [35], [36]. Network coding over wireless networks are modeled and analyzed by [37], [38], [39]. The performance of one hop network coding is considered in [40] by looking at the interaction of network coding with MAC fairness. [41] addressed the problem of under utilization of one-hop network coding with TCP.

The advantages of network coding are extended from inter-session network coding to intra-session network coding (in which packets in the same flow are coded and transmitted) over wireless networks. In this context, random network coding has been shown to facilitate routing, and increase robustness to node and link failures [42], [43]. The error-correcting capabilities of intra-session network coding have recently been used in conjunction with the

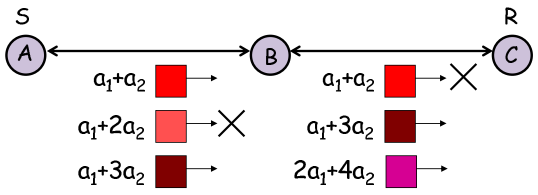


Figure 2.3: Error correction with network coding.

TCP sliding window in [44]. To better explain the error correction capabilities of network coding, let us consider the following illustrative example.

Example 3 The example shown in Fig. 2.2 illustrates error correction capability of network coding. In this topology, source S transmits a flow to receiver R over the node B. We assume that link capacities of $A - B$ and $B - C$ are one packets per time slot and loss probability over each link is $1/3$. In this setting, two packets a_1 and a_2 are transmitted with intra-session network coding, *i.e.,* three packets; $a_1 + a_2$, $a_1 + 2a_2$, and $a_1 + 3a_2$ are generated and transmitted by source S considering $1/3$ loss probability over link $A - B$. At node B, only $a_1 + a_2$ and $a_1 + 3a_2$ are received. Node B creates one more parity packet $2a_1 + 4a_2$ from its received packets and transmits three packets to the receiver. The receiver R receives packets $a_1 + 3a_2$ and $2a_1 + 4a_2$ and decodes a_1 and a_2. $\square$

To summarize, network coding is promising in several aspects; it improves throughput, facilitates scheduling and routing, and increases robustness to node and link failures. In this work, our objective is to optimize and design algorithms to exploit the advantages of network coding for specific applications such as video streaming, improve the network coding benefit by optimizing cross layer interaction of network coding and protocols, and exploit the network coding benefit in lossy wireless networks.

2.2 Video Streaming

Developments in video compression and streaming, wireless networking, and cross-layer design, are continuously advancing the state-of-the art in wireless video [24]. Yet, providing high quality video over wireless networks is still a challenging problem due to limited bandwidth and time-varying nature of wireless links.

Several network-adaptive techniques have been proposed to support streaming media over unreliable and/or time-varying networks [45]. Supporting video over wireless is particularly challenging due to the limited, time-varying resources of the wireless channel [24]. There is a large body of work on cross-layer design for video over wireless, such as [46, 47, 48, 49], exploiting the fact that packets in a video stream have different importance and therefore should be treated differently by network mechanisms. Packet scheduling is an important control at the medium access control layer. The problem of rate-distortion optimized packet scheduling has been studied in the RaDiO family of techniques [50, 51, 52, 53, 54, 55]: in every transmission opportunity, media units are selected for transmission so as to maximize the expected quality of received video subject to a constraint in the transmission rate, and taking into account transmission errors, delays and decoding dependencies.

Network coding is a promising solution for video streaming over wireless networks due to its throughput improvement which is crucial in video streaming. Combining techniques from network coding and media streaming can make the best of both worlds. Below are some fundamental properties of multimedia traffic and their implications for network coding [56].

Unequal Packet Importance: The fact that different packets, within the same media stream, have different contributions to distortion (due to video content, encoding, or playout deadlines) is well understood in the multimedia community. This fact lies at the heart of multimedia streaming: the unequal importance of packets is used to guide prioritized transmission over a network. Depending on the transmission scenario, available differentiation mechanisms are used to ensure that the most important packets of a particular stream are given priority, thus providing a graceful degradation in the presence of adverse network conditions. One challenge that arises from this fundamental property of multimedia, with respect to network coding, is that network coding, so far, has been agnostic to the content of the packets that are coded together. In inter-session network coding, the goal is to mix together several packets from different flows, thus increasing the information per packet and eventually the throughput. However, for media streaming it is not only the quantity of delivered packets that matters but also their quality.

Different Flow Characteristics and Requirements: Moving from the granularity of packets to the granularity of flows, we observe that entire flows may also have different importance, *e.g.,* due to their traffic characteristics, sensitivity or pricing. When there are multiple media and/or data flows in a system, the question is which flows should be coded together? The rate and delay requirements of media streams should be taken into account when deciding which of them to code together and/or with data flows.

Delay Requirements: Another inherent characteristic of media streaming and real-time communications is that they have strict delay requirements, which poses both a challenge and an opportunity when network coding is used. On one hand, network coding increases delay due to additional encoding/decoding and possibly due to waiting at intermediate nodes for enough packets to arrive and be coded together. On the other hand, the increase in throughput can decrease the end-to-end delay. The design of scheduling and coding algorithms can trade-off throughput for delay so as to meet media requirements.

There is also a growing body of work within the multimedia community that studies network coding techniques for multimedia and delay-sensitive traffic. [57] looked a downlink scenario and formulated the scheduling and coding problem within a Markov decision process framework, which can also incorporate delay through its contribution to distortion. In the context of generation-based network coding, the throughput vs. delay tradeoff can be explicitly controlled by tuning the generation size [27]. A detailed review on the interaction of network coding and multimedia streaming can be found in [15]. In this work, we consider specific media traffic characteristics and requirements (such as the difference importance of packets, the strict delay requirements and the time-varying video rate) that introduce unique challenges and opportunities for network coding, and we advocate the need for cross-layer design of video-aware network coding schemes that specifically take these features into account.

2.3 Network Utility Maximization

Resource allocation problems in network utility maximization framework (NUM), including rate control, scheduling, and routing, have been extensively studied in communication networks, [58], [59], [60]. Resource allocation problems have also been studied specifically in the context of wireless networks, which are more challenging due to their dynamic, time-varying and multi-access nature; an excellent review can be found in [61]. In [62], joint routing, rate control and scheduling in wireless networks has been studied and two approaches have been proposed for the problem: a node-centric and a link-centric approach. Both can decompose the problem but the scheduling part remains hard; to overcome this problem a greedy approach has been proposed in [63] and implemented in [64].

Resource allocation problems when network coding is used are gaining interest, especially for multicast flows. In [65], schemes were proposed for minimum cost multicast over network coded wireline and wireless networks. This work was extended for rate control in [66] for wireline networks. Another extension was [67], on rate control, routing and scheduling for intra-session network coded wireless networks, using the generation-based network coding proposed in [26]. The rate region when network coding is used was studied in [68, 69]. Optimal scheduling and optimal routing for COPE are considered in [37] and [39], respectively. Network utility maximization is used in [34] for end-to-end pairwise inter-session network coding. Energy efficient opportunistic inter-session network coding over wireless are proposed in [70], following a node-based NUM formulation and its solution based on back-pressure. A linear optimization framework for packing butterflies is proposed in [71]. Forward error correction over wireless for pairwise network coding in network utility maximization framework is proposed in [72].

In this work, we focus on the NUM problem for multiple unicast flows over wireless with a given inter-session network coding scheme. We first discuss the necessity of network coding-aware NUM formulation. We consider video specific rate requirements jointly with network coding in NUM formulation. We focus on the congestion control problem for TCP over coded wireless networks. To the best of our knowledge, our work is the first, to take the step from theory (optimization) to practice (protocol design), specifically for the problem of congestion control over inter-session network coding. We propose implementation changes, which have a number of desired features: they are justified and motivated by analysis, they perform well (double the throughput in simulations), and they are minimal (only queue management is affected, while TCP and MAC remain intact). Finally, we jointly consider intra- and inter-session network coding NUM framework to improve network coding performance over lossy wireless networks.

2.4 Summary

Network coding is a promising technique to improve throughput, facilitate scheduling and routing, and increase robustness to node and link failures. It has generated significant amount of research activity and it is obvious that in its interaction with applications such as video streaming or protocols such as transport protocol (TCP), scheduling, and routing, joint optimization and cross layer design are required. Network utility maximization framework facilitates the understanding of cross layer design.

Chapter 3

Video-Aware Opportunistic Network Coding

Providing high quality video over wireless networks is a challenging problem, due to both the erratic and time-varying nature of a wireless channel and the stringent delivery requirements of media traffic. In this chapter, we propose a novel technique for video streaming in a wireless environment inspired by network coding paradigm [28, 26, 27].

Our work builds on [11, 10] that used network coding to improve throughput in a wireless mesh network. In particular, [11, 10] proposed that wireless routers mix packets from different flows, so as to increase the information content of each -broadcast- transmission and therefore the throughput for data applications. In this work, we build on this idea, and propose a network coding and scheduling scheme for transmitting several video streams over a wireless mesh network.

Our key insight is that the transmission of video streams over coded wireless networks should be optimized not only for network throughput but also, and more importantly, for video quality. The fact that video packets have unequal importance is well understood and extensively studied in the video streaming community, *e.g.,* for rate-distortion optimized streaming [50, 52, 51, 53]. The fact that mixing different information flows can increase throughput in multicast networks is well understood in the network coding community [28, 1, 2]. Our work bridges the gap between the two approaches, and proposes a new video-aware scheme for network coding and packet scheduling that improves both aspects, namely video quality and throughput.

In this work, we consider a wireless mesh network, in which routers can mix different incoming flows/streams, using simple network coding operations (XOR). The resulting network code is broadcast to the neighborhood of the router. Nodes in the same neighborhood listen to each other′s transmission and store overheard packets; these are used later to decode received coded packets and also to construct new coded packets. The core question in this architecture is how to select the best -according to an appropriate metric- network code for transmission among all possible codes. In [10, 11], a transmitting node chooses a network code that can be decoded by several neighbors at the same time slot; this policy increases the information per packet transmission thus the throughput. However, when the transmitted flows are video streams, this is not necessarily the best choice. Video quality can be improved by intelligently selecting network codes that combine those video packets that are decodable by several neighbors but also contribute the most to video quality. In other words, when video streams are transmitted, it is not only the quantity but also the quality/content of information transferred that should be taken into account in the selection of network codes. In this chapter, we develop schemes for network code selection and packet scheduling that take into account both (i) the importance and deadlines of video packets and (ii) the network state and the received/overheard packets in the neighborhood.

In the rest of this chapter, we first give an overview of the system model in Section 3.1. Section 3.2 presents the algorithms for network coding. Section 3.3 presents simulation results that demonstrate the benefits of the proposed algorithms over baseline schemes, in

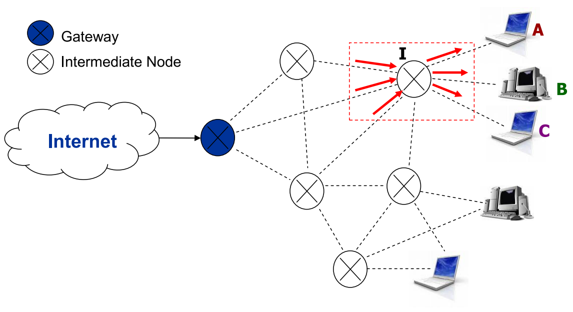


Figure 3.1: A wireless mesh network. I is an intermediate node, A, B, C are receiving (and/or sending) nodes.

terms of video quality and application-level throughput. Section 3.4 concludes the chapter.

3.1 System Model

We consider video streaming over wireless mesh networks where intermediate nodes (wireless mesh routers) are able to forward packets to other intermediate nodes and/or clients, as shown in Fig. 3.1. In this chapter, we propose algorithms that can be used at the intermediate node to maximize video quality and throughput. We assume that intermediate nodes can perform simple network coding operations (bit-wise XOR) and combine packets from several incoming streams into a single outgoing packet. This packet is broadcast to the entire neighborhood, thus reaching several nodes at the same time. We assume that nodes can overhear all transmissions in their neighborhood, whether they are intended for them or not; they can decode a network-coded packet using overheard packets. The idea of combining network coding with broadcast to increase the information content per transmission, is well understood in the network coding community. This idea has been applied in 802.11-based multi-hop wireless networks and throughput benefits have been demonstrated for data applications [10].

Our key observation is that, when the transmitted flows are video streams, this is not necessarily the best choice and video quality must also be considered. The importance and deadlines of video packets must be taken into account to select those codes that contribute the most to the quality of video streams. In this chapter, we develop schemes for network coding across different flows, and packet selection within each flow, to improve both video quality and throughput.

3.1.1 Code Selection at an Intermediate Node:

Let us consider an intermediate node that receives N packets from different video streams and forwards them to N nodes in its neighborhood. The intermediate node maintains a transmission (Tx) queue with incoming video packets. At a given time slot a packet is selected from the Tx queue for transmission. The selected packet is called the primary packet and its destination node is called the target node. The primary packet can be thought as the main packet we try to transmit during a time slot. Depending on the network coding scheme, the primary packet may be the first packet from the head of the queue, or any packet in Tx queue that is marked as active (*i.e.,* not transmitted within the last round-trip time, as discussed later). In addition to the primary packet, all packets in the queue are considered as candidate side packets, *i.e.,* candidates for a transmission in the same time slot together with the primary packet; they are useful to nodes other than the target node. The primary and the side packets are all XOR-ed together into a single packet, called the network code. The core question then is: which network code (*i.e.,* XOR of the primary and side packets) to select and transmit so as to maximize the total video quality and throughput. The algorithms addressing this question are the main part of this chapter, and will be discussed separately in the next section (3.2). In the rest of this section, we describe the remaining components and functions of the system. The terminology is summarized in Table 3.1.

Table 3.1: Terminology

Term	Definition
Primary Packet	The packet selected from the Tx queue before network coding. It must be included in all network codes. It can be thought as the main packet we try to transmit in a given time-slot.
Side Packet	Packet in the Tx queue, other than the primary, included in the network code.
Network Code	One primary and all side packets XOR-ed together into a single packet.
Active Packet	Packet in the Tx queue that can be considered as primary. (Not transmitted within the last RTT.)
Inactive packet	Packet in the Tx queue that cannot be considered as primary. (It has already been transmitted within the last RTT, and the acknowledgement is still pending.)
Target Node	The intended recipient of the primary packet.
Tx Queue	The output queue of the transmitting node.
Rx Buffer	The receive queue of the receiving node. It stores received packets, destined to this node.
Virtual Buffer	Also maintained at a receiving node. It stores overheard packets, destined to other nodes.

3.1.2 Receiving, Overhearing and ACKing a Packet (at Receiving Nodes)

Once the network code is chosen, it is broadcast to all nodes in the neighborhood. Depending on the channel conditions, some nodes successfully receive it. When the target node receives it, it decodes it (which is guaranteed by the construction of the network code in the next section), stores the primary packet in its receive (Rx) buffer, and sends an acknowledgement (ACK) back to the intermediate node. Nodes, other than the target node, overhear the transmitted packet and try to decode it; if they overhear a new packet destined to them, they store it in their Rx buffer and send an ACK back to the intermediate node; if they obtain a packet destined for another node, they store it in their virtual buffer. An overheard packet stays in the virtual buffer until an ACK from the target is overheard or until its deadline expires. We also assume that the asynchronous ACK mechanism, proposed in [10], is used to combat ACK implosion.

The intermediate node waits for a mean round-trip time (RTT) from the time it transmits the network code until it receives an ACK. During that period, all packets that were part of the code stay in the Tx queue but are marked as inactive. Inactive packets are not considered for primary transmission (in order to avoid unnecessary duplicate transmissions) but are still considered as candidates for side packets (to increase coding opportunities). When the transmitter receives an ACK, it removes the corresponding packet from the Tx queue. If an RTT expires without receiving an ACK, the packet is marked as active again and the process is repeated. A packet stays in the Tx queue until either it is successfully transmitted or its deadline expires; when either of these occur, the packet is removed from the transmission buffer.[1]

3.1.3 Requirements

Our system relies on the following capabilities. First, broadcast is needed to harvest the benefits of network coding. Although wireless is inherently a broadcast medium, this may be hidden by some communication protocols. We make use of the broadcast capability, implemented as pseudo-broadcast on top of 802.11 unicast in [10, 11]. Second, nodes need to learn the contents of the virtual buffers of all their neighbors, in order to select a code that is decodable in their neighborhood. This can be achieved by explicitly exchanging and/or implicitly learning this information as in [10, 11]. Third, nodes must be capable of coding/decoding in real time, which is a realistic assumption for simple (bit-wise XOR) operations. Network coding is implemented as a thin layer between IP and MAC, exactly as

[1] Note that although a transmitted packet remains inactive for an RTT, it does not block the head of the queue: the next active packets in the queue are coded and transmitted during this period. Also note that, although the Tx queue is basically a FIFO, considering any active packet as primary may lead to reordering in packet delivery. Although this may be a concern for TCP, as it was the case in [10], it is clearly better for video that requires timely delivery and can reorder packets at the playout buffer.

in [10]. Nodes are considered fixed (not mobile) and routing is considered given, *i.e.,* decided by a routing module orthogonal to the network coding algorithms considered in this chapter.

3.1.4 Importance of Packets

Nodes make network coding decisions taking into account the importance of video packets. The distortion value (Δ) of every packet can be determined by the source and communicated to the intermediate nodes in order to enable them to take decisions about transmission of video units in a rate-distortion optimized manner [50]. This information can be marked on a special field of the packet header. This field can be at the application level (*e.g.,* RTP extended headers) or part of the network coding header; alternatively, the typically unused TOS/DiffServ byte in the IP header can be overridden. In addition to the individual importance of packets (Δ) within a flow, our formulation also considers the importance of flows (γ). In general, the overall importance of a packet can be a function of the flow priority and the packet distortion value; in this chapter, we use a simple product $\gamma \cdot \Delta$.

The main focus of this chapter is on network coding for video, and most of the discussion is presented in terms of queues that contain only video packets. This could be implemented in practice on top of 802.11e, using the differentiation mechanisms to separate real-time traffic (in our case video) from data traffic. Our network coding algorithms could then be applied only on the video queue. Independently, our framework could also handle a mixture of video and data packets in the same queue by assigning them different flow priorities.

3.2 Video-Aware Network Coding Algorithms

The main questions we consider in this chapter have to do with the construction and selection of network codes. The code construction problem is concerned with finding candidate codes that guarantee decodability by the target node. The code selection problem is concerned with selecting the best among the candidate codes so as to optimize video quality. The first proposed algorithm, NCV, achieves the same throughput gains as in [10] but also intelligently chooses the network codes that maximize video quality. The second algorithm, NCVD, uses NCV as a building block but considers more coding options thus further improving video quality and throughput. The third algorithm, NC-RaDiO, generalizes these ideas: it extends the rate-distortion optimized (RaDiO) packet scheduling framework [50], so as to find the optimal network coding and transmission policy at every transmission opportunity.

3.2.1 NCV Algorithm: Network Coding for Video

Assume that there are several video streams coming to an intermediate node. Depending on the content of the virtual buffers at the clients, there may be several combinations of these streams, *i.e.,* several network coding opportunities. The main idea behind the Network Coding for Video (NCV) algorithm is to select the best network code to improve video quality. The following example demonstrates this idea.

Example 4 Consider the example shown in Fig. 3.1 and let us focus on a single-hop shown in more detail in Fig. 3.2. Node I receives three independent video streams, *e.g.,* from the Internet through the gateway, destined to its neighbors A, B, C. I maintains a FIFO Tx queue that stores packets $\{A_1, A_2, ...\}$ destined to node A, $\{B_1, B_2, ...\}$ destined to node B, and $\{C_1, C_2, ...\}$ destined to node C. Fig. 3.2 also shows the contents of the virtual buffers at each client: node A has overheard packets $\{B_1, C_1\}$ and nodes B and C have both overheard packet A_1, from previous transmissions. A_1 is the first active packet from head of the queue and is selected as the primary packet. Any packet (active or inactive) in the output queue, other than A_1, can be chosen as a side packet, on the condition that the constructed network code should be decoded at node A, *i.e.,* A_1 can be retrieved. To satisfy this condition, side

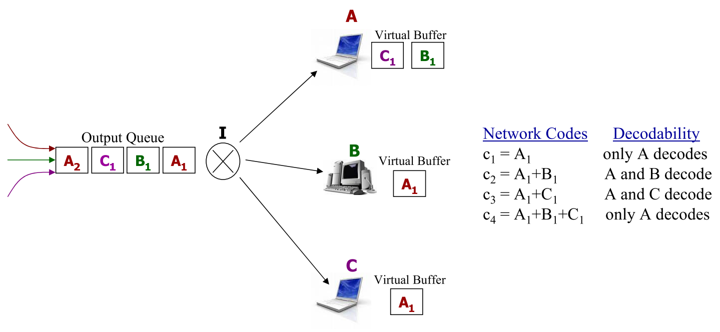


Figure 3.2: Example of Network Coding for Video (NCV), for a one-hop downlink scenario with three different streams.

packets that will be used in the network code should already be available at node A; in other words, the decodability of a network code depends on the overheard packets at node A. Network codes $c_1 = A_1$, $c_2 = A_1 \oplus B_1$, $c_3 = A_1 \oplus C_1$, and $c_4 = A_1 \oplus B_1 \oplus C_1$ can all be decoded by A and thus are eligible network codes. $\square$

The Code Construction Problem

More generally, consider that there are N nodes $\mathbf{N} = \{n_1, n_2, ..., n_N\}$ in the wireless network. Consider an intermediate node $n \in \mathbf{N}$, which transmits to its neighbor nodes. Let ϕ_n be the number of packets in the Tx queue of node n, and the packets themselves be $\Phi_n = \{p_1, p_2, ..., p_{\phi_n}\}$. Choose the first active packet, p_i, from the head of the Tx FIFO queue as the primary packet; p_i has a target node $t(p_i) \in \mathbf{N}$. Node n will construct and broadcast a network code, which consists of the primary packet p_i XOR-ed together with some side packets, so that the target node $t(p_i)$ can decode and obtain p_i. For this to be guaranteed, all side packets must be among the packets that are already overheard at the target $t(p_i)$. Assume that $\psi_{t(p_i)}$ packets are overheard at node $t(p_i)$ and denoted by

Algorithm 1 The NCV Algorithm

```
 1: Initialization: $I^i_{max} = 0$, $c^i_{max} = \emptyset$
 2: Choose the first head-of-queue active packet as primary $p_i$.
 3: Let $t(p_i)$ be the target node of packet $p_i$. Let $\{\nu_1, ..., \nu_{\Psi_t(p_i)}\}$ be the overheard packets at $t(p_i)$.
 4: for $k = 1...2^{\psi_t(p_i)}$ do
 5:     $c^i_k = \{p_i\} \bigcup S^{t(p_i)}_k$
 6:     Calculate $I^i_k$ with Eq. (3.5)
 7:     if $I^i_k > I^i_{max}$ then
 8:         $I^i_{max} = I^i_k$, $c^i_{max} = c^i_k$
 9:     end if
10: end for
11: Choose $c^i_{max}$ as the network code. XOR all packets and transmit
```

$\Psi_{t(p_i)} = \{\nu_1, \nu_2, ..., \nu_{\Psi_t(p_i)}\}$. Therefore, the candidate network codes at node n are:

$$c^i_k = \{p_i\} \bigcup S^{t(p_i)}_k, k = 1, 2, ..., 2^{\psi_{t(p_i)}} \tag{3.1}$$

where $S^{t(p_i)}_k$ is the k^{th} subset of $\Psi_{t(p_i)}$. Note that, since linear operations are limited to bit-wise XOR, a network code $p_1 \oplus p_2 \oplus ... \oplus p_k$ is completely specified by the set of packets $\{p_1, p_2, ..., p_k\}$ that are XOR-ed together. The next step is to select the best among all candidate codes.

Example 4 Continued Node A can get packet A_1 from all four possible network codes. Codes c_2 and c_3 improve the video quality at node sets $\{A, B\}$ and $\{A, C\}$, respectively. It is clear that c_2 and c_3 are better codes than c_1 and c_4 both for throughput (they are useful to two instead of one node) and video quality. Comparing c_2 to c_3, we observe that they are equivalent in terms of throughput but they may contribute differently to video quality depending on the content of video packets A_1, B_1, C_1. Deciding which candidate code to select between $c_2 = A_1 \oplus B_1$ and $c_3 = A_1 \oplus C_1$ should depend on the importance and urgency of the original video packets B_1 and C_1. NCV exploits this observation. $\square$

The Code Selection Problem

After constructing all candidate codes at a node n, we need to choose the best code according to an appropriate metric, which we define here so as to capture the contribution of each candidate code to the video quality improvement. Recall that p_i is the primary packet targeted to node $t(p_i)$, and $\{c_k^i\}_{k=1}^{k=2^{\psi_{t(p_i)}}}$ are all the candidate codes. Let $I_k^i(n_\eta)$ be the improvement in video quality at node n_η for $\eta = 1, 2, ..., N$, when c_k^i is received and decoded:

$$I_k^i(n_\eta) = \sum_{l=1}^{L_k} (1 - P(l))\Delta(l)\gamma(l)g_l^k(n_\eta)d_l^k(n_\eta) \tag{3.2}$$

where each factor in this formula is defined as follows:

- L_k is the number of original packets included in network code c_k^i. Notice that at most one out of these L_k packets can be useful to a particular node n_η, but different packets are useful to different nodes.
- $d_l^k(n_\eta)$ and $g_l^k(n_\eta)$ are indicator functions that express whether code k is useful for node n_η. We define $d_l^k(n_\eta) = 1$ if c_k^i is decodable at node n_η, or 0 otherwise. We define $g_l^k(n_\eta) = 1$ if packet l is targeted to node n_η, or 0 otherwise.
- $\Delta(l)$ is the improvement in video quality (PSNR) if packet l is received correctly and on time at client n_η. To compute $\Delta(l)$, we decode the entire video sequence with this packet missing and we compute the resulting distortion.[2] We assume that this computation is performed at the source offline and that the distortion value is marked on each packet.[3]

[2]This is an approximation as the actual distortion that may also depend on the delivery status of prior and subsequent NALs. The distortion model can be extended to capture these loss correlations [73, 74, 75]. Furthermore, we assume that distortions caused by loss of multiple packets are additive, which is reasonable for sparse losses. These approximations reduce the computational complexity by separating the total distortion function into a set of individual packet distortion functions and optimizing for each one of them.

[3]For real-time traffic, one can still estimate the distortion by performing online analysis with a delay of a

- $\gamma(l)$ is the importance/priority of the flow that packet l belongs to. All packets in the same flow have the same (flow) importance, but different flows may have different importance. If some flows are more important, then higher importance should be assigned to them; otherwise they should all be assigned $\gamma = 1$. If the average quality differs among encoded video sequences and we still want to treat flows equally, the flow importance can be a normalization factor (the inverse of the average PSNR per sequence); this is what we do in the simulations.

- $P(l)$ is the probability that packet l is lost due to either channel errors or late arrival for playout:

$$P(l) = P\{FTT' > t_d(l) - t_c\} \tag{3.3}$$

where $t_d(l)$ is the deadline of packet l, t_c is the current time and $\tau = t_d(l) - t_c$ is the remaining time until the playout deadline; FTT' is the forward trip time in the presence of delay and loss. The complementary cumulative distribution function of FTT' can be calculated as follows:

$$P\{FTT' > \tau\} = \varepsilon_F + (1 - \varepsilon_F) \int_{\tau}^{\infty} p_F(t)dt \tag{3.4}$$

The first part in Eq.(3.4) describes the probability that a packet is lost in the forward channel, due to noise, fading, and interference in the wireless. The second part in Eq.(3.4) describes the probability that a packet, which is not lost, arrives late, *i.e.,* after its playout deadline; $p_F(t)$ is the distribution of the forward-trip time.

few frames. Most distortion occurs in the first few frames after a loss and breaks after the next I frame; the error depends on the video content of subsequent frames and on the coding decisions. Another approach is to assign distortion values based solely on the GOP structure, ignoring the video content and coding decisions, or to use a model for dependencies [75].

After defining the contribution of code c_k^i to the video quality at a single node n_η, $I_k^i(n_\eta)$, we define the total video quality improvement of code c_k^i as the sum of the video quality improvements at all clients $\eta = 1, ...N$, due to code c_k^i:

$$I_k^i = \sum_{\eta=1}^{N} I_k^i(n_\eta) \tag{3.5}$$

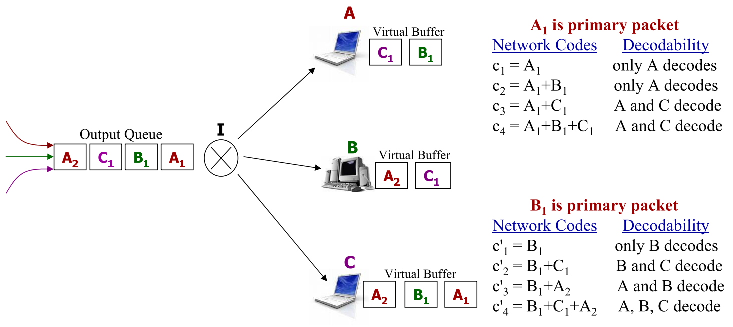
The NCV algorithm is summarized in Alg. (1). At each time slot, the NCV algorithm chooses the primary packet p_i and constructs all candidate network codes $\{c_k^i\}_{k=1}^{k=2^{\psi_t(p_i)}}$. Among all candidate network codes, NCV chooses the code that maximizes the total video quality improvement:

$$\max_k I_k^i \tag{3.6}$$

Depending on the contents of the virtual buffers, it is possible that no side packets can be used together with a given primary packet p_i. In that case, the network code is simply $\{p_i\} \cup \emptyset = \{p_i\}$.

3.2.2 NCVD Algorithm: Looking Into The Queue in Depth

As described, NCV selects the primary packet from the head of the queue but ignoring packets marked as inactive, and then optimally chooses the side packets. However, the fact that NCV does not optimize the primary packet has two implications: (i) the primary packet itself is important for video quality and (ii) the candidate side codes are limited to those that are decodable for this single primary packet. The second algorithm improves over NCV by also optimizing the selection of the primary packet. NVCD looks into the entire Tx queue in depth and considers all, not just the head-of-line, packet as candidates for the primary packet, thus increasing the options for candidate codes. A different set of candidate



A_1 is primary packet

Network Codes	Decodability
$c_1 = A_1$	only A decodes
$c_2 = A_1+B_1$	only A decodes
$c_3 = A_1+C_1$	A and C decode
$c_4 = A_1+B_1+C_1$	A and C decode

B_1 is primary packet

Network Codes	Decodability
$c'_1 = B_1$	only B decodes
$c'_2 = B_1+C_1$	B and C decode
$c'_3 = B_1+A_2$	A and B decode
$c'_4 = B_1+C_1+A_2$	A, B, C decode

Figure 3.3: Example of NCVD in the scenario of one-hop downlink transmission of three different receivers.

codes can be constructed for each primary packet. We explain NCVD through the following example.

Example 5 Let us look at Fig. 3.3. The topology is the same as in Fig. 3.2, but the contents of the Tx queue and of the virtual buffers are different. Assume that all packets are active packets, *i.e.*, they can all be considered as primary. One option is to select the head-of-line packet A_1 as the primary packet. As discussed in Example 1, the best codes for this primary packet are $c_3 = A_1 \oplus C1$ or $c_4 = A_1 \oplus B_1 \oplus C_1$. A different choice is to select B_1 as the primary packet, which leads to a different set of candidate network codes (listed on the Fig. 3.3). Code $c'_4 = B_1 \oplus C_1 \oplus A_2$ achieves the maximum throughput improvement, and potentially the maximum video quality, depending on the importance and urgency of all packets. This example demonstrates that increasing our options of primary packet, increases the set of candidate codes, and thus can improve both throughput and video quality. $\square$

More generally, NCVD constructs candidate codes $c^i_k, k = 1, 2, ..., 2^{\psi_{t(p_i)}}$ for each candidate primary packet p_i in the Tx queue. Among all constructed codes, NCVD selects the code

Algorithm 2 The NCVD Algorithm

```
 1: Initialization: $c_{max} = \emptyset$, $I_{max} = 0$
 2: for every packet $i = 1, ..., , \phi_n$ from the head of Tx queue do
 3:     Consider this packet, $p_i$, as candidate for primary
 4:     Construct all possible codes $c_k^i$ for $p_i$
 5:     Determine the max improvement $I_{max}^i = \max_k I_k^i$
 6:     and the corresponding code $c_k^i$: $k = argmax I_k^i$ as in NCV
 7:     if $I_{max}^i > I_{max}$ then
 8:         $I_{max} = I_{max}^i$, $c_{max} = c_k^i$
 9:     end if
10: end for
11: Choose $c_{max}$ as the network code. XOR all packets and transmit.
```

that maximizes the total improvement in video quality for all clients:

$$\max_{p_i} \max_k (I_k^i) \tag{3.7}$$

Algorithm 2 summarizes NCVD.

NCVD can be parameterized by the depth d of the Tx queue considered in the selection of the primary packet. NCVD($d = 1$) is simply NCV, while NCVD($d = \infty$) considers all packets in the Tx queue. The larger the value of d, the more coding options, the better the performance of NCVD. Because queue sizes are small for real time applications, we can focus on NCVD($d = \infty$), simply referred to as NCVD.

3.2.3 NC-RaDiO: Rate-Distortion Optimized Network Coding

The NCV and NCVD algorithms choose the network code for the next transmission opportunity, so as to maximize the video quality. Now, we formulate this problem within the rate-distortion optimized (RaDiO) packet scheduling framework [50]. Starting from the RaDiO formulation [50], especially for multiple streams sharing the same medium [51], we modify and extend it to account for network coding, instead of just packet, transmission policies (NC-RaDiO). We show how to find the optimal solution and that our previous algorithms (especially NCVD) are efficient heuristic solutions to the general NC-RaDiO optimization problem under some mild assumptions.[4]

Formulation

Let us consider a single node $n \in \mathbf{N}$ in the wireless mesh network, with packets $\Phi_n = \{p_1, p_2, ..., p_{\phi_n}\}$ in its queue, and let us focus on a single transmission opportunity. Without network coding, in order to do classic RaDiO packet scheduling, the node would choose a policy π for the next transmission opportunity. The policy would indicate for every packet in the queue, $p_j \in \Phi_n$, whether this packet is transmitted $\pi(j) = 1$ or not $\pi(j) = 0$, so as to minimize a weighted function of distortion and rate $J(\pi) = D(\pi) + \lambda R(\pi)$.

With network coding, the node $n \in \mathbf{N}$ chooses some network codes, consisting of packets in the queue XOR-ed together, to transmit. All possible network codes at node n are $C_n = \{c_k^i\}_{k=1...2^{\psi_{t(p_i)}}}^{i=1,...\phi_n}$. The code transmission policy at node n consists of a vector Π_n that indicates for every possible code $c_u \in C_n,\ u = 1, ..|C_n|$, whether it is transmitted $\Pi_n(c_u) = 1$ or not $\Pi_n(c_u) = 0$, in the next transmission opportunity. To avoid transmitting two network codes $c_u, c_v \in C_n$ that have common packets from the set Φ_n, we restrict our attention to valid network code policies Π_n^{valid} that do not allow that to happen; *i.e.,* $\Pi_n^{valid} \subset \Pi_n$ s.t. $\Pi_n^{valid}(c_u) = 1 \bigwedge \Pi_n^{valid}(c_v) = 1$ if and only if $c_u \bigcap c_v = \emptyset$. Our goal is to find the optimal code transmission policy on all nodes $\Pi^{valid} = \{\Pi_n^{valid}\}_{\forall n \in \mathbf{N}}$, so as to minimize the total distortion $D(\Pi^{valid})$, subject to the rate constraint $R(\Pi^{valid}) \leq R_{av}$ where R_{av} is the available bit rate. With Lagrangian relaxation, our problem turns to finding the code transmission policy Π^{valid} so as to minimize $J(\Pi^{valid}) = D(\Pi^{valid}) + \lambda R(\Pi^{valid})$.

Instead of finding the optimal code transmission policy, we can map each code to the packets

[4]We note that our NC-RaDiO formulation assumes that the distortion of a flow is approximated by the sum of the distortion incurred at each hop along its path. This allows for the centralized RaDiO framework to be solved in a distributed way, *i.e.,* to make decision at each node as examined in [76].

it contains (*i.e.,* XOR-ed together), and find the optimal packet transmission policy. The reason for converting the problem from a code to a packet transmission policy selection is that it is more natural to express distortion values per packet. Let π be the packet transmission policy on all nodes, $\pi = \{\pi_n(j)\}_{\forall n \in \mathbf{N}, \forall p_j \in \Phi_n}$. π depends on the code transmission policy Π^{valid} as follows:

$$\pi_n(j) = \begin{cases} 1 & \text{if } \exists\ c_u \in C_n \text{ s.t. } p_j \in c_u \text{ and } \Pi_n^{valid}(c_u) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

An equivalent problem is to choose a packet policy π, s.t.:

$$\min_{\pi,\lambda} J(\pi) = \min_{\pi,\lambda} \{D(\pi) + \lambda R(\pi)\} \tag{3.9}$$

In Eq.(3.9), $D(\pi)$ is the total distortion over all nodes under policy π: $D(\pi) = \sum_{n=1}^{N} D(\pi_n)$. $D(\pi_n)$ is the approximate distortion of the flows transmitted from node n under the policy π_n. Following a similar definition as in [51], $D(\pi_n) = \sum_{p_j \in \Phi_n} \gamma(j)\Delta(j)P(\pi_n(j))$, where: $\gamma(j)$ is the priority/importance of the flow to which packet p_j belongs; $\Delta(j)$ is video quality distortion when packet p_j is lost as defined in section 3.2.1; and $P(\pi_n(j))$ is the probability that packet p_j is lost under policy $\pi_n(j)$. In particular, $P(\pi_n(j)) = P_p(j)P_c(\pi_n(j))$ consists of two parts: the probability $P_p(j)$ that the packet is lost in previous transmissions; and the probability $P_c(\pi_n(j))$ that the packet is lost in its current transmission under policy $\pi_n(j)$. Let also $t_d(j)$ be the deadline of packet p_j, t_c the current time, and t_m the time of m^{th} transmission assuming M transmissions so far. FTT' and RTT' are the random variables corresponding to the forward and round trip times, respectively[5]. Then the loss probability

[5] FTT' is distributed as in Eq. (3.4) and RTT' has CCDF $P\{RTT' > \tau\} = \varepsilon_R + (1 - \varepsilon_R)\int_{\tau}^{\infty} p_R(t)dt$, where $p_R(t)$ is the distribution of round trip time, and ε_R is the loss probability, considering the forward and backward channels together.

can be further expressed as follows:

$$P_p(j) = \prod_{m=1}^{M} P\{FTT' > t_d(j) - t_m | RTT' > t_c - t_m\} \tag{3.10}$$

$$P_c(\pi_n(j)) = \begin{cases} P\{FTT' > t_d(j) - t_c\} & \text{if } \pi_n(j) = 1, \\ 1 & \text{if } \pi_n(j) = 0 \end{cases} \tag{3.11}$$

In Eq. (3.9), $R(\pi)$ is the total rate function over all node rates under policy π: $R(\pi) = \sum_{n=1}^{N} R(\pi_n)$. $R(\pi_n)$ is the rate of the flows transmitted from node n under policy π_n: $R(\pi_n) = \sum_{\forall c_u \in C_n} \max_{p_j \in c_u}\{B(j)\rho(\pi_n(j))\}$, where $B(j)$ is the size of packet p_j in bytes, and $\rho(\pi_n(j))$ is the average cost of transmitting packet p_j. Note that the maximization $\max_{p_j \in c_u}\{B(j)\rho(\pi_n(j))\}$ term comes from network coding: before getting XOR-ed together, packets may need to be padded up to the length of the longest packet.

Note that $\sum_{\forall c_u \in C_n} \sum_{p_j \in c_u}$ and $\sum_{p_j \in \Phi_n}$ are equivalent. Also $\rho(\pi_n(j)) = \Pi_n^{valid}(c_u) = 0$ or 1 (s.t. $p_j \in c_u$) depending on whether code c_u is transmitted or not. With these observations and by replacing the distortion $D(\pi)$ and rate $R(\pi)$ terms with their detailed expressions discussed above, the NC-RaDiO problem of Eq.(3.9) can be re-written as follows:

$$\min_{\pi,\lambda} \sum_{n=1}^{N} \sum_{\forall c_u \in C_n} \Big(\sum_{p_j \in c_u} \gamma(j)\Delta(j)P(\pi_n(j)) + \lambda \Pi_n^{valid}(c_u) \max_{p_j \in c_u}\{B(j)\}\Big) \tag{3.12}$$

Optimal Solution

Since current systems typically transmit one packet (network code in our case) at each transmission opportunity, we will focus on this case from now on; *i.e.,* we will find the optimal network code (instead of finding several network codes) so as to minimize the above rate-distortion function. This was also the case in [10, 11] as well as in our NCV and NCVD algorithms. An approach, introduced in [50], was to increase the Lagrange multiplier λ so that exactly one network code is selected as the optimal code for the total rate-distortion function. However, this approach requires centralized knowledge.

A distributed approach is to solve the problem in Eq. (3.12) for every network code $c_u \in C_n, n = 1, ..., N$, find a threshold value, $\lambda_n(c_u)$, used to make the decision whether to transmit $c_u \in C_n$ and select the network code that maximizes that threshold $\max_{n,c_u}\{\lambda_n(c_u)\}$. In particular, let us define the per network code cost function as:

$$J_n(c_u) = \sum_{p_j \in c_u} \gamma(j)\Delta(j)P_p(j)P_c(\pi_n(j)) + \lambda \Pi_n^{valid}(c_u) \max_{p_j \in c_u}\{B(j)\} \tag{3.13}$$

for every code $c_u \in C_n$ and node $n = 1, ..., N$. When we decide to not transmit this network code, *i.e.,* $\Pi_n^{valid}(c_u) = 0$, the cost becomes $J_n^0(c_u) = \sum_{p_j \in c_u} \gamma(j)\Delta(j)P_p(j)$. When we decide to transmit the network code, *i.e.,* $\Pi_n^{valid}(c_u) = 1$, the cost is $J_n^1(c_u) = \sum_{p_j \in c_u} \gamma(j)\Delta(j)P_p(j)P_c(\pi_n(j)) + \lambda \max_{p_j \in c_u}\{B(j)\}$. Depending on which of the two costs is smaller, we decide whether to transmit c_u or not. The maximum λ that satisfies the inequality $J_n^1(c_u) \leq J_n^0(c_u)$ is

$$\lambda_n(c_u) = \frac{\sum_{p_j \in c_u} \gamma(j)\Delta(j)P_p(j)(1 - P_c(\pi_n(j))}{\max_{p_j \in c_u}\{B(j)\}} \tag{3.14}$$

The optimal policy decides, in a rate-distortion optimized manner, which node n should transmit and what code c_u should be transmitted, by choosing the maximum Lagrange multiplier: $\max_{\{n,c_u\}}\{\lambda_n(c_u)\}$. This can be achieved in practice in two rounds: first, we compare $\lambda_n(c_u)$ in the same node n and we can find $\lambda_n = \max_{\{c_u\}}\{\lambda_n(c_u)\}$ for this node; then all nodes $n \in \mathbf{N}$ need to exchange their λ_n values with all the neighbors; finally, the node with $\lambda = \max_{\{n\}}\{\lambda_n\}$ is the one transmitting. This is repeated at each transmission opportunity.

Relation to NCV and NCVD

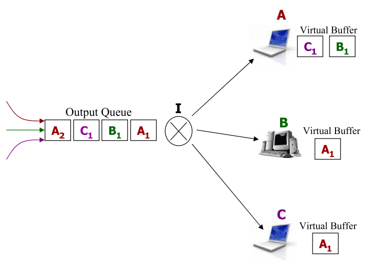
The NC-RaDiO framework includes NCV and NCVD as special cases for a system that makes the following implementation choices (consistently with COPE [10, 11] and the system discussed here):

- All packets have the same size (possibly using padding): $B(j) = B, \forall p_j \in \Phi_n, n = 1, ..., N$.
- A deterministic rule is used to decide whether a previously transmitted packet is lost or not: if average RTT time (RTT_{avg}) has passed since its last transmission and no ACK has been received, the packet is considered lost; otherwise, it is considered successfully received. Let $\tau(j)$ be the time duration since the most recent transmission of packet p_j, if p_j has been transmitted before; or $\tau(j) = \infty$ if p_j has not been transmitted before. Then, we can re-write the probability of loss in previous transmissions $P_p(j)$, in the NC-RaDiO formulation, as follows:

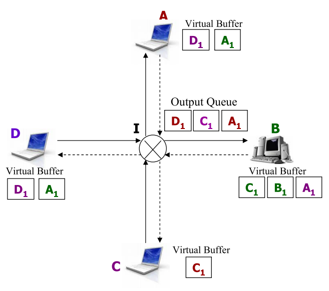
$$P_p(j) = \begin{cases} 1 & \text{if } \tau(j) \geq RTT_{avg} \\ 0 & \text{otherwise} \end{cases}$$

Furthermore, we note the correspondence between the NCV/NCVD algorithms and the NC-RaDiO formulations.

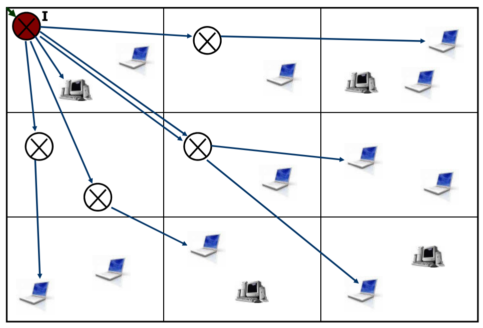
- Let $d(\pi_n(j))$ be the indicator function, which equals 1 if packet p_j is decodable at its next hop node when transmitted with network code $c_u \in C_n$; or 0 otherwise. This corresponds to the decodability indicator function $d_l^k(n_\eta)$ in NCV/NCVD.
- The probability of loss in the current transmission, $P_c(\pi_n(j))$, in the NC-RaDiO for-



(a) One-hop Downlink Topology



(b) Two-hop Cross Topology



(c) Multi-hop Grid Topology

Figure 3.4: Topologies and traffic scenarios used in simulations. (We also vary the number of nodes in each topology.)

mulation, can be re-written as follows:

$$P_c(\pi_n(j)) = \begin{cases} P\{FTT' > t_d(j) - t_c\} & \text{if } d(\pi_n(j)) = 1 \\ 1 & \text{if } d(\pi_n(j)) = 0 \end{cases}$$

Then $1 - P_c(\pi_n(j)) = (1 - P\{FTT' > t_d(j) - t_c\})d(\pi_n(j))$. Further considering Eq. (3.3), it turns out that $1 - P_c(\pi_n(j)) = (1 - P(j))d(\pi_n(j))$.

Under the above assumptions and notations, the Lagrange multipliers in the NC-RaDiO formulation can be re-written:

$$\lambda'_n(c_u) = \sum_{p_j \in c_u s.t. \tau(j) \geq RTT_{avg}} \gamma(j)\Delta(j)(1 - P(j))d(\pi_n(j)) \quad (3.15)$$

Note that the Lagrange multiplier $\lambda'_n(c_u)$ in Eq. (3.15) is equivalent to the improvement value I_k^i in Eq. (3.5).[6] NCV and NCVD are suboptimal solutions to the NC-RaDiO optimization problem due to: (i) the aforementioned assumptions, *i.e.* not taking into account the exact packet size or the effect of previous transmissions and (ii) the fact that nodes in a practical low-complexity system (such as COPE or NCV/NCVD) take local decisions and do not exchange the improvement values (Lagrange multipliers) to decide which node should transmit. However, the equivalence of Eq. (3.15) and Eq. (3.5) is the intuition why NCV and especially NCVD are efficient heuristics to the NC-RaDiO problem. Next, we confirm via simulation that the performance of our algorithms is near the optimal NC-RaDiO in a wide range of scenarios.

3.3 Performance Evaluation

In this section, we evaluate the performance of the proposed schemes; NCV, NCVD, and NC-RaDiO in terms of video quality and network throughput in a wide range of scenarios. We compare them to four baseline schemes: no network coding; noNC, multimedia streaming; MM, network coding optimized for throughput; NCT as in [10], and an improved version of it; NCTD. Simulation results show that (i) NCV, NCVD, and NC-RaDiO can significantly improve video quality and application-level throughput, without compromising MAC-level

[6]In Eq. (3.15), the $g_l^k(n_\eta)$ term does not exist, because its value is naturally 1 since we only consider the improvements of packets at their next hop nodes, instead of considering the possible improvement at all nodes in the neighborhood as in Eq. (3.5). However, this difference is only a matter of notation and it clear that Eq. (3.15) and Eq. (3.5) are equivalent.

throughput and (ii) NCVD is an efficient heuristic solution to NC-RaDiO. In 3.3.1 we describe the simulation setup, in 3.3.2 we present the simulation results, and in 3.3.3 we discuss complexity issues.

3.3.1 Simulation Setup

We used the GloMoSim simulation environment [77] to implement the proposed algorithms and the baseline schemes. Below, we describe the simulation setup, which includes: the topologies and traffic scenarios considered, the MAC model, the wireless channel model, the video sequences, and the baseline algorithms used for comparison.

Simulation Topologies

The topology and traffic scenario can strongly affect the gain from using network coding. We considered three practical scenarios shown in Fig. 3.4.

Single-Hop Downlink Topology: In this topology, we consider the single-hop downlink scenario shown in Fig. 3.4(a). The intermediate node I receives different video streams, which it forwards downstream towards their destinations. I can apply different schemes for network coding and packet scheduling. We assume that receivers are placed on a circle with radius $90m$ and the intermediate node I which is placed in the center of the circle. Receivers are the only ones using the downlink, hence there is no congestion. However, packets may still be lost due to errors on the wireless channel, and can also experience a random MAC propagation delay, 2ms on average. The one-way delay budget for this single-hop is set to $100ms$. We also performed simulations for different delay budgets ($50-200ms$), for different number of nodes ($N : 3-11$) including the intermediate node and the receivers, and for different channel conditions.

Cross Topology: In this topology, we consider multiple crossing flows at an intermediate node as shown in Fig. 3.4(b): pairs of nodes A, C and B, D communicate over an intermediate node I, *e.g.*, A transmits to C and C transmits to A via I. A single channel is used for both uplink and downlink transmissions. MAC scheme will be explained later in the section. In this scenario, each node buffers a packet it has just transmitted as well as all overheard packets; an illustrative example is shown in Fig. 3.4(b). The intermediate node I makes decisions on network coding and scheduling. We assume again that nodes are placed on a circle with center I and radius 90m. The remaining settings are similar to the single-hop downlink.

Grid Topology: In this topology, we consider the wireless mesh network (WMN) shown in Fig. 3.4(c). Nodes are distributed over a $300m \times 300m$ terrain according to a grid topology: the area is divided into 9 cells of equal size, 20 nodes are divided into 2 or 3 node sets randomly, and each set is assigned to a different cell. Nodes in a set are randomly placed within their assigned grid. The WMN is connected to the Internet via a high speed lossless link through a gateway I, placed in the upper leftmost grid shown in Fig. 3.4(c). Each node receives a video stream from the Internet going through the gateway. Depending on the location of the receiving node R, the stream is either transmitted directly (one-hop) or routed (two hops). If both I and R are either in the same cell or in neighboring cells, there will be a one-hop transmission; otherwise a node in the cell between I and R is selected as an intermediate hop, indicated by $\otimes$ on Fig. 3.4(c); if there are more than one neighboring cells, one is selected randomly. The remaining settings are similar to the single-hop downlink.

MAC Model

IEEE 802.11 is used in the MAC layer, with the following modifications needed for network coding. First, to obtain the network coding benefit we need a broadcast medium, which is hidden by the 802.11 protocol. Similarly to [10] we used the pseudo-broadcasting mechanism:

packets are XORed in a single unicast packet, an XOR header is added for all nodes that should receive that packet, and the MAC address is set to the address of one of the receivers. A receiver knows whether a packet is destined to it from the MAC address or the XOR header. Second, 802.11 waits for an ACK after a packet is transmitted, which reduces the network coding opportunities and increases the overhead. Instead, we consider that packets are transmitted one after the other without waiting for an ACK and the active-inactive mechanism is used to reduce unnecessary re-transmissions. For the NC-RaDiO scheme in particular, we assume that the Lagrange multipliers are exchanged through a separate channel.

Wireless Channel Model

We consider the two-ray path loss model and Rayleigh fading channel model implemented in GloMoSim. The two-ray path loss model is a propagation path loss model using free space path loss for near sight and plane earth path loss for far sight. For the Rayleigh fading model, we consider average channel SNR $\{3, 5, 7, 9\}$ dB.

Video Sequences

As our test sequences, we used standard sequences: *Carphone*, *Foreman*, *Mother & Daughter*, *Claire*, *Coastguard*, *News*, *Grandma*, and *Salesman*. These were QCIF sequences encoded using the JM 8.6 version of the H.264/AVC codec [78], [79]. The group of pictures consisted of one I and nine P frames. All encoded sequences had data rate $70kbps$ and frame rate $30fps$. Each frame consists of at least one slice. Each slice was packetized into an independent NAL (network abstraction layer) unit of size $250B$. NAL units are encapsulated using the Real-time Transport Protocol (RTP) and User Datagram Protocol (UDP).

As metric for the video quality of an encoded sequence, we use the average PSNR, *i.e.,* the peak-signal-to-noise ratio based on the luminance (Y) component of video sequences, measured in dB, and averaged over the entire duration of the video sequence. The PSNR of the encoded sequences *Carphone, Foreman* and *Mother & Daughter*, before any transmission, was 29.95dB, 28.70dB and 40.74dB respectively; these PSNR values are denoted as No Error in Table 3.2. We repeated and concatenated the standard sequences to create longer test sequences of duration 30sec each. At the receiver side, basic copy-concealment scheme is used when an entire frame is lost.

Baseline Algorithms for Comparison

We compare our algorithms, NCV, NCVD, and NC-RaDiO, against four baselines for packet scheduling: no Network Coding (noNC), Multimedia Streaming Algorithm (MM), Network Coding for Throughput (NCT), and its improved version (NCTD).

Fig. 3.5 summarizes all algorithms and classifies them, in increased sophistication, across two dimensions: packet scheduling and network coding. noNC takes no action in either dimension - it is a simple FIFO. NCT and NCTD do network coding and combine several packets in one transmission so as to maximize throughput; NCT optimizes only the side packet selection while NCTD optimizes both primary and side packet selection. Both NCT and NCTD are agnostic to the content of the packets. In contrast, MM does not use any network coding but prioritizes packet transmission, considering packet distortion and deadlines. The proposed algorithms, NCV and NCVD, combine both ideas and NC-RaDiO further extends these ideas by prioritizing nodes. NCVD can be thought as a combination of network coding (NCTD) and content awareness (MM). NCV can be thought as a combination of NCT and MM, but unlike MM, it is restricted in its choice of primary packet. For a fair comparison, the active-inactive mechanism described in section 3.1, is employed in all algorithms except for NC-RaDiO (which relies on the success probabilities of previous transmissions).

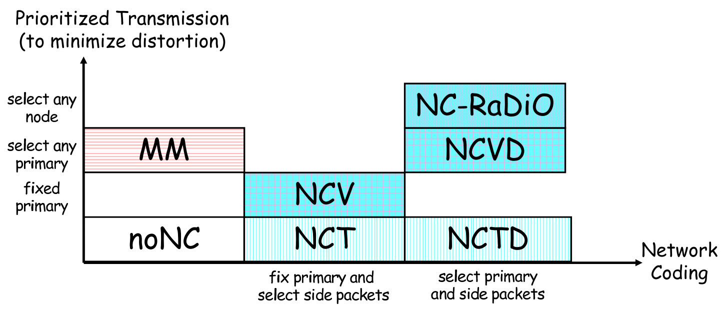

Prioritized Transmission (to minimize distortion)
select any node
select any primary
fixed primary
MM
noNC
NCV
NCT
NC-RaDiO
NCVD
NCTD
fix primary and select side packets
select primary and side packets
Network Coding

Figure 3.5: Summary of Algorithms under Comparison. NCV, NCVD, and NC-RaDiO are the proposed algorithms that combine packet scheduling and network coding. The rest are baselines that use at most one of the mechanisms: NoNC is simply FIFO, MM uses optimal packet selection to minimize distortion, NCT and NCTD use network coding to maximize throughput.

No Network Coding (noNC): This is a FIFO Tx queue without network coding. Consider again Example 4 and Fig. 3.2: node I stores packets for all three streams destined to nodes A, B, C. In every time slot, I transmits the first packet from the head of the queue.

Multimedia Streaming Algorithm (MM): This is a scheduling scheme that optimally chooses the packet to be transmitted without network coding. We consider it in order to see how much benefit comes from prioritized packet transmission alone, apart from network coding. MM is essentially a reduced version of NCVD with network codes having one packet at most (no network coding), *i.e.*, for primary packet p_i, and the only eligible network code being $c_0^i = p_i$. The improvement at node n_η is $I_0^i(n_\eta) = (1 - P(i))\Delta(i)g_i^0(n_\eta)$, where $P(i)$ is the loss probability of packet p_i given in Eq. (3.3), $\Delta(i)$ is the improvement of video quality if packet p_i is received correctly and on time at client n_η, and $g_i^0(n_\eta)$ is the indicator function that shows whether packet p_i is destined to node n_η. Among all packets in the Tx queue, MM selects the p_i that maximizes the total video quality, considering the improvement to all nodes in its neighborhood:

$$p_i = argmax \sum_{\eta=1}^{N} I_0^i(n_\eta) \tag{3.16}$$

In other words, MM transmits the most important packet in the Tx queue, considering per packet distortion and loss probability across all streams.

Network Coding for Throughput (NCT): This is an improved version of the algorithm proposed in [10]. The packet transmission mechanism is the same as in the *noNC* scheme, but network coding is used to maximize throughput, as follows. The first active packet in the Tx queue is selected as primary; side packets are chosen to be XOR-ed together with the primary packet so as to construct a network code that is decodable by the maximum number of receivers possible.

There are two improvements in NCT compared to the coding algorithm in [10] that allow NCT to achieve even higher throughput than [10]. First, NCT follows the same ACK and retransmission mechanism described in section 3.1: packets with pending acknowledgments are marked as inactive for one RTT, while the channel is used to transmit other packets as primary. In [10] and in general MAC retransmissions, a packet stays at the head of the queue blocking other packets, until it goes through successfully or it exceeds the maximum number of retransmissions. Another difference is that NCT uses an improved version of the coding procedure in [10]: NCT considers all possible subsets of the candidate side packets thus maximizing the number of receivers that can decode; while [10] considers only the earliest packet from each stream as candidate side packets, thus sacrificing some throughput for reduced complexity and for maintaining the packet ordering. Therefore, we use NCT as our baseline for the maximum achievable throughput per transmission using network coding.

NCT & looking into the queue in Depth (NCTD): NCT selects as primary packet the first active packet in the Tx queue. Similarly to NCV, this limits the candidate codes to those that are decodable only for this single primary packet. Similarly to NCVD, we extend NCT to NCTD, which looks into the entire Tx queue and considers all packets as candidate for the primary packet. Thus, NCTD optimizes throughput by primary packet selection and network code construction.

3.3.2 Simulation Results

In this section, we present simulation results that compare the proposed to baseline algorithms and demonstrate that the former can improve video quality and application-level throughput, without compromising MAC-level throughput.

Summary of Results

Some general observations across all simulation scenarios are summarized below:

The best and worst algorithms: The optimal solution to NC-RaDiO is clearly the best in terms of PSNR and close to the best in terms of throughput. NCVD closely approximates the optimal solution of NC-RaDiO in terms of both PSNR and throughput. As expected, noNC is consistently the worst algorithm in all aspects.

Media awareness added on network coding: The proposed media-aware network coding algorithms, NC-RaDiO, NCVD, and NCV consistently outperform the corresponding network coding-only algorithms, NCTD and NCT, in terms of PSNR, while achieving similar throughput.

Media awareness vs. network coding: The MM algorithm achieves higher PSNR than the weaker (i.e. other than NCVD) network coding algorithms (NCT, NCTD, NCV) in harsh channel conditions while the network coding algorithms (NCT, NCTD, NCV) perform better in mild channel conditions. In other words, in the former case the quality/importance of the transmitted packets matters, while in the latter case the quantity has a greater effect. Note that NC-RaDiO and NCVD achieves the highest PSNR in all conditions.

Throughput: In addition to achieving the highest PSNR, NCV, NCVD, and NC-RaDiO achieve higher application throughput, since they consider playout deadlines. Furthermore, all network coding schemes (NCT/NCTD, NCV/NCVD/NC-RaDiO) achieve similar MAC throughput, much higher than the non-network coding schemes (noNC, MM).

Node selection: NC-RaDiO performs slightly better than NCVD, in terms of PSNR, thanks to (i) explicit consideration of previous transmission probabilities and packet sizes and (ii) exchanging Lagrange multipliers among nodes to select a node to transmit. However, the performance improvement of NC-RaDiO over NCVD is negligible in a wide range of scenarios, indicating that NCVD is an efficient heuristic for the NC-RaDiO optimization problem.

Primary packet selection: NC-RaDiO and NCVD achieve higher PSNR that NCV, thanks to their node selection (NC-RaDiO) and primary packet optimization (NC-RaDiO and NCVD). MM outperforms NCV (but never NC-RaDiO and NCVD) in some scenarios, for the same reason, *i.e.,* because NCV can only choose the side but not the primary packet. The optimization of primary packet selection is more important for the media-aware than for the network coding schemes: the similar performance of NCT and NCTD indicates that the primary packet optimization does not significantly increase the number of packets in a network code and thus the throughput.

Comparison of topologies: The one-hop downlink is a building block for other topologies and can be used as a baseline for comparison; it does not incur any delay in accessing the channel and all flows are network-coded. Furthermore, this is the only topology in which there is no need to exchange Lagrange multipliers among nodes. In the cross topology, there is more delay due to the two-hop transmission and in accessing the uplink channel; therefore

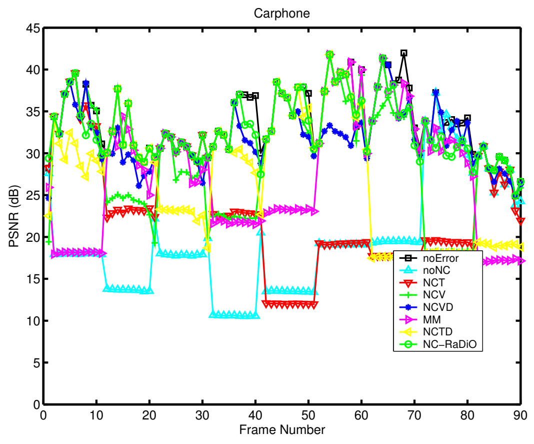


Figure 3.6: PSNR per frame for part of the Carphone sequence and for an example channel realization. One-hop downlink scenario with $N = 4$ (I and three receivers A, B, C), channel SNR = 5dB, delay budget 100ms, and data rate 500kbps. Seven schemes (noError, noNC, MM, NCT, NCTD, NCV, NVCD, NC-RaDiO) are compared. (The average PSNR values over the entire sequence are summarized in Table 3.2.)

there are less network coding opportunities and the network coding schemes perform worse. Similar arguments apply to the grid topology. We now discuss each simulation scenario in detail.

Single-Hop Downlink Topology

We consider the single-hop downlink topology of Fig. 3.4(a), when node I streams sequences *Foreman*, *Mother & Daughter*, and *Carphone* to clients A, B, C, respectively. When the number of nodes (N) is greater than four the video sequences *Coastguard*, *Salesman*, *News*, *Grandma*, and *Claire* are used sequentially. First we focus on the scenario with three receivers in the system and evaluate the performance of the algorithms for different delay budgets and channel SNR levels.

Video Quality Improvements: Fig. 3.6 shows the video quality experienced by one client (PSNR over frame number for parts of the Carphone sequence) for the seven algorithms under comparison, namely noNC, MM, NCT, NCTD, NCV, NCVD, and NC-RaDiO as well

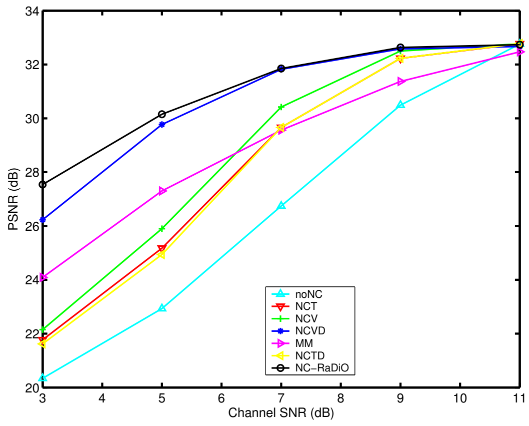
Table 3.2: Average PSNR for the scenario of Fig.3.6 (video: at 70$kbps$ and 100ms playout deadline; channel with $SNR = 5dB$ and 500$kbps$ data rate)

avg PSNR (dB)	Carphone	Foreman	Mother&Daughter
No Error	29.95	28.70	40.74
NC-RaDiO	28.46	27.51	35.08
NCVD	27.98	26.87	35.36
NCV	25.40	25.14	28.66
NCTD	24.91	24.60	28.61
NCT	23.95	24.38	27.19
MM	25.17	24.61	32.12
noNC	22.32	22.64	23.84

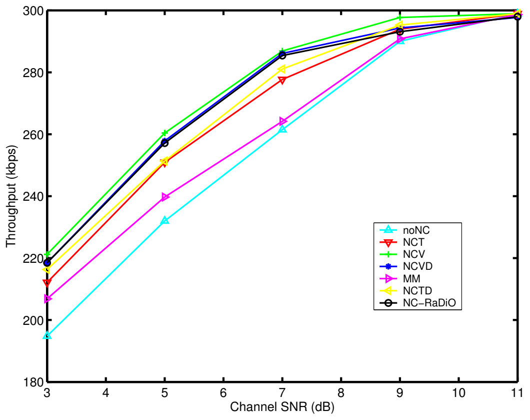
as for the encoded sequences before transmission (noError). The simulation is performed for channel SNR 5dB with 100ms delay budget and 500$kbps$ channel data rate; for comparison, the same wireless channel trace is used as input to all six algorithms. As expected, there are time periods, during which the channel is bad, the quality degrades for all algorithms. However, the degradation for NC-RaDiO, NCVD, and NCV is much less than for NCTD, NCT, and noNC, because NC-RaDiO, NCVD, and NCV select network codes to protect and deliver the most important packets on time, thus improving the video quality; in contrast, NCTD, NCT, and noNC treat all packets similarly. The degradation of MM is less than NCT, NCTD, and noNC for most of the region even when the channel goes bad, because it transmits more important packets and is comparable to or worse than NCV, and worse than NCVD and NC-RaDiO, because NCV and NCVD transmit more packets using network coding as well as considering the importance of some (NCV) or all (NCVD, NC-RaDiO) packets.

The average PSNR for each sequence and algorithm is summarized in Table 3.2. We see that, as expected, the noNC scheme performs poorly. NCT improves over noNC because it delivers more packets per time slot. NCV improves over NCT because it chooses important video packets as side packets, NCTD improves over NCT, because it also optimizes the primary packet selection. MM outperforms noNC, NCT, and NCTD. This result is quite interesting, because NCT and NCTD transmit more information than MM; however, not only the amount but also the content of information transmitted is important. MM exhibits similar or better performance than NCV. This is intuitive, because NCV optimizes side packets for video quality improvement but not primary packets; since primary packets are the main packets that are transmitted to all receivers, optimized packet scheduling is performed only in a few of the transmitted packets. NCVD achieves higher video quality compared to NCV, NCTD, NCT, MM, and noNC, since both primary and side packet selection is optimized. NC-RaDiO is slightly better than NCVD thanks to considering different packet sizes and to the exact calculation of previous packet transmission probability. However, since packet sizes are almost the same in different packets, and the deterministic decision is a good estimator of probabilistic decision, the improvement of NC-RaDiO over NCVD is very small. Actually, both NC-RaDiO and NCVD achieve a PSNR close to that of the original encoded sequence (noError), even for harsh channel conditions (*e.g.*, $5dB$ channel SNR).

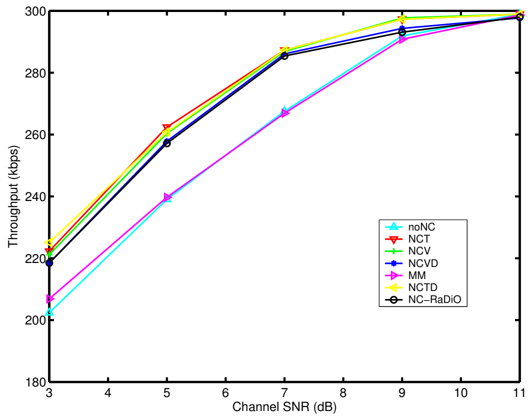
The same scenario as in Fig. 3.6 is considered, but with channel SNR varying in a range from $3dB$ to $11dB$. Fig. 3.7(a) shows the average PSNR achieved by each algorithm. Clearly, NC-RaDiO, NCVD and NCV outperform NCT (by $1-4$dB) and noNC (by $3-5$ dB) for all channel SNRs. NCT and NCTD exhibit similar performances. NC-RaDiO and NCVD always outperforms to MM (by 3dB). MM outperforms NCT, NCV and NCTD for low channel SNR levels, but does not for higher channel SNR levels. When the channel is good ($11dB$ channel SNR), all algorithms transmits almost all of their packets. For a worse channel (SNR $9dB$), the network coding algorithms NC-RaDiO, NCVD, NCV, NCTD, NCT are better than the no network coding schemes (noNC and MM), because they transmit more packets. When the channel is really bad ($5dB$ channel) NC-RaDiO, NCVD and MM outperform NCTD, NCT, NCV because some packets have to be dropped under these harsh conditions, even if network coding is used, and the quality of selected packets has a dominant effect. Actually, as we will show later, MM transmits less packet than NCTD, NCT and NCV but since it optimizes packet scheduling, the video quality remains high even for bad channels. NC-RaDiO and



(a) PSNR



(b) Application-level throughput



(c) MAC-level throughput

Figure 3.7: One-hop downlink topology with three receivers (and one intermediate node I, i.e. $N = 4$). Performance for different channel SNR levels in terms of: (a) PSNR (averaged across each sequence and across all three sequences) (b) total application-level throughput (added over all three streams) (c) total MAC-level throughput. (The delay budget is $100ms$ and the data rate is $500kbps$.)

NCVD outperforms MM for all channel SNR levels, because they have the advantages of both network coding and multimedia streaming: they transmit more packets using network coding and do packet scheduling considering packet importance and deadlines. NC-RaDiO improves slightly over NCVD, especially for harsh channel conditions.

In the scenarios discussed so far, we have considered a delay budget of $100ms$. In Fig. 3.8, we show the average PSNR for a delay constraint ranging from 50 to $200ms$. NC-RaDiO, NCVD, and NCV improve video quality for the entire range of delay values as compared to

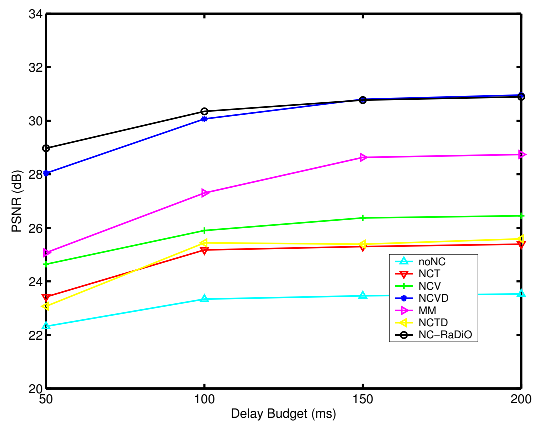


Figure 3.8: PSNR values (averaged over each sequence and across sequences) for different delay budgets in downlink topology, $N = 4$. Channel SNR is $5dB$ and data rate is $500kbps$.

NCTD, NCT and noNC. MM is better than NCV since NCV lacks primary packet scheduling. The network coding algorithms bring less improvement for a tight delay budget, which limits the number of retransmissions and the lifetime of packets both at the Tx queue and in the virtual buffers, thus decreasing network coding and selection opportunities. However, even with tight delay constraints, there is significant video quality improvement from NC-RaDiO and NCVD compared to all other algorithms.

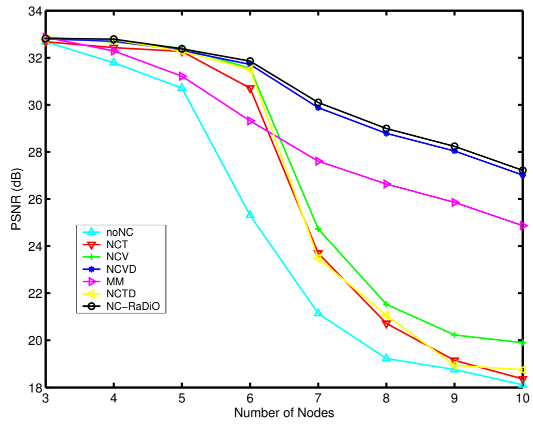
Throughput Improvements: The video-aware schemes improve video quality because they explicitly take it into account in the code selection. In this section, we show that, our schemes also significantly improve application-level throughput while maintain the same levels of MAC-level throughput. In other words, our algorithms deliver the same amount of packets but choose to deliver more useful video packets.

Application Throughput. Fig. 3.7(b) shows the total throughput as seen by the application-layer (*i.e.,* NAL units per sec) added over all clients. The figure clearly shows that NC-RaDiO, NCVD, and NCV achieve higher throughput as compared to NCT, NCTD, noNC and MM. The main reason is that NC-RaDiO, NCVD, and NCV do not select codes consisting of packets whose deadlines are within one transmission time, while NCT and NCTD transmit all packets. Late packets do not contribute to application-level throughput, because those packets are discarded at the client even if they are received successfully. All the network coding schemes (NC-RaDiO, NCVD, NCV, NCTD and NCT) are better than non-network coding schemes (MM and noNC) because they transmit more packets. As expected, MM is better than noNC, because it considers packet deadlines for packet scheduling.

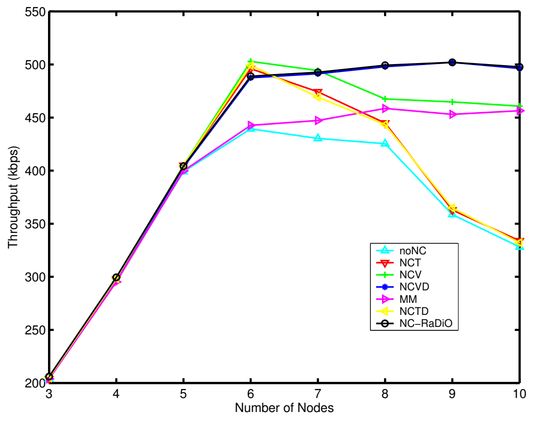
MAC Throughput. For completeness, we also show the MAC-layer throughput in Fig. 3.7(c). As expected, all network coding schemes (NC-RaDiO, NCVD, NCV, NCTD, NCT) achieve higher MAC-level throughput than noNC and MM, because they convey more information content per transmission. Interestingly, all network coding schemes achieve similar throughput, although NCT/NCTD are the ones explicitly designed to maximize throughput.

Next, we consider the performance of the proposed algorithms, when varying the number of nodes (N), for fixed delay budget $100ms$, data rate $1Mbps$, and channel SNR $5dB$. Fig. 3.9(a) shows the average PSNR (averaged over three video sequences; *Foreman, Carphone, Mother & Daughter* when $N = 4$ or higher; or averaged over two video sequences, namely *Carphone* and *Mother & Daughter*, when $N = 3$ in the system. Fig. 3.9(b) and (c) are the application and MAC level throughput seen at all receivers, respectively.

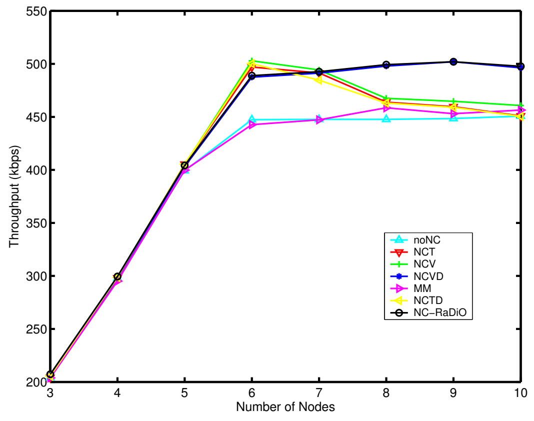
Fig. 3.9(a) shows that PSNR values of all algorithms are almost the same when $N = 3$. The reason is that the data rate is sufficiently large ($1Mbps$) to transmit and re-transmit almost all packets. When N increases, the PSNR of all algorithms decreases. When $N = 5$, NC-RaDiO, NCVD, NCV, NCTD, and NCT have almost the same PSNR while noNC and MM start deteriorating. The reason is that the network coding algorithms transmit effectively more packets than the non-network coding algorithms (noNC and MM). For $N = 6$, the network coding algorithms are still better than noNC and MM; we also note that NC-RaDiO, NCVD, NCV, and NCTD are better than NCT because NC-RaDiO, NCVD and NCTD transmit more packets due to primary packet optimization, and NCV transmits more important packets. When N increases further, the PSNR performance becomes more



(a) PSNR



(b) Application-Level Throughput



(c) MAC-Level Throughput

Figure 3.9: One-hop downlink topology for different number of nodes in the system. Performance of all algorithms in terms of (a) video quality (PSNR) (b) application-level and (c) MAC-level throughput. (Channel SNR is $5dB$, delay budget is $100ms$, and channel data rate is $1Mbps$).

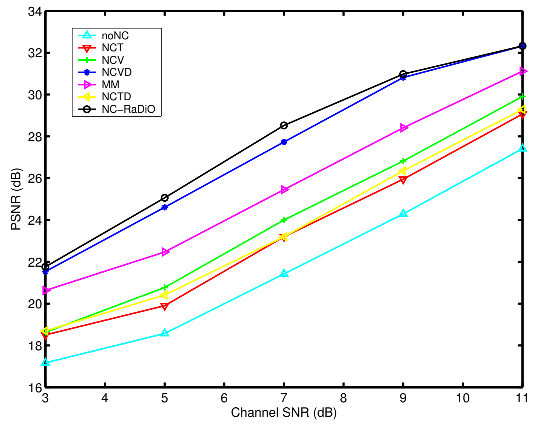
interesting: noNC is the clearly the worst; NCT and NCTD are similar to each other and better than noNC; NCV is better than NCT and NCTD, because it optimizes side packet selection, hence transmits more important packets. The most interesting part is that while the PSNR of noNC, NCT, NCTD, and NCV decreases sharply, the decrease in MM's PSNR is roughly linear. The reason is that MM utilizes limited resources to transmit important packets. On the other hand, NCT and NCTD combines packets to transmit effectively more packets; however, since they do not consider the deadlines of the packets, they transmit obsolete packets. NCV is also worse than MM, because it does not optimize the primary packets for video quality. However, when the resources get scarce (larger N), the optimal selection of each packet becomes more important than the amount of data transmitted. NC-RaDiO and NCVD outperform all algorithms by $2dB$-$5dB$.

The application- and MAC-level throughput are shown in Fig. 3.9(b) and Fig. 3.9(c). For up to $N = 5$, all algorithms deliver the same amount of packets. For $N = 5, ...8$, the network coding algorithms deliver more data (both application and MAC level) as compared to noNC and MM. For $N > 8$, the MAC throughput of NCV, NCT, and NCTD decreases, for two reasons: (i) having more streams sharing the same Tx queue decreases the lifetime of packets, hence the network coding opportunities (ii) NCT and NCTD transmit obsolete packets. NCV and MM have similar application throughput, since there are less network coding opportunities for NCV with increasing N. NC-RaDiO and NCVD achieve the highest application and MAC level throughput, because they create more network coding opportunities.

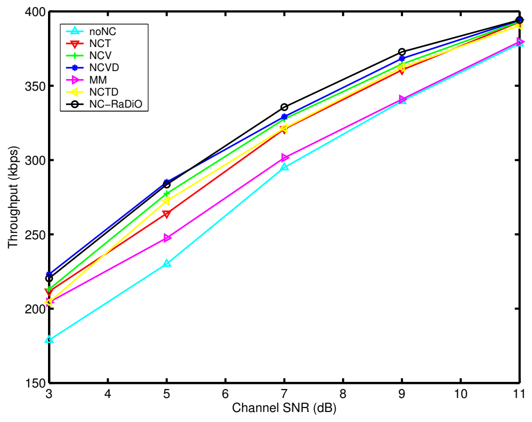
Cross Topology

We consider the cross topology shown in Fig. 3.4(b) when A, C transmit *Foreman* and *Mother & Daughter* to each other and B, C transmit *Carphone* and *Coastguard* to each other over the intermediate node I. When the number of nodes (N) gets larger *Salesman*, *News*, and *Grandma*, *Claire* pairs are used sequentially. First, we focus on the scenario with four nodes actively transmitting and receiving video ($N = 5$ including I) and evaluate the performance of the algorithms for different delay budgets and channel SNR. We fix the delay budget to $100ms$, the data rate to $1.3Mbps$ and vary the channel SNR from 3 to $5dB$.

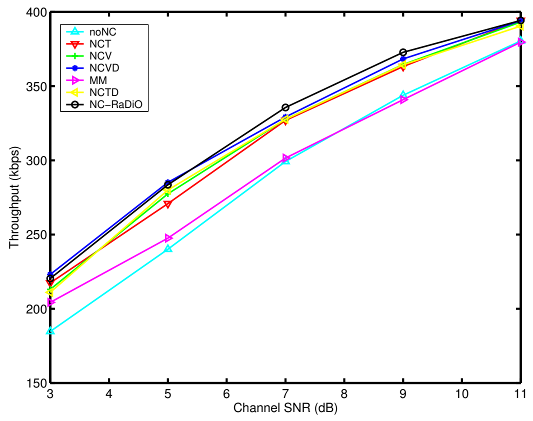
Fig. 3.10 shows the performance of all algorithms in this topology and for channel SNR in the range 3-$11dB$. Fig. 3.10(a) shows the PSNR: The ranking of the algorithms in decreasing PSNR is similar to the downlink scenario shown in Fig. 3.7(a). However, there are some



(a) PSNR



(b) Application Level Throughput



(c) MAC Level Throughput

Figure 3.10: Cross topology, with $N = 5$ nodes including I, for different channel SNR levels. Performance in terms of (a) PSNR (b) application throughput and (c) MAC throughput. (Delay budget is $100ms$, channel SNR is $5dB$, and data rate is $1.3Mbps$.)

differences. First, the performance gap between NC-RaDiO and NCVD is larger in the cross topology, because the exchange of Lagrange multipliers among nodes, to decide which node should transmit, becomes more important since all nodes transmit. NCVD is again very close to the optimal NC-RaDiO, which confirms that it is a good heuristic. Second, in the downlink scenario, all algorithms have the same PSNR at channel SNR $11dB$, while this is not the case in the cross topology. Even with a good channel ($11dB$), there are still packet lost in the channel. In the cross topology, more packets are lost, on the uplink and the downlink channels. Third, in the downlink topology, the network coding algorithms improve PSNR more than MM, for channel SNR greater than $7dB$. However, in the cross topology

MM's improvement is higher than that of NCV, NCT, and NCTD for all channel SNR levels since MM optimizes packet transmission in both uplink and downlink.

Fig. 3.10(b) and Fig. 3.10(c) depict the application and MAC throughput, respectively, for the same setting as in Fig. 3.10(a). noNC and MM have the same MAC throughput while all network coding algorithms have similar MAC throughput. MM achieves slightly higher application-level throughput than noNC. NC-RaDiO, NCVD, NCV, NCTD, and NCT achieve decreasing order of application level throughput. As compared to the downlink scenario and the throughput values shown in Fig. 3.7, the throughput difference between network coding and no network coding algorithms is less in the cross topology compared to other topologies; the reason is that there are more independent flows and thus less network coded packets.

Fig. 3.11 shows the PSNR achieved by all algorithms when we vary the number of nodes in the cross topology. We fix the channel SNR to $5dB$, the delay budget to $100ms$, and the data rate to $2Mbps$. NC-RaDiO and NCVD perform best, MM is second for the interesting part of the N range; NCV is better than NCT and NCTD which exhibits similar performance; and noNC is the worst as expected. If we compare this graph with the corresponding graph shown in Fig. 3.9(a), we see again that the benefit of network coding is less in the cross topology. One reason is that there are more independent than network coded flows in the system. Another reason is the increased delay in the two-hop transmission, while the deadline remains the same. However, NC-RaDiO and NCVD still improve over all other algorithms by $2.5 - 5$dB.

Grid Topology

We consider the grid topology shown in Fig. 3.4(c). I receives sequences over a high-speed error-free link and transmits a different sequence to each receiver over the grid topology,

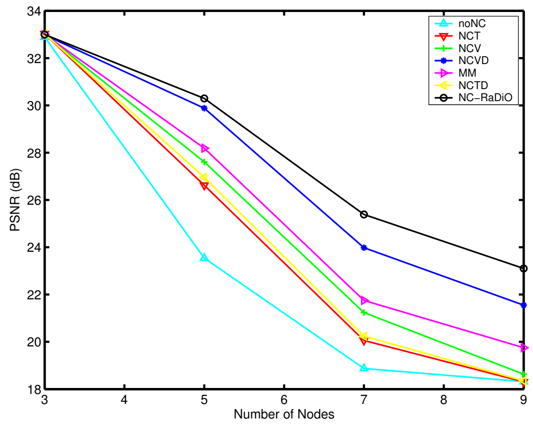


Figure 3.11: Cross Topology. PSNR performance for a different number of nodes (N). (Channel SNR is $5dB$, delay budget is $100ms$, and data rate is $2Mbps$.)

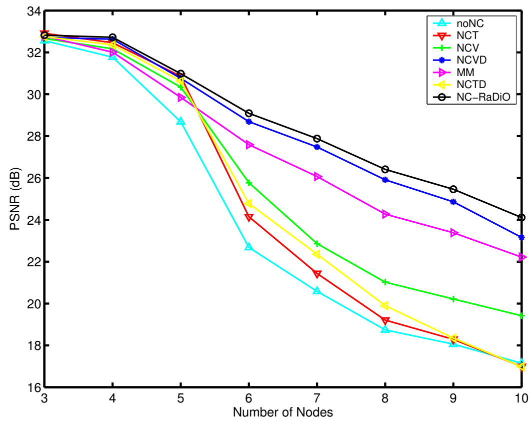


Figure 3.12: Grid topology. PSNR values achieved by all algorithms, for a different number of streams in the system. (Channel SNR is $5dB$, delay budget is $100ms$, and data rate is $1Mbps$.)

using one-hop or two-hops. In particular, node I goes sequentially through the list of the 8 available videos (*Foreman, Mother & Daughter, Carphone, Coastguard, Salesman, News, Grandma, and Claire*) and sends one to each receiver. When there are more than 8 nodes, the 8^{th}, 9^{th}, etc. sequence is chosen from the beginning of the list (*Foreman, Mother & Daughter, etc*).

In Fig. 3.12, we show the PSNR achieved by all algorithms for a varying number of streams in this topology. *E.g.*, $N = 5$ means there is one transmitter I and 4 receivers over either one- or two-hops. The delay budget is $100ms$, the channel SNR $5dB$ and the channel data rate $1Mbps$. The figure shows a similar trend with the corresponding graph for the downlink topology in Fig. 3.9(a). This is because the traffic scenarios in the downlink and the grid topologies are similar: the grid scenario consists of one and two hop downlink transmissions. However, there are two differences. First, the decrease in PSNR is sharper in the grid topology: when the number of nodes increases, more nodes are involved in one- and two-hop transmissions, as compared to the downlink topology. Second, the difference between network coding and non network coding schemes is smaller, because after the first hop there are not many network coding opportunities.

3.3.3 Complexity

The main complexity of NCV comes from considering all possible candidate codes. However, this is no worse than the complexity of NCT: they both consider all possible codes but they evaluate them using a different metric. An important observation is that real-time delay requirements significantly reduce the number of packets in the virtual buffers and therefore the complexity, making the brute-force approach feasible. For a larger delay budget, approximation algorithms for NCV and NCT can be developed by formulating them as a maximum weight independent set problem. Although this problem is NP-complete, it is also well-studied and approximation algorithms can be found in the literature [80].

NCVD runs NCV for each packet (considered as primary) in the Tx queue and selects the best overall code. The NCVD complexity is linear in the number of packets in the Tx queue, which is also small for real-time applications. The main complexity is still due to the NCV part.

Finally, the optimal solution to NC-RaDiO can be thought of as employing NCVD at all nodes with the additional ability to compare improvement values (Lagrange multipliers) among nodes in a neighborhood, in order to decide which node should transmit. This brings a small increase in complexity (on the order of $\log(N)$ where N is the number of nodes) but can be costly in terms of network resources. In this work, we consider NC-RaDiO mainly as a baseline for comparison with the simpler and near optimal NVC/NCVD algorithms.

3.4 Summary

In this chapter, we proposed a novel approach to opportunistic video coding for video streaming over wireless networks that take into account the importance of video packets in network code selection. Essentially, our approach combines for the first time together ideas from (i) network coding for increasing throughput and (ii) prioritized transmission for improving video quality, taking into account distortion and deadlines. Simulation results show that the proposed schemes improve video quality up to $5dB$ compared to baseline schemes. Furthermore, they significantly improve the application-level throughput while achieving the same or similar levels of MAC throughput.

Chapter 4

Rate Control and Scheduling over Coded Wireless Networks

In this chapter, we study rate control over coded wireless networks, and optimize rate control for video applications and TCP flows over such networks. First, we argue the importance of cross-layer optimization, *i.e.,* making end-to-end rate control and local scheduling aware of the underlying network coding operations. Our key intuition is as follows. When network coding is used, the achievable rate region gets extended by coding some flows together and broadcasting them. However, this introduces constraints on transmission rates of applications, *i.e.,* they affect the throughput over coded networks. Furthermore, additional conflicts exist due to network coded flows. In this chapter, we argue that these constraints and conflicts should be explicitly taken into account in terms of cross-layer control, in order to fully exploit network coding benefit.

Second, we consider video streaming over coded wireless networks and the effect of its rate requirements to network coding benefit. We show that the rates at the video/application layer affect the availability of network coding opportunities at the underlying network coding layer and thus the achievable region. In this chapter, we propose schemes that exploit network coding opportunities better by delaying some video scenes, and optimizing the rate allocation over long time intervals.

Third, we consider TCP flows over coded wireless networks. TCP flows do not fully exploit the network coding opportunities due to their bursty behavior and due to the fact that TCP is agnostic to the underlying network coding. In this chapter, we formulate the the problem as network utility maximization and we develop a distributed solution. Based on the structure of the optimal solution, we propose minimal modifications to congestion control and queue management mechanisms so as to make them network-coding aware. The results indicate that we are able to double TCP throughput over wireless coded networks.

In the rest of this chapter, we first present the system model that we consider in this chapter. In Section 4.2, we discuss the importance of network coding awareness by comparing our network coding aware (NC-aware) and unaware (NC-unaware) rate control and scheduling schemes over coded wireless networks. In Sections 4.3 and 4.4, we present cross layer optimization of video and TCP flows over coded wireless networks based on our useful insights from NC-aware and NC-unaware schemes. Finally, in Section 4.5, we summarize our contributions and conclude the chapter.

4.1 System Model

Sources/Flows: Let $\mathcal{S}$ be the set of unicast flows between some source-destination pairs. Each flow $s \in \mathcal{S}$ is associated with a rate x_s and a utility function $U_s(x_s)$, which we assume to be a strictly concave function of x_s. The goal is to maximize the total utility function $U_t = \sum_{s\in\mathcal{S}} U_s(x_s)$.

Wireless Network: A hyperarc $(i, \mathcal{J})$ is a collection of links from node $i \in \mathcal{N}$ to a non-empty set of next-hop nodes $\mathcal{J} \subseteq \mathcal{N}$ that are interested in receiving the same network code through a broadcast transmission from i. A hypergraph $\mathcal{H} = (\mathcal{N}, \mathcal{A})$ represents a wireless mesh network, where $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ is the set of hyperarcs. For simplicity, $h = (i, \mathcal{J})$ denotes a hyperarc, $h(i)$ denotes node i and $h(\mathcal{J})$ denotes node $\mathcal{J}$, *i.e.*, $h(i) = i$ and $h(\mathcal{J}) = \mathcal{J}$. We use these terms interchangeably in the rest of the chapter.

Due to the shared nature of the wireless media, transmission over different hyperarcs may interfere with each other. We consider the protocol model of interference [81], according to which, each node can either transmit or receive at the same time and all transmissions in the range of the receiver are considered as interfering. Given a hypergraph $\mathcal{H}$, we can construct the conflict graph $\mathcal{C} = (\mathcal{A}, \mathcal{I})$, whose vertices are the hyperarcs of $\mathcal{H}$ and edges indicate interference between hyperarcs. A clique $\mathcal{C}_q \subseteq \mathcal{A}$ consists of several hyperarcs, at most one of which can transmit at the same time without interference.

Network Coding: We assume that intermediate nodes use COPE [10] for one-hop opportunistic network coding[1]. Each node i listens all transmissions in its neighborhood, stores the overheard packets in its decoding buffer, and periodically advertises the content of its decoding buffer to its neighbors. Then, when a node i wants to transmit a packet, it checks or estimates the contents of the decoding buffer of its neighbors. If there is a network coding opportunity, the node combines the relevant packets using simple coding operations (XOR) and broadcasts the combination to $\mathcal{J}$. Note that it is possible to construct more than one network code over a hyperarc $(i, \mathcal{J})$. Let $\mathcal{K}_{i,\mathcal{J}}$ be the set of network codes over a hyperarc $(i, \mathcal{J})$. Let $\mathcal{S}_k \subseteq \mathcal{S}$ be the set of flows, whose packets are coded together using code $k \in \mathcal{K}_{i,\mathcal{J}}$ and broadcast over $(i, \mathcal{J})$.

Routing: We consider that each flow $s \in \mathcal{S}$ follows a single path $\mathcal{P}_s \subseteq \mathcal{N}$ from the source to the destination. This path is pre-determined by a routing protocol, *e.g.,* OLSR or AODV, and given as input to our problem. However, note that several different hyperarcs may

[1]Note that we present the multi-hop extension in Section 4.4.4.

connect two consecutive nodes along the path. We set an indicator function $H_{i,\mathcal{J}}^{s,k} = 1$ if flow s is transmitted through hyperarc $(i, \mathcal{J})$ using network code $k \in \mathcal{K}_{i,\mathcal{J}}$. Otherwise, $H_{i,\mathcal{J}}^{s,k} = 0$. Similarly, we consider an indicator function $H_{i,\mathcal{J}}^{s} = 1$ if flow s is transmitted through hyperarc $(i, \mathcal{J})$. Otherwise, $H_{i,\mathcal{J}}^{s} = 0$.

4.2 The Importance of Network Coding-Aware Rate Control and Scheduling

In this section, we discuss the importance of network coding-aware rate control and scheduling. Our key intuition is as follows. When network coding is used, the achievable rate region gets extended by coding some flows together and broadcasting them. However, this introduces additional scheduling conflicts. For example, when two flows transmitted in reverse directions are coded together at an intermediate node, there is a new network coded flow created, which may conflict with other flows transmitted in the neighborhood.

Example 6 The example shown in Fig. 4.1 demonstrates the key intuition why we need network coding awareness in such scenarios. There are two flows in reverse directions: node A transmits the first flow with rate x_1 to node D via nodes B and C; node C transmits the second flow with rate x_2 to node A via node B. All nodes transmit in the same channel and at the same power level. The link capacities (C_1, C_2, C_3 for links $A - B$, $B - C$, $C - D$) inversely depend on the distance between the node pairs.

Let $x_{A,B}^1$, $x_{B,C}^1$, and $x_{C,D}^1$ be the uncoded parts of the first flow, while $x_{C,B}^2$ and $x_{B,A}^2$ are the uncoded parts of the second flow. $x_{B,\{A,C\}}^1$ and $x_{B,\{A,C\}}^2$ are the network coded part of both flows: node B combines parts of x_1 and x_2 and broadcasts to both A and C. From the flow conservation we have that: $x_1 = x_{A,B}^1 = x_{B,C}^1 + x_{B,\{A,C\}}^{1,2} = x_{C,D}^1$ and $x_2 = x_{C,B}^2 = x_{B,A}^2 + x_{B,\{A,C\}}^{1,2}$. If the scheme is NC-aware, it will take into account that there is a new

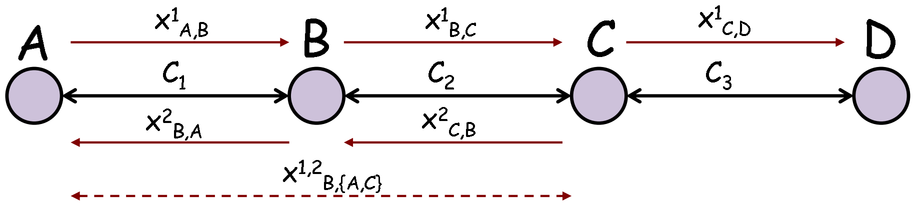


Figure 4.1: Motivating example. A sends the first flow with rate x_1 to D through nodes A and B. C sends the second flow with rate x_2 to A through B.

network coded flow, transmitted with rate $\max(x^1_{B,\{A,C\}}, x^2_{B,\{A,C\}})$ and interfering with all other transmissions in the system. Therefore, no simultaneous transmissions will take place. However, if the scheme is NC-unaware, it does not know that there is a transmission for the network coded flow from B to A and C. Therefore, when link $B-A$ is used for transmission of the second flow and network coding is possible with packets of the first flow, then packets are combined and transmitted. However, since transmissions from B to A and C to D are considered as interference-free, in the NC-unaware scheme, they will be scheduled at the same time. Thus, network coding at B may lead to a collision at C. As a result, network coding opportunities can be wasted and the total achieved flow rate in the worst case reduces to the scenario where no network coding is applied. □

4.2.1 Optimal Rate Control and Scheduling

Optimal NC-Aware Scheme: We follow the link-based approach in [62, 61] for cross-layer optimization of wireless networks and extend it to include network coding. Our goal is to optimize the total utility U_t.

$$\begin{aligned}
\max_{\boldsymbol{x},\boldsymbol{\tau}} \quad & \sum_{s\in S} U_s(x_s) \\
s.t \quad & H_h^s x_h^s \le R_h \tau_h^k \xi_h^s, \forall k \in \mathcal{K}_h, \forall s \in \mathcal{S}_k, \forall h \in \mathcal{A} \\
& x_s = \sum_{\{h(\mathcal{J})|h\in\mathcal{A}, i\in\mathcal{P}_s\}} x_h^s, \forall s \in \mathcal{S}, i \in \mathcal{P}_s \\
& \sum_{h\in\mathcal{C}_q} \sum_{k\in\mathcal{K}_h} \tau_h^k \le \gamma, \forall \mathcal{C}_q \subseteq \mathcal{A}
\end{aligned} \tag{4.1}$$

The rate control aims at selecting the rate $x_s \ge 0$ at each source $s \in \mathcal{S}$, and the parts of it ($x_h^s \ge 0$) that are transmitted over each hyperarc h on the path $\mathcal{P}_s$ using predetermined network codes $k \in \mathcal{K}_h$, where $\mathcal{K}_h$ is the set of network codes over h. The first constraint refers to flows $s \in \mathcal{S}_k$ coded together in the same network code k and transmitted over h if flow s is transmitted over h (*i.e.,* $H_h^s = 1$). These flows coexist and do not compete for the total rate $R_h\tau_h^k\xi_h^s$ allocated to code k, where R_h is the maximum achievable rate over h, τ_h^k is the percentage of time that h is used for network code $k \in \mathcal{K}_h$, and ξ_h^s is the probability of successful transmission from node i to its destination node $j \in \mathcal{J}$. Thus, $R_h\tau_h^k\xi_h^s$ is the effective rate used by code k after excluding packet losses. The maximum rate of these network coded flows ($s \in \mathcal{S}_k$) should be up to the effective rate of the code k, which means that the rate of each one of them ($H_h^s x_h^s$) should be up to $R_h\tau_h^k\xi_h^s$. The second constraint is the flow conservation of flow x_s at each hop on its path $\mathcal{P}_s$ towards the destination. Note that a flow may be transmitted over different hyperarcs with different network codes and rates x_h^s which are summed up to x_s with this constraint. The third constraint captures the conflicts due to interference, similarly to [39]: different network codes over the same hyperarcs and nodes in the same clique cannot transmit at the same time. They share the available transmission time, γ.

Distributed Solution: The set of constraints in Eq. (4.1) couples together the rate control and scheduling problems. Using Lagrangian relaxation for the first set of constraints in Eq. (4.1) with multiplier $q_h^{k,s}$ and by appropriately re-arranging the terms and constraints, the problem is decomposed into the following parts.

Rate Control: The rate control problem is decoupled from scheduling and can also be further decomposed into a number of rate-control subproblems, each of which can be solved independently at each source using only feedback from the network (about the Lagrange multipliers $q_h^{k,s}$ which are interpreted as queue sizes).

$$
\begin{aligned}
\max_{x_s} \quad & [U_s(x_s) - (\sum_{i \in \mathcal{P}_s} \sum_{\{h(\mathcal{J})|h \in \mathcal{A}, i \in \mathcal{P}_s\}} q_h^{k,s} H_h^s x_h^s)] \\
s.t. \quad & x_s = \sum_{\{h(\mathcal{J})|h \in \mathcal{A}, i \in \mathcal{P}_s\}} x_h^s
\end{aligned}
\tag{4.2}
$$

This problem can be solved by constrained convex optimization. However, the objective function in Eq. (4.2) is concave but not strictly concave. Therefore, the variables x_h^s's oscillate when we solve this optimization problem directly. To address this problem we consider proximal method which solves the equivalent problems of Eq. (4.2) and eliminates the oscillations, [82]. In particular, the proximal method considers the following problem, which is equivalent to the original one in Eq. (4.2):

$$
\begin{aligned}
\max_{x_s} \quad & [U_s(x_s) - (\sum_{i \in \mathcal{P}_s} \sum_{\{h(\mathcal{J})|h \in \mathcal{A}, i \in \mathcal{P}_s\}} q_h^{k,s} H_h^s x_h^s)] \\
& -c((x_s - y_s)^2 + \sum_{i \in \mathcal{P}_s} \sum_{\{h(\mathcal{J})|h \in \mathcal{A}, i \in \mathcal{P}_s\}} (x_h^s - y_h^s)^2) \\
s.t. \quad & x_s = \sum_{\{h(\mathcal{J})|h \in \mathcal{A}, i \in \mathcal{P}_s\}} x_h^s
\end{aligned}
\tag{4.3}
$$

where c is a small positive constant, and $\{y_s : \forall s \in \mathcal{S}\}$ and $\{y_h^s : \forall s \in \mathcal{S}, h \in \mathcal{A}\}$ are auxiliary variables introduced by the proximal method to eliminate oscillations. Periodically, y_s is set to x_s and y_h^s is set to x_h^s and the iteration continues using the new values of auxiliary variables.

Scheduling:

$$\begin{aligned} \max_{\boldsymbol{\tau}} \quad & \sum_{h \in \mathcal{A}} \sum_{k \in \mathcal{K}_h} R_h \tau_h^k Q_h^k \\ s.t. \quad & \sum_{h \in \mathcal{C}_q} \sum_{k \in \mathcal{K}_h} \tau_h^k \leq \gamma, \forall \mathcal{C}_q \subseteq \mathcal{A} \end{aligned} \tag{4.4}$$

where $Q_h^k = \sum_{s \in \mathcal{S}} q_h^{k,s} \xi_h^s$. This problem must be solved for all the hyperarcs in the network considering the interference model, so as to determine the percentage of time τ_h^k that hyperarc h should be used for network code k. This problem is known to be NP-hard and can be converted to a maximum weighted matching problem as explained in [62], for which heuristics have been developed in other contexts [83, 84], and specifically in the context of scheduling for ad-hoc networks [63, 64]. Since the focus of this section is on rate control and not on scheduling, we assume perfect scheduling but we note that (i) such heuristics should be employed in practice and (ii) the effect of imperfect scheduling on rate control must be investigated.

Parameter Update: We use a subgradient method to iteratively calculate the solution to problems (4.3) and (4.4):

$$q_h^{k,s}(t+1) = \{q_h^{k,s}(t) + \beta_t [H_h^s x_h^s - R_h \tau_h^k \xi_h^s]\}^+, \tag{4.5}$$

where β_t is a small constant that determines the convergence rate of our algorithm. $q_h^{k,s}(t)$ is the Lagrange multiplier at iteration t and can also be interpreted as the queue size at node i for the part of flow s transmitted over hyperarc h with code k.

Numerical Results for Convergence: We now show results for the rate allocation problem for the example of Fig. 4.2, using the proximal method in Eq. (4.3) and assuming perfect scheduling. In particular, we show the convergence of rates (x_1, x_{11}, x_{12}, x_{13} and x_2, x_{21}, x_{22}, x_{23}) and of the queue sizes $q_h^{k,s}$. For brevity of notation in this discussion as well as in Fig. 4.3 and Fig. 4.4, we renamed the flow rates mentioned in the example in Fig. 4.2. [2] There are also six Lagrange multipliers (queue sizes), one associated with each of these flow rates.[3] Although any concave function can be used as utility function, we choose to use logarithms, *i.e.,* the form $U(x_s) = w_s \log(x_s)$, which are typically used to provide weighted proportional fairness where w_s is the weight term. We consider two scenarios: (i) sources with the same utility functions; $U(x_1) = \log(x_1)$, $U(x_2) = \log(x_2)$ and (ii) sources with different utility functions; $U(x_1) = 4\log(x_1)$, $U(x_2) = \log(x_2)$.

Fig. 4.3 shows the numerical results for the first scenario. The main observation is that both rates and queue sizes converge. Furthermore, x_1 and x_2 converge to the same value because they have the same utility function. A closer look also reveals that the crossing flows x_1 and x_2 are always coded at the intermediate node: the network-coded flows (x_{13} and x_{23}) converge to the values of the total flows x_1, x_2, while the non-network coded flows (x_{12} and x_{22}) converge to 0.

Fig. 4.4 shows the numerical results for the second scenario. Both rates and queue sizes converge again. However, because the utility of the first user is now weighted more ($w_1 = 4$) than the second ($w_2 = 1$), this user ends up transmitting at higher rate. In this case, some part of the flows is network coded (x_{13}, x_{23}) while the rest is transmitted without any coding (*e.g.,* $x_{12} > 0$).

[2]The renaming is as follows. $x_{11} = x^1_{A,\{Relay\}}$ and $x_{21} = x^2_{B,\{Relay\}}$ are still the original flows sent on the uplink. x_{12}, x_{13} are the non-coded and coded parts of x_1 sent on the downlink: $x_{12} = x^1_{Relay,\{B\}}$ and $x_{13} = x^1_{Relay,\{A,B\}}$. x_{22}, x_{23} are the non-coded and coded parts of x_1 send on the downlink: $x_{22} = x^2_{Relay,\{A\}}$ and $x_{23} = x^2_{Relay,\{A,B\}}$. The flow conservation dictates that $x_1 = x_{11} = x_{12} + x_{13}$ and $x_2 = x_{21} = x_{22} + x_{23}$.

[3]q_1 and q_2 are associated with x_1 and x_2; q_3 and q_4 are associated with x_{12} and x_{13}; q_5 and q_6 are associated with x_{23} and x_{22}.

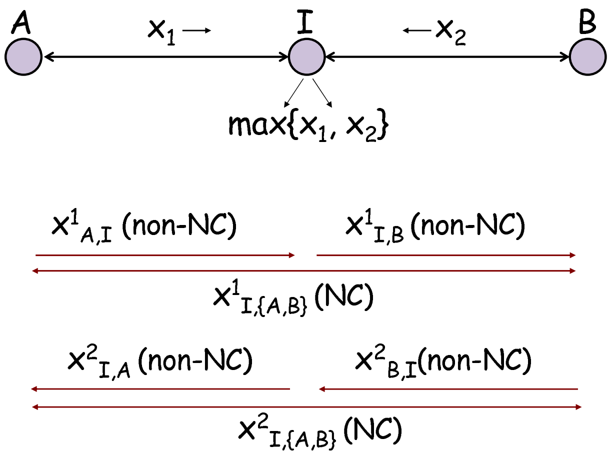
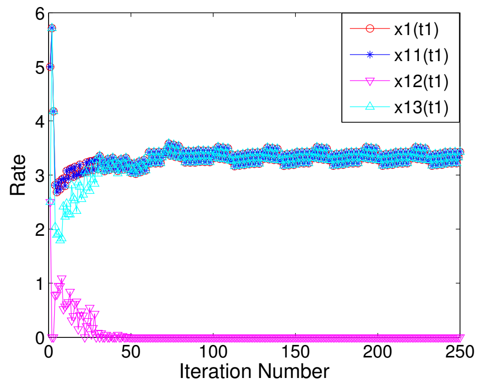
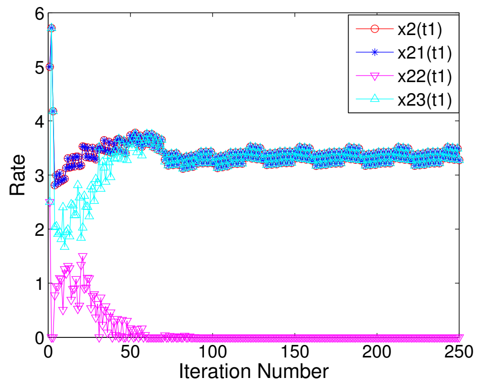


Figure 4.2: A and B send flows x_1 and x_2 to each other through an intermediate node I. Nodes use the same frequency and share the channel capacity (assuming all hyperarcs have the same channel capacity C) using time sharing. The relay node can transmit (broadcast) parts of these flows uncoded ($x^1_{A,I}$, $x^1_{I,B}$, $x^2_{B,I}$, $x^2_{I,A}$) or coded ($x^1_{I,\{A,B\}}$, $x^2_{I,\{A,B\}}$). Packets in the coded flows are constructed by XOR-ing packets from the two flows. The flow conservation for the first flow requires that $x_1 = x^1_{A,I} = x^1_{I,B} + x^1_{I,\{A,B\}}$ and similarly for the second flow. There are five hyperarcs: (A, I), (I, B), (B, I), (I, A) and $(I, \{A, B\})$.

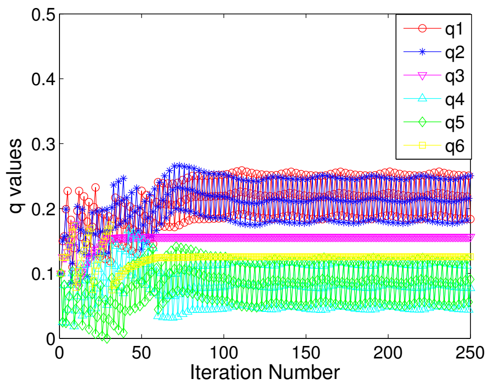
Optimal NC-Unaware Scheme: To quantify the benefit of network coding awareness, we compare the optimal NC-aware scheme to its NC-unaware counterpart. The latter scheme takes decisions based on (i) the total queue length at each link of every node and (ii) the conflict graph; however, it does not know the more detailed information about the queue size per coded or uncoded flow. In the example of Fig. 4.1, there are actually four queues at node B (two for uncoded flows; $x^1_{B,C}$, $x^2_{B,A}$, and two for the coded flows; $x^2_{B,\{A,C\}}$ and $x^2_{B,\{A,C\}}$), which are known to the NC-aware scheme. However, the NC-unaware scheme only knows the existence of two output queues: one for link $B - A$ and one for link $B - C$. Apart from being agnostic to the existence of network coding, the optimal NC-unaware scheme is similar



(a) Convergence of rate x_1 (where $x_{11} = x_1$: uplink rate, x_{12}: downlink non-coded rate, x_{13}: downlink coded rate).



(b) Convergence of rate x_2 (where $x_{21} = x_1$: uplink rate, x_{22}: downlink non-coded rate, x_{23}: downlink coded rate).

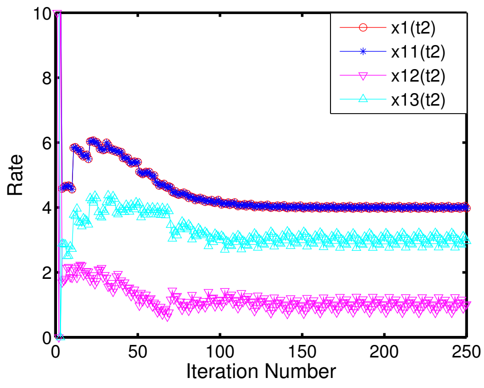


(c) Convergence of queue sizes associated with the rates (q_1 with x_{11}, q_2 with x_{22}, q_3 with x_{12}, q_4 with x_{13}, q_5 with x_{23}, q_6 with x_{22}.

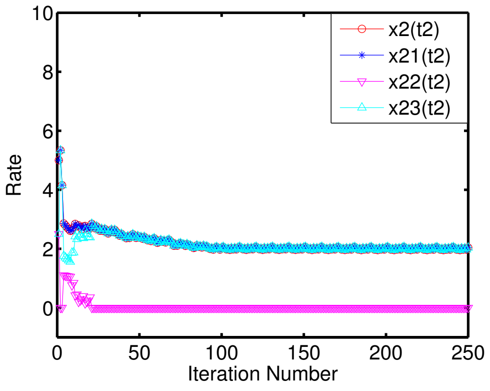
Figure 4.3: Convergence of variables for the example in Fig. 4.2 and similar utilities: $U(x_1) = \log(x_1)$, $U(x_2) = \log(x_2)$.

to the NC-aware one as formulated in the following;

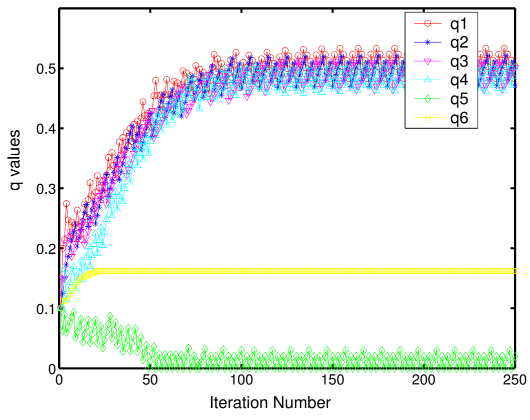
$$\begin{aligned} \max_{\boldsymbol{x},\boldsymbol{\tau}} \quad & \sum_{s\in\mathcal{S}} U_s(x_s) \\ s.t \quad & \sum_{s\in\mathcal{S}} H^s_{i,j} x_s \leq R_{i,j}\tau_{i,j}\xi_{i,j}, \forall (i,j)\in\mathcal{A} \\ & \sum_{(i,j)\in\mathcal{C}_q} \tau_{i,j} \leq \gamma, \forall \mathcal{C}_q \subseteq \mathcal{A} \end{aligned} \tag{4.6}$$



(a) Convergence of rate x_1 (where $x_{11} = x_1$: uplink rate, x_{12}: downlink non-coded rate, x_{13}: downlink coded rate).



(b) Convergence of rate x_2 (where $x_{21} = x_1$: uplink rate, x_{22}: downlink non-coded rate, x_{23}: downlink coded rate).



(c) Convergence of queue sizes associated with the rates (q_1 with x_{11}, q_2 with x_{22}, q_3 with x_{12}, q_4 with x_{13}, q_5 with x_{23}, q_6 with x_{22}).

Figure 4.4: Convergence of variables for the example in Fig. 4.2 and different utilities: $U(x_1) = 4\log(x_1)$, $U(x_2) = \log(x_2)$.

Eq. (4.6) can also be solved using Lagrangian decomposition. In the rate control part, each source simply determines its rate x_s. The scheduling problem is solved considering all conflicting links in the system. Each queue is updated when packets are transmitted or received, without distinguishing whether they are delivered with or without network coding.

Optimal NC-Aware versus Optimal NC-Unaware Scheme: In Table 4.1, we present numerical results of the achieved rates for the example of Fig. 4.1 for four different scenarios,

Table 4.1: Optimal Schemes. Achievable rates for the example of Fig. 4.1. Scenario 1: $C_1 = C_2 = C_3 = 1$; Scenario 2: $C_1 = C_3 = 1, C_2 = 2$; Scenario 3: $C_1 = 1, C_2 = C_3 = 2$; Scenario 4: $C_1 = C_3 = 1, C_2 = 4$.

	Optimal NC-aware			Optimal NC-unaware		
Scenarios	x_1	x_2	$x_1 + x_2$	x_1	x_2	$x_1 + x_2$
1	0.21	0.39	0.60	0.20	0.40	0.60
2	0.25	0.50	0.75	0.33	0.32	0.65
3	0.31	0.46	0.77	0.32	0.30	0.62
4	0.33	0.50	0.83	0.38	0.37	0.75

which correspond to different values of link rates C_1, C_2, C_3 for links $A - B$, $B - C$, $C - D$, respectively. The results are generated with 500 iterations. The utility function is assumed to be $U_s(x_s) = \log(x_s)$. We can see that, in all scenarios, the NC-aware scheme achieves the same or higher total rate ($x_1 + x_2$) compared to the NC-unaware scheme. In some scenarios, the NC awareness brings significant improvement: this usually happens in scenarios where (i) the underlying conflicts of network coded flows reduce the network coding opportunities and (ii) there are no network opportunities over different links.

4.2.2 Practical Rate Control and Scheduling

We now propose a practical implementation of the optimal NC-aware and NC-unaware rate control and scheduling schemes. We use simulations to show that they are good approximations of the optimal schemes and we also compare them to quantify the benefit from NC-awareness in practice.

To create a practical scheme we need to make the following modifications: (i) implement a packet-based rate control that approximates the optimal flow-based rate control, (ii) implement a low-complexity yet efficient heuristic that approximates the optimal scheduling, (iii) signal information about queue size and conflicts, and (iv) construct and update queues. These modifications apply similarly to both the NC-aware and NC-unaware schemes with some minor differences, as described below.

Rate Control: The optimal NC-aware rate control determines the flow rate of each network coded flow according to the queue length of each network code on every node on the flow's path. Instead, similar to [64], each source maintains an average rate value for each of its network coded rates. At each transmission opportunity the optimal rates are compared to the average rates. If the average rate of a network coded part of a flow is less than its optimal value, a packet is inserted to the transmission buffer and labeled with the network coding policies for each node on its path; a packet is coded according to its label at each node on its path to the destination. The NC-unaware rate control employs exactly the same scheme but maintains the average rate at each source instead of the average rates for all coded flows.

Scheduling: Scheduling determines the nodes and flows that will transmit and the percentage of time a node will transmit. This problem is NP-hard [61] and we use a heuristic similar to [64]. Each node maintains information about the queue size of its own flows and of its neighbors' flows and exchanges this information with its neighbors at the end of each packet transmission. This way, every node learns the queue sizes of its one- and two-hop neighbors, and compares its own queue sizes with the queue sizes of these neighbors. If a node has the largest queue, it tries to transmit a packet from this queue, by selecting a small initial value for the contention window in 802.11. The practical NC-unaware scheduling uses the same mechanism.

Signaling: Queue size information is exchanged after transmitting a packet. In practice, when 802.11 is used as an underlying MAC mechanism, the queue size information can be exchanged via RTS/CTS control packets, as in [64]. In addition to the queue sizes, we consider that queue destination information is also appended to the control packets, in order for each node to determine its exposed terminals. Note that we consider that two nodes whose destination nodes are in different interference regions can transmit at the same time even if they are interfering to each other (exposed terminal problem) by figuring out conflicts considering destination nodes. We consider a synchronization among nodes to eliminate collisions of ACKs and control packets. In summary, at each RTS/CTS transmission, nodes exchange their queue sizes, destination nodes and their neighbors' queue sizes and destination nodes. In addition, queue size information at each node over the path of a source is passed to the source via feedback. The signaling mechanism of the practical NC-unaware scheme is exactly the same.

Parameter Update: In the NC-aware case, each node maintains a queue for each source and network code, which gets updated when a packet is transmitted or received. In the NC-unaware case, information is maintained for each output queue (link) at each node. However, even when two or more packets are transmitted from/to a queue or more than one queue transmits due to network coding, the queues are updated accordingly.

Simulation Results: We now present simulation results for the practical NC-aware and NC-unaware schemes for the example of Fig. 4.1. Table 4.2 shows the rates achieved by the practical schemes in the same four scenarios discussed in section 4.2.1. There are two observations to make from this table. First, the NC-aware scheme achieves higher total rate than the NC-unaware one, in this practical case as well and for all four scenarios. Second, comparing the practical schemes to the corresponding optimal schemes, we see that the rates achieved are lower. This is expected as the practical schemes are suboptimal: *e.g.,* the control packets (RTS/CTS) use 10% of transmission time and reduce the total rate. Third, the rates achieved by the practical schemes are close to the optimal rates, which indicates that they are efficient heuristics.

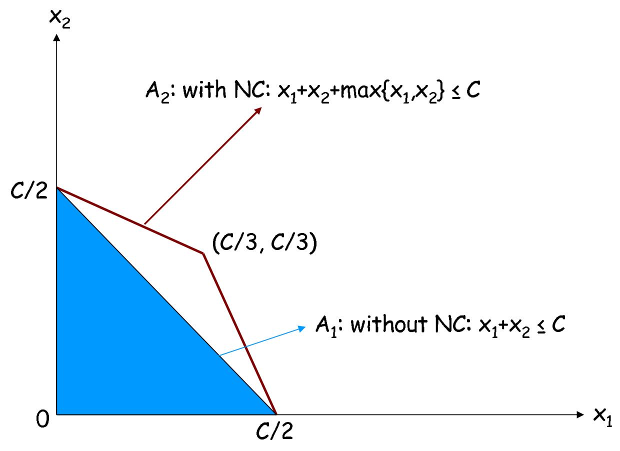
Table 4.2: Practical Schemes. Achievable rates for the example of Fig. 4.1. Scenario 1: $C_1 = C_2 = C_3 = 1$; Scenario 2: $C_1 = C_3 = 1, C_2 = 2$; Scenario 3: $C_1 = 1, C_2 = C_3 = 2$; Scenario 4: $C_1 = C_3 = 1, C_2 = 4$.

	Practical NC-aware			Practical NC-unaware		
Scenarios	x_1	x_2	$x_1 + x_2$	x_1	x_2	$x_1 + x_2$
1	0.19	0.36	0.55	0.19	0.35	0.54
2	0.24	0.46	0.70	0.29	0.30	0.59
3	0.28	0.44	0.72	0.29	0.30	0.59
4	0.28	0.46	0.74	0.35	0.37	0.72

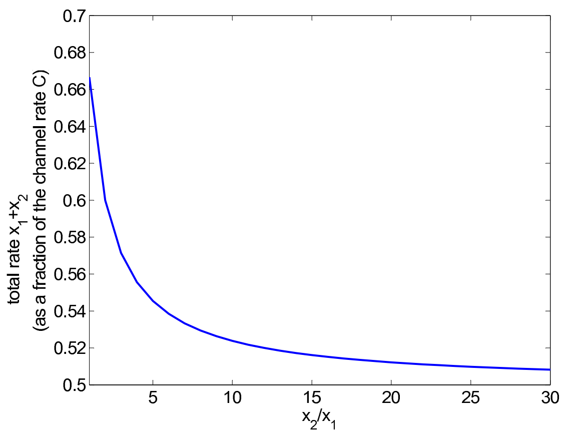
4.3 Network Coding-Aware Rate Control for Video

In this section, we study network coding-aware rate control for video streaming over coded wireless networks. Motivated by the analysis we made in the previous section, we have observed that the time-varying nature of video content implies time-varying utilities and affects the underlying network coding opportunities. Our key motivation in this section is that by delaying some scenes and by optimizing the rate allocation over longer time intervals, we can create more network coding opportunities and thus achieve higher total utility.

Let us revisit the basic topology presented in Fig. 4.2. Without network coding, A and B transmit at rates x_1, x_2 such that $2x_1 + 2x_2 \leq C$, leading to the triangle rate region A_1 shown in Fig. 4.5(a). The total achievable rate is constant: $x_1 + x_2 = C/2$. When network coding is used, and A and B transmit at rates x_1, x_2, then, on the downlink, the network coded flow has rate $\min(x_1, x_2)$ and the total flow has rate $\max(x_1, x_2)$. The achievable rate region is extended to the kite region; $A_2 = \{(x_1, x_2) : \; x_1 + x_2 + \max(x_1, x_2) \leq C\}$, shown in Fig. 4.5(a). The maximum total achievable rate is now higher, $2C/3$, and is achieved when both flows transmit at the same rate $x_1 = x_2 = C/3$. If $x_1 \neq x_2$ the total throughput is less than this maximum achievable value. Fig. 4.5(b) shows that the total rate $x_1 + x_2$ decreases from the maximum $2C/3$ (when $x_1 = x_2$) to $C/2$ (the value without network coding) for increasing discrepancy between the two rates (x_2/x_1). In summary, the more similar the



(a) Achievable rate region (x_1, x_2) with and without network coding.



(b) With network coding the rate region is $A_2 = \{(x_1, x_2) : x_1 + x_2 + max\{x_1, x_2\} \leq C\}$. Let $\frac{x_2}{x_1} = \alpha \geq 1$. The total achievable rate $(x_1 + x_2 = \frac{\alpha+1}{2\alpha+1}C)$ decreases as the ratio between individual rates increases.

Figure 4.5: Achievable rates for the example in Fig. 4.2, with and without network coding. C is the channel capacity of each hyperarc which is time shared among all flows.

rates of the cross flows, the more coding opportunities and the higher the total throughput. This key observation must be taken into account in rate allocation and is exploited in this section.

In the video community, video specific rate control and playout optimization have been extensively studied in the past, albeit not for networks with network coding. Works close to ours include, but are not limited to, the following: [85], which controls the playout to make the video rate smoother by adapting to varying wireless channel capacity; [86], which studied joint scheduling and playout considering the video content; [87], which developed congestion-distortion optimized distributed rate allocation for streaming over wireless with heterogeneous links. The time-varying video content was also taken into account in [88]: scenes with different rates were transmitted over a channel with fixed and limited data rate; by delaying some scenes when the rate of a scene was higher than the channel rate, delay-distortion optimized streaming was achieved. Different from these works, we consider network coding over wireless networks; in this case, the maximum achievable channel data rate is no longer fixed but depends on the transmitted data rate. Our focus is on the interaction between flow rates and achievable rate region so as to maximize the total video utility.

In the rest of this section, we first consider utility optimal video streaming, which fits naturally within the rate control framework presented in the previous section. However, specific to video, the utility functions must reflect the characteristics of video sequences, including their rate-distortion characteristics and the importance of different scenes to the users. A key observation is that the video quality varies over time depending on the video content; as a result, two or more video streams can have different rates over a short-time scale even though they may have similar rates over longer time scales. We propose to introduce additional delay in some scenes and to optimize the rate allocation over longer time intervals, so as to increase the network coding opportunities and eventually the total utility. Furthermore, this time interval should be selected so as to optimize the delay vs. utility tradeoff. We formulate the problem, develop a distributed solution and evaluate its performance, compared to the general rate control scheme, via numerical simulations in some illustrative scenarios.

4.3.1 Rate Control for Video Streaming over Coded Wireless Networks

Intuition: We are particularly interested in rate allocation for video streaming over coded wireless networks. Rate allocation for video is rate-based and fits naturally within the rate control framework presented in the previous section. Furthermore, the following aspects are specific to video. First, the utility functions should reflect the PSNR-rate curves for realistic video sequences and/or the importance of different scenes based on video content. The first key observation is that the video utility function varies over time, based on the video content. Therefore a natural question arises: what is the right time interval for computing the rate and utility and for optimizing the rate allocation? Clearly, short term variations in rate and utility will be smoothed out if we consider longer time intervals. Second, video streaming can tolerate some small delay. The second key observation is that by delaying some scenes and by considering a longer time interval we can create more network coding opportunities, which can eventually lead to higher total throughput and utility. As discussed at the end of the previous section, more network coding opportunities are created when the rates of cross flows are similar.

Combining these two observations, one should consider an appropriate time interval T, over which to compute the utility and perform optimal rate allocation, so as to smooth out short-term rate variations, thus create more network coding opportunities and increase the total rate and utility. The choice of the T involves a delay vs. utility tradeoff and depends on the video content and the delay constraints.

Example 7 Let us revisit the example in Fig. 4.2 to further clarify the above ideas. If A and B has the same utility function, so they will transmit at the same constant rate until the end of the communication and operate at the rate optimal corner $(C/3, C/3)$ in Fig.4.5(a). However, if A and B send video, their utility functions change in time (*e.g.,* for every scene) and are different from each other. As each source optimizes its own rate according to its scene utility function, the utility-optimal rates are not on the rate-optimal corner $(C/3, C/3)$ anymore, but on the border of the feasible region A_2 in Fig.4.5(a), and thus the network is not fully used. The key idea is that if users transmit at roughly similar and smooth rates, they will operate at a utility-optimal point which will be close to or on the rate-optimal point, thus resulting to higher operating rates and eventually higher total utility. In order to achieve this goal, instead of just optimizing the current scene's rate requirements, sources should optimize over a longer time period as described next. □

Formulation: We now formulate the problem of optimal rate allocation for time-varying video over a fixed time-interval T. Every video stream s is partitioned into temporal segments, which we call scenes. Every scene f of flow s is characterized by its duration $\Delta(f)$, rate $x_s(f)$ and utility function $U_s(x_s(f))$; the rate and utility change over time according to the video content. The optimization period T contains multiple scenes (the set $\mathcal{F}_s$) of stream s: $T = \sum_{f \in \mathcal{F}_s} \Delta(f)$. The fact that we want to do optimal rate allocation for all $\mathcal{F}_s$ scenes in the period T means that the scenes can be transmitted in parallel instead of sequentially within T; thus, they can be thought of as separate flows sharing the channel capacity during T. The optimization problem is in the following:

$$\begin{aligned}
\max_{\boldsymbol{x},\boldsymbol{\tau}} \quad & \sum_{s \in \mathcal{S}} \sum_{f \in \mathcal{F}_s} U_s(x_s(f))\delta(f) \\
s.t \quad & x_s(f) = \sum_{\{h(\mathcal{J})|h \in \mathcal{A}, i \in \mathcal{P}_s\}} x_h^s(f), \forall s \in \mathcal{S}, f \in \mathcal{F}_s \\
& \sum_{f \in \mathcal{F}_s} H_h^s x_h^s \leq R_h \tau_h^k \xi_h^s, \forall k \in \mathcal{K}_h, s \in \mathcal{S}_k, h \in \mathcal{A} \\
& x_s(f) \geq x_s^{min}(f), \forall s \in \mathcal{S}, f \in \mathcal{F}_s \\
& \sum_{h \in \mathcal{C}_q} \sum_{k \in \mathcal{K}_h} \tau_h^k \leq \gamma, \forall \mathcal{C}_q \subseteq A_c
\end{aligned} \tag{4.7}$$

This problem is similar to the general problem in Eq.(4.1), but has the following differences. First, the optimization is performed over the time interval T. Although T is not explicitly mentioned, it implicitly affects the number of consecutive scenes considered for transmission: $T = \sum_{f \in \mathcal{F}_s} \Delta(f)$. We assume that T is large enough to guarantee convergence of the solution within this interval. Second, we define the utility $U(x_s)$ of a stream s during the time period T as the weighted average of the utilities of individual scenes $\sum_{f \in \mathcal{F}_s} U_s(x_s(f))\delta(f)$. The weight $\delta(f) = \Delta(f)/T$ indicates the length of a scene as a fraction of T; it ensures that, for the same utility, longer scenes will transmit more. Another difference is in the second constraint: compared to the first constraint in the general problem in Eq.(4.1), there is now a summation over all the scenes considered. The reason is that each scene can be considered as a separate flow, which should share the total capacity with other scenes transmitted during T. Finally, we introduce the third constraint, which is new in this formulation: it guarantees that some minimum short-term rate requirement $x_s^{min}(f)$ will be met for each scene $f \in \mathcal{F}_s$, in order to guarantee that video quality does not drop below an acceptable level even in the short term. The parameter $x_s^{min}(f)$ can be pre-computed based on the content of the video and/or the user requirements and is an input to our problem. As a concrete example, in this section, we consider as minimum requirement that each scene f should achieve at least the average optimal rate $x_s^*(f)$ that would achieve if we optimized over a single scene duration, $T = \Delta(f)$, instead of multiple scenes: $x_s^{min}(f) = \frac{1}{T}\int_{\Delta(f)} x_s^*(f)dt$.

Distributed Solution: By a Lagrangian relaxation of the fourth constraint, the problem decomposes into the following parts.

Rate Control: By re-arranging the terms and constraints and by using the proximal method, similarly to what we did in Eq.(4.3), we obtain the following rate control sub-problems that can be solved independently at each source:

$$\begin{aligned}
\max_{x_s} \quad & \sum_{f\in\mathcal{F}_s}[U_s(x_s(f)) - (\sum_{i\in\mathcal{P}_s}\sum_{\{h(\mathcal{J})|h\in\mathcal{A},i\in\mathcal{P}_s\}} q_h^{k,s}H_h^s x_h^s(f)) \\
& -c((x_s(f)-y_s(f))^2 + \sum_{i\in\mathcal{P}_s}\sum_{\{h(\mathcal{J})|h\in\mathcal{A},i\in\mathcal{P}_s\}} (x_h^s(f)-y_h^s(f))^2)] \\
s.t. \quad & x_s(f) = \sum_{\{h(\mathcal{J})|h\in\mathcal{A},i\in\mathcal{P}_s\}} x_h^s(f) \\
& x_s(f) \geq x_s^{min}(f), \forall f \in \mathcal{F}_s
\end{aligned} \tag{4.8}$$

Scheduling: By rearranging the terms in the Lagrangian:

$$\begin{aligned}
\max_{\boldsymbol{\tau}} \quad & \sum_{h\in\mathcal{A}}\sum_{k\in\mathcal{K}_h} R_h\tau_h^k Q_h^k \\
s.t. \quad & \sum_{h\in\mathcal{C}_q}\sum_{k\in\mathcal{K}_h} \tau_h^k \leq \gamma, \forall\mathcal{C}_q \subseteq \mathcal{A}
\end{aligned} \tag{4.9}$$

where $Q_h^k = \sum_{s\in\mathcal{S}_k} q_h^{s,k}\xi_h^s$.

Parameter Update:

$$q_h^{k,s}(t+1) = \{q_h^{k,s}(t) + \beta_t[\sum_{f\in\mathcal{F}_s} H_h^s x_h^s(f) - R_h\tau_h^k\xi_h^s]\}^+ \tag{4.10}$$

Discussion: Although the previous model has been presented in terms of a single fixed value of T, there is flexibility in choosing this value. Different sources s can use different T_s (so as to smooth out their own short-term variations in their video content) independently from each other (without synchronizing with each other or coordinating to choose the same T). In addition, it is possible and beneficial to solve the problem not just for one but for several different values of T. Intuitively, the longer the optimization interval, the higher the achieved utility at the cost of higher delay. Therefore, each source s should tune the value of T_s taking into account its content so as to meet a desired delay-utility tradeoff; an example will be discussed in the numerical results below. Furthermore, the model can naturally include user arrivals and departures, by simply optimizing over the scenes that are active in each optimization interval.

The partitioning of a video stream into scenes was essential for us to deal with time-varying video content: we treated a scene as the minimum time slot, within which there is no time variation. However, the definition and analysis of scenes in a video stream is a research topic on its own and out of the scope of this work. *E.g.*, series of commercials, music clips, newscasting can be considered as different scenes in the same video stream; or scenes may refer to one or a few GOPs; in [88], scenes from a soccer game have been extracted and different importance has been assigned to them based on the pitch of the commentator's voice; in [86], we have assigned different importance to 2-3 second scenes, based on their motion intensity. In the context of this section, we consider a scene to consist of a number of consecutive frames with similar content and importance; we also consider the partitioning of a stream into scenes and their utility functions to be determined by a separate process and provided as input to our problem. Our goal is to optimize the rate allocation given this input.

4.3.2 Performance Evaluation

In the rest of the section, we perform simulations to evaluate the performance improvement from video rate allocation (Eq. (4.7)) compared to the general rate allocation (Eq. (4.1)), which did not take into account the variation of video utility over time. We consider again the illustrative example in Fig. 4.2 and two traffic scenarios: the first with logarithmic time-

Table 4.3: Scenario 1. Average rate of scenes for stream 1.

Scene Number	1	2	3	4	5	6
General Rate Control	4.53	2.01	4.23	2.00	4.23	2.00
Video Rate Control	4.98	2.04	4.66	2.00	4.66	2.00

varying utilities; and the second considering PSNR-rate curves and the importance of video content.

Scenario 1: We consider two video sequences, each consisting of six scenes. All scenes have the same fixed duration, 250 packets. (We note that we chose the scene durations long enough to allow for convergence. The optimization algorithm proceeds in iterations, each iteration corresponds to sending one packet.) The utility of each scene f of video sequence s is of the form $\sigma_f \log(x_s(f))$ where the weight σ_f changes across scenes. In particular, the utility of the first video for each scene is: $\vec{U_1}$ =[$4\log(x_1(1))$, $\log(x_1(2))$, $4\log(x_1(3))$, $\log(x_1(4)$, $4\log(x_1(5))$, $\log(x_1(6)$]; the utility of the second video is: $\vec{U_2}$ =[$\log(x_2(1)$, $4\log(x_2(2))$, $\log(x_2(3)$, $4\log(x_2(4))$, $\log(x_2(5)$], $4\log(x_2(6))$. These utilities have been chosen on purpose to illustrate the value of considering a longer time period so as to smooth out short-term time variations. The channel capacity of each link is considered 10. (We note that we purposely omit units for the rates, in this example only, as the results will only scale with the unit for channel and video rate.) For this scenario, we compare the general rate control (which does not consider an optimization interval but continuously computes the rate at each single iteration) and the video rate allocation (with the interval $T = 500$ iterations, *i.e.,* the duration of two scenes).

Fig. 4.6 shows the total rate transmitted by the two video streams together, plotted over iteration number for both rate control schemes. The general rate control computes the optimal rate at every iteration. *E.g.*, we can see than at every scene change (around 250, 500,... iterations) it reacts to the change of utility and eventually converges to an optimal

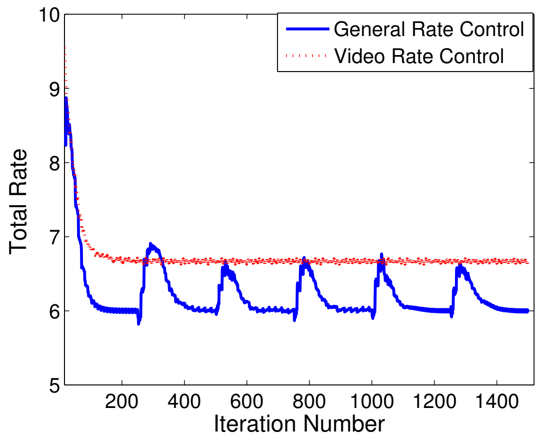


Figure 4.6: Scenario 1 (logarithmic utility functions). Total rate achieved by general and video (for $T = 500$) rate control.

value before the end of the scene. In contrast, the video rate control performs optimization over the duration T (500 iterations); thus, it achieves higher total rate (on average) and increases the total utility. In addition, it achieves smooth rate because, over the period of T, the utilities are optimal at similar rates. A natural question at this point is what happens to individual streams and scenes: does this increase in total rate happen at the expense of some less important streams or scenes suffering? The third constraint in Eq.(4.7) guarantees that this does not happen. In Table 4.3, we show that the average rate of individual scenes (calculated as the total amount of data transmitted during T divided by the scene duration) with video rate control is at least as high as with general rate control. In summary, the video rate control over a longer time interval T increases the total rate and utility, without hurting individual scenes. The magnitude of improvement depends on the video content and the choice of T.

Scenario 2: We now consider the same basic topology but now the nodes transmit two real video sequences, whose content is different and varies over time. We created two test video sequences by cropping and concatenating frames from standard video sequences as follows. First, we considered some standard video sequences, namely *Carphone, Foreman, Grandma,*

Table 4.4: Video Scenes (a concatenation of which is used to construct the test video sequences in Scenario 2).

Scene Number	**Original Video Sequence**	**Frame Number**	**Importance (MI)**
1	*Carphone*	171-230	4.45
2	*Carphone*	281-340	3.57
3	*Foreman*	144-203	2.56
4	*Grandma*	1-60	0.14
5	*Mother & Daughter*	101-160	0.19
6	*Mother & Daughter*	391-450	0.18

and Mother & Daughter. These were QCIF, encoded at 30 fps, using the JM 8.6 version of the H.264/AVC codec [78], [79]. The group of pictures consists of one I frame followed by 9 P frames. Each frame consists of at least one slice, packetized into an independent NAL unit of size 1000 B.

From these standard sequences, we selected six scenes summarized in Table 4.4: some of the scenes correspond to high motion and some to low motion parts of these sequences. Clearly, the six selected scenes have different video content and therefore different PSNR-rate characteristics. We encoded each scene with 50 different quantization parameters and obtained the corresponding distortion-rate (DR) curve. Then, we fitted this curve to the DR model developed in [89]: $D_e = \frac{\theta}{R_e - R_0} + D_0$, where D_e is the distortion of the encoded sequence, R_e is the output rate of video encoder and θ, R_0, D_0 are parameters of the model.

Then, we constructed our two test sequences by concatenating some of these six scenes. The first test sequence consists of the concatenation of *(scene 5, scene 1, scene 6, scene 2)*; the second test sequence consists of *(scene 3, scene 5, scene 3, scene 4)*. One option for assigning a utility $U(x)$ to each scene would be to simply use the PSNR value as a function of the rate from the aforementioned DR model. To further amplify the difference in the utilities of different scenes, we also multiplied the PSNR with a weight factor that indicates the importance of each scene, based on the content. As discussed earlier, there are many

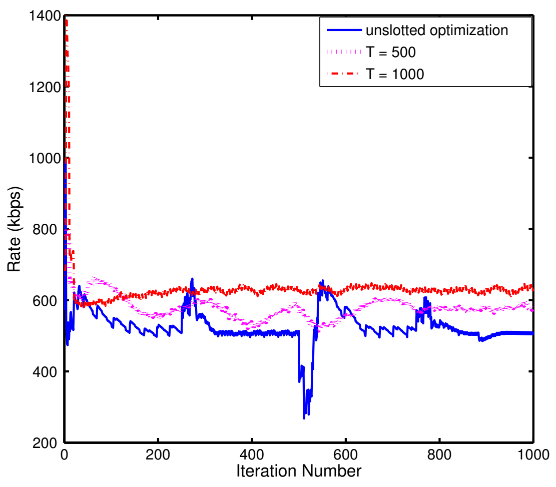


Figure 4.7: Scenario 2 (real sequences). Total rate achieved by the general (unslotted) and video (for two values of T) rate control schemes.

ways to assign importance to scenes. As a concrete example, we used the average motion intensity (MI) of each scene, as defined and computed in our prior work [86]: this way, the importance weights assigned to the scenes of the two test sequences are $(0.19, 4.45, 0.18, 3.57)$ and $(2.56, 0.19, 2.56, 0.14)$.

Finally, we simulated the transmission of these two test video sequences over the example topology, using the two rate allocation schemes developed in this section. The results are shown in Fig.4.7. The general rate control achieves the lower and variable total rate (shown in solid blue): the variations are triggered by the variation in content of the test sequences. The video rate control over an interval of $T = 500$ achieves rate that is higher and less variable (shown in dotted magenta). If we are willing to tolerate more delay and consider more scenes over the longer time interval $T = 1000$, we can achieve even higher and less variable total rate (shown in dash-dotted red line).

As expected, increasing T improves the total rate and thus the total utility. Fig.4.8 shows the increase in average PSNR for a range of values of T. For $T = 0$, the video rate control is essentially the general rate control, continuously computing the optimal rate at every

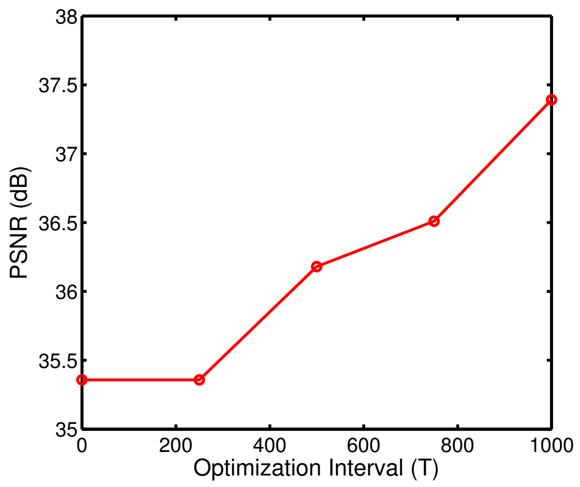


Figure 4.8: Scenario 2 (real sequences). Average PSNR vs. optimization interval T.

iteration. The exact shape of the utility vs. T curve depends on the content of the video, e.g., how its importance/utility varies over time. The curve in Fig.4.8 is specific to the test sequences and the A-B topology considered. In general, for a specific scenario, we can use this curve to select the right value of T so as to meet certain delay and utility requirements.

To summarize, we formulated rate allocation problem for video streaming over coded wireless networks and presented a distributed solution. Our key intuition was that time varying video rate and utility affect the network coding opportunities and the total achieved rate. To deal with time variability, we proposed video rate control over an appropriate time interval so as to optimize the utility vs. delay tradeoff. In the next section, we consider TCP flows over coded wireless networks. Similarly to video transmission, the time varying nature of TCP flows affect the achieved throughput over coded wireless networks and TCP should be aware of network coding to fully exploit network coding benefit.

4.4 Network Coding-Aware Queue Management

Over coded wireless networks, TCP flows flows do not fully exploit the network coding opportunities due to their bursty behavior and due to the fact that TCP is agnostic to the underlying network coding. Rate mismatch between flows can significantly reduce the coding opportunities, as there may not be enough packets from different flows at intermediate nodes to code together.

Example 8 The example shown in Fig. 4.9 illustrates the problem we consider. Since I can transmit $a \oplus b$ in one time slot, instead of a, b in two time-slots, network coding has the potential to improve throughput. However, if there is mismatch between the rates x_1, x_2 of the two flows, I may not have packets from the two flows to code together at all times, and thus does not exploit the full potential of network coding. We confirmed this intuition through simulations in this example topology. When the buffer size was set to 10 packets at each node and the bandwidth was $1Mbps$ for each link, we observed that 50% of the time, there were no packets from the two flows at the same time at node I to code together. For smaller queue sizes and larger transmission rates, there were even fewer coding opportunities. This means that there is potential for improvement by updating the protocols so as to mitigate the rate mismatch between TCP flows. This is the observation that motivates this paper. $\square$

One possible solution to this problem is to artificially delay packets at intermediate nodes [41], until more packets arrive and can be coded together. However, the throughput increases with small delay (due to more coding opportunities), but decreases with large delay (which reduces the TCP rate); the optimal delay depends on the network topology and the background traffic and also may change over time. Thus, in many practical networking scenarios, introducing delay at intermediate nodes is not practical.

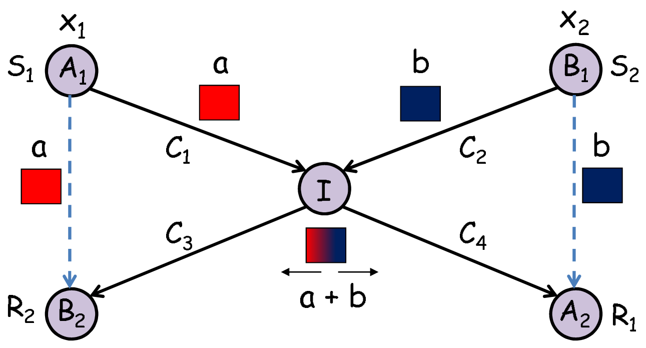


Figure 4.9: X topology. Source S_1 transmits a flow with rate x_1 to receiver R_1 and source S_2 transmits a flow with rate x_2 to receiver R_2, over the intermediate node I. A_1 and B_1 transmit their packets a and b, in two time slots, and node I receives them. Furthermore, A_2 overhears b and B_2 overhears a, because $A_1 - B_2$ and $B_1 - A_2$ are in the same transmission range and they can overhear each other. In the next time slot, I broadcasts the network coded packet, $a \oplus b$ over hyperarc $(I, \{A_2, B_2\})$. Since A_2 and B_2 have overheard b and a, they can decode their packets a and b, respectively.

We consider the same problem but we propose a different approach. Our main observation is that the mismatch between flow rates is due to the dynamic/bursty nature of TCP. Therefore, the problem can be eliminated by making modifications to congestion control mechanisms (at the end-points) and/or to queue management schemes (at intermediate nodes) to make them network coding-aware (in the sense that they can match the rates of flows coded together). Based on this observation, we take the following steps.

First, we formulate congestion control for unicast flows over wireless networks with inter-session network coding within the network utility maximization (NUM) framework similar to the previous sections. We consider the same system model presented in Section 4.1 and we assume that a known constructive network coding scheme is deployed in a wireless mesh network; examples include COPE [10] for one-hop network coding and BFLY [33] for two-hop network coding. The optimal solution of the NUM problem decomposes into several parts, each of which has an intuitive interpretation, such as rate control, queue management, and scheduling.

Second, motivated by the analysis, we propose modifications to congestion control mechanisms, so as to mimic the optimal solution of the NUM problem and to fully exploit the potential of network coding. It turns out that the optimal solution dictates minimal and intuitive implementation changes. We propose a network coding-aware queue management scheme at intermediate nodes (NCAQM), which stores coded packets and drops packets based on both congestion state and network coding. We note that the queues at intermediate nodes, which are already used for network coding, are a natural place to implement such changes with minimal implementation cost. In contrast, we do not propose any practical modifications to TCP or MAC (802.11) protocols, which significantly simplifies practical deployment of our proposal. Finally, we evaluate our proposal via simulation in GloMoSim [77] and we show that TCP over NCAQM significantly outperforms TCP over baseline schemes (*e.g.,* doubles the throughput improvement in some scenarios), and achieves near-optimal performance.

In the rest of this section, we first present the optimization problem and solutions in sub-section 4.4.1. Sub-section 4.4.3 presents simulation results. Sub-section 4.4.4 extends our framework to multi-hop network coding. Sub-section 4.4.5 presents numerical results for the convergence of the optimal solution.

4.4.1 Network Utility Maximization Formulation

Problem Formulation

The objective is to maximize the total utility function, by appropriately selecting: the flow rates x_s at sources $s \in \mathcal{S}$; their traffic splitting parameter $\alpha_h^{s,k}$ (following the terminology of [66]) into network codes $k \in \mathcal{K}_h$ over hyperarc h at intermediate nodes; and the percentage of time τ_h each hyperarc is used:

$$\begin{aligned}
&\max_{\boldsymbol{x},\boldsymbol{\alpha},\boldsymbol{\tau}} \sum_{s\in\mathcal{S}} U_s(x_s) \\
&\text{s.t.} \sum_{k\in\mathcal{K}_h} \max_{s\in\mathcal{S}_k}\{H_h^{s,k}\alpha_h^{s,k}x_s\} \le R_h\tau_h,\ \forall h\in\mathcal{A} \\
&\sum_{h(\mathcal{J})|h\in\mathcal{A}}\ \sum_{k\in\mathcal{K}_h|s\in\mathcal{S}_k} \alpha_h^{s,k} = 1,\ \forall s\in\mathcal{S}, i\in\mathcal{P}_s \\
&\sum_{h\in\mathcal{C}_q} \tau_h \le \tau,\ \forall \mathcal{C}_q \subseteq \mathcal{A}
\end{aligned} \tag{4.11}$$

The first constraint is the capacity constraint. $H_h^{s,k}\alpha_h^{s,k}x_s$ indicates the part of flow rate x_s allocated to the k-th network code over hyperarc h. The rate of the k-th network code is the maximum rate among flows $s\in\mathcal{S}_k$ coded together in code k: $\max_{s\in\mathcal{S}_k}\{H_h^{s,k}\alpha_h^{s,k}x_s\}$ [65]. Different network codes $k\in\mathcal{K}_h$ over h share the available capacity $R_h\tau_h$, where R_h is the transmission capacity of h; since h is a set of links, R_h is the minimum: $R_h = \min_{j\in h(\mathcal{J})}\{R_{i,j}\xi_{i,j}\}$ where $R_{i,j}$ is the capacity of link (i,j), and $\xi_{i,j}$ is the probability of successful transmission over link (i,j). The second constraint is the flow conservation constraint: at every node i on the path $\mathcal{P}_s$ of source s, the sum of $\alpha_h^{s,k}$ over all network codes and hyperarcs should be equal to 1. Indeed, when a flow enters a particular node i, it can be transmitted to its next hop j as part of different network coded and uncoded flows. The third constraint is due to interference. As mentioned, τ_h is the percentage of time h is used. Its sum over all hyperarcs in a clique should be less than an over-provisioning factor, $\gamma\le 1$, because all hypearcs in a clique interferes, and should time share the medium.

Solution

By relaxing the capacity constraint in Eq. (4.11), we get the Lagrangian:

$$L(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\tau}, \boldsymbol{q}) = \sum_{s \in \mathcal{S}} U_s(x_s) - \sum_{h \in \mathcal{A}} q_h \left(\sum_{k \in \mathcal{K}_h} \max_{s \in \mathcal{S}_k} \{ H_h^{s,k} \alpha_h^{s,k} x_s \} - R_h \tau_h \right) \tag{4.12}$$

where q_h is the Lagrange multiplier, which can be interpreted as the queue size at hyperarc h, as discussed later. To decompose the Lagrange function, we rewrite $\max_{s \in \mathcal{S}_k} \{H_h^{s,k} \alpha_h^{s,k} x_s\}$ as $\max_{m_h^{s,k}} \sum_{s \in \mathcal{S}_k} H_h^{s,k} \alpha_h^{s,k} x_s m_h^{s,k}$ *s.t.* $\sum_{s \in \mathcal{S}_k} m_h^{s,k} = 1$, where $m_h^{s,k}$ is a new variable, which we call the the dominance indicator. It indicates whether the source s has the maximum rate among all flows coded together in the k-th network code, or not. In the next section, we will see that only the dominant flow in a network code needs to back-off during congestion.

The Lagrange function in Eq. (4.12) is not strictly concave in $m_h^{s,k}$ and this causes oscillation in its solution. We use the proximal method [82] to eliminate oscillations;

$$\begin{aligned} \max_{\boldsymbol{m}} & \sum_{s \in \mathcal{S}_k} (H_h^{s,k} \alpha_h^{s,k} x_s m_h^{s,k} - c(m_h^{s,k} - \mu_h^{s,k})^2) \\ \text{s.t.} & \sum_{s \in \mathcal{S}_k} m_h^{s,k} = 1, \end{aligned} \tag{4.13}$$

where c is a constant and $\mu_h^{s,k}$ is an artificial variable of the proximal method [82]. Its value is set to $m_h^{s,k}$ periodically. Let $(m_h^{s,k})^*$ be the solution to this problem.

By rewriting the summation $\sum_{k \in \mathcal{K}_h} \sum_{s \in S_k}$ as $\sum_{s \in \mathcal{S}} \sum_{k \in K_h | s \in \mathcal{S}_k}$, the Lagrange function in Eq. (4.12) can be expressed as:

$$L(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\tau}, \boldsymbol{q}) = \sum_{h \in \mathcal{A}} q_h R_h \tau_h + \sum_{s \in \mathcal{S}} \left(U_s(x_s) - x_s \sum_{h \in \mathcal{A}} \sum_{k \in \mathcal{K}_h | s \in \mathcal{S}_k} q_h H_h^{s,k} \alpha_h^{s,k} (m_h^{s,k})^* \right). \tag{4.14}$$

Now, we can decompose the Lagrangian into the following intuitive problems: rate control, traffic splitting, scheduling, and parameter update (queue management).

Rate Control. First, we solve the Lagrangian w.r.t x_s:

$$x_s = (U_s')^{-1} \left(\sum_{h \in \mathcal{A}} \sum_{k \in \mathcal{K}_h | s \in \mathcal{S}_k} q_h H_h^{s,k} \alpha_h^{s,k} (m_h^{s,k})^* \right), \tag{4.15}$$

where $(U_s')^{-1}$ is the inverse function of the derivative of U_s. If we define $w_h^s = \sum_{k \in \mathcal{K}_h | s \in \mathcal{S}_k}$ $H_h^{s,k} \alpha_h^{s,k} (m_h^{s,k})^*$ and $q_{h(i)}^s = \sum_{h(\mathcal{J})|h \in \mathcal{A}} q_h w_h^s$, the rate x_s can be expressed as $x_s = (U_s')^{-1}$ $(\sum_{i \in \mathcal{P}_s} q_i^s)$, noting that $i = h(i)$.

In the special case where proportional fairness is desired, $U_s(x_s) = \log(x_s), \forall s \in \mathcal{S}$, leading to $x_s = \left(\sum_{i \in \mathcal{P}_s} q_i^s\right)^{-1}$, *i.e.*, x_s is inversely proportional to the total network coded queue sizes over the path of flows s, which we will be explained later.

Traffic Splitting. Second, we solve the Lagrangian for $\alpha_h^{s,k}$: at each node i along the path (*i.e.*, $i \in \mathcal{P}_s$), the traffic splitting problem can be expressed as

$$\begin{aligned} \min_{\boldsymbol{\alpha}} & \sum_{h(J)|h \in \mathcal{A}} \sum_{k \in K_h | s \in \mathcal{S}_k} q_h H_h^{s,k} (m_h^{s,k})^* \alpha_h^{s,k} \\ \text{s.t.} & \sum_{h(J)|h \in \mathcal{A}} \sum_{k \in K_h | s \in \mathcal{S}_k} \alpha_h^{s,k} = 1, \ \forall i \in \mathcal{P}_s \end{aligned} \tag{4.16}$$

Similarly to Eq. (4.13), we also use the proximal method [82] to solve the optimization problem in Eq. (4.16).

Scheduling. Third, we solve the Lagrangian for τ_h. This problem is solved for every hyperarc and every clique in the conflict graph in the hypergraph.

$$\begin{aligned} \max_{\boldsymbol{\tau}} & \sum_{h \in \mathcal{A}} q_h R_h \tau_h \\ \text{s.t.} & \sum_{h \in \mathcal{C}_q} \tau_h \leq \tau, \ \forall \mathcal{C}_q \subseteq A. \end{aligned} \tag{4.17}$$

Parameter (Queue Size) Update. We find q_h, using a gradient descent algorithm: $q_h(t+1) = \{q_h(t) + c_t[\sum_{k\in\mathcal{K}_h}\sum_{s\in\mathcal{S}_k} H_h^{s,k}\alpha_h^{s,k}(m_h^{s,k})^* x_s - R_h\tau_h]\}^+$. Equivalently;

$$q_h(t+1) = \{q_h(t) + c_t[\sum_{k\in\mathcal{K}_h}\max_{s\in\mathcal{S}_k}\{H_h^{s,k}\alpha_h^{s,k}x_s\} - R_h\tau_h]\}^+ \tag{4.18}$$

where t is the iteration number, c_t is a small constant, and the $^+$ operator makes the Lagrange multipliers positive. q_h can be interpreted as the queue size at hyperarc $\forall h \in \mathcal{A}$. Indeed, in Eq. (4.18), q_h is updated with the difference between the incoming $\sum_{k\in\mathcal{K}_h}\max_{s\in\mathcal{S}_k}\{H_h^{s,k}\alpha_h^{s,k}x_s\}$ and outgoing $R_h\tau_h$ traffic at h. Therefore, we call q_h the hyperarc-queue, or h-queue for brevity. We confirmed the convergence of q_h's via numerical calculations as seen in sub-section 4.4.5.

4.4.2 Network Coding-Aware Implementation (NCAQM)

In the previous sub-section, we saw that the NUM problem decomposed into Eq. (4.15), Eq. (4.16), Eq. (4.17), Eq. (4.18), each of which has an intuitive interpretation. Now, we mimic the properties of the optimal solutions to these problems and propose modifications to the corresponding protocols to make them network coding-aware. It turns out that only changes to queue management at intermediate nodes are crucial, while TCP and scheduling can remain intact. This makes our work amenable to practical deployment.

Summary of Proposed Scheme:

We refer to our Network Coding-Aware Queue Management scheme as NCAQM. NCAQM builds on and extends COPE [10]. Its goal is to interact with TCP congestion control in such a way that it matches the rates of TCP flows coded together and thus increases network coding opportunities. It achieves this goal through the following minimal changes at intermediate nodes. First, NCAQM stores coded packets in the output queue $\mathcal{Q}_i$, as opposed to COPE that stores uncoded packets. Second, NCAQM maintains state per hyperarc queue q_h and per network code transmitted over each hyperarc $k \in \mathcal{K}_h$; this is feasible in the setting of wireless mesh with limited number of flows. Third, during congestion, packets are dropped from the flow that has the largest number of packets, where this number is computed only over h-queues where the flow is dominant. Consider several flows coded together in the same code: the rate of the dominant flow is the rate of the code; and dropping from the dominant flow matches the rates, as desired. We note that intermediate nodes do already network coding operations and can be naturally extended to implement these changes.

Detailed Description of Proposed Scheme:

Maintaining Queues: In [10], a wireless node i stores all packets uncoded in a single output queue $\mathcal{Q}_i$ and takes decisions at every transmission opportunity about whether to code some of these packets together or not. In contrast, we propose to network code packets, if an opportunity exists, at the time we store them in the queue. Motivated by the fact that Lagrange multiplier (h-queue) q_h in Eq. (4.18) can be interpreted as the queue size at hyperarc h, we maintain h-queue virtually[4] for each hyperarc at every node, which keeps track of packets that are network coded and broadcast over h. The size of an h-queue is Q_h and how it is determined in practice will be explained later. Each node i maintains a single physical output queue, $\mathcal{Q}_i$, which stores all packets (coded and uncoded depending on the opportunities) passing through it.

Network Coding (Alg. 1): Motivated by the fact that the incoming traffic in Eq. (4.18) is the sum of the network coded flows over h, we code packets when they are inserted to output queues. If a network coding opportunity does not exist when the packet arrives at node i, we just store it in $\mathcal{Q}_i$ in a FIFO way. Periodically, Alg. 3 runs to check all packets in the

[4]We maintain a virtual, not a physical, h-queue, because the latter would be difficult in practice: (i) the total buffer size is limited and allocating it to h-queues is another control parameter; (ii) h-queues may change over time depending on changes in the topology and traffic scenario; (iii) storing packets in h-queues may reduce network coding opportunities in a packet-based system (although it is optimal in a flow-based system) due to opportunistic network coding.

Algorithm 3 Network coding in output queue Q_i at node i

```
 1: for $m = 1...L$ do
 2:     if $\exists p_m \in \mathcal{Q}_i$ then
 3:         for $n = (m+1)...L$ do
 4:             if $p_m \oplus p_n$ is eligible then
 5:                 $p_m \leftarrow p_m \oplus p_n$
 6:             end if
 7:         end for
 8:     end if
 9:     Update $\mathcal{Q}_i$
10: end for
```

queue for network coding.

Let $\mathcal{Q}_i = \{p_1, p_2, ..., p_l\}$ where p_1 is the first and p_l is the last packet in the queue; $l \leq L$, where L is the buffer size, *i.e.,* the maximum number of packets that can be stored in $\mathcal{Q}_i$. First, p_1 is picked for network coding. Since $\mathcal{Q}_i$ stores network coded packets, p_1 may be already coded. Independently of whether p_1 is network coded or not, it can be further coded with other packets in the queue beginning from p_2, if the following two conditions are satisfied; (i) the packets constructing p_1 and p_2 should be from different flows, and (ii) $p_1 \oplus p_2$ should be decodable at the next hop of all packets that construct the network code. If these conditions are satisfied, we say that the network code is an eligible network code, and p_1 is replaced by $p_1 \oplus p_2$. Then $p_1 \oplus p_3$ is checked for network coding, etc. After all packets are checked for network coding, the output queue $\mathcal{Q}_i$ is updated: (i) the final packet p_1 is stored in the first slot of the output queue, and (ii) the memory allocated to other packets are freed. Then, the same algorithm is run for packet p_2, etc. When a transmission opportunity arises, the first packet from the output queue is checked for network coding again and broadcast over the hyperarc.

Let the number of packets from flow s in node i be Q_i^s. Q_i^s captures the difference between the incoming and outgoing traffic for flow s at node i. Since an h-queue captures the difference between the incoming and outgoing traffic over a hyperarc, we calculate its size using the following heuristic: $Q_h = \sum_{k \in \mathcal{K}_h} \max_{s \in \mathcal{S}_k} \{H_h^{s,k} \breve{\alpha}_h^{s,k} Q_i^s\}$, where $\breve{\alpha}_h^{s,k}$ is the approximate traffic

Algorithm 4 Packet dropping at node i during congestion

```
 1: Initialization: $\Phi_i^s = 0$, $\forall s \in \mathcal{S}$, $\mathcal{S}_i^{'} = \emptyset$
 2: if $l > L$ then
 3:     for $\forall s \in \mathcal{S}$ do
 4:         Calculate $\Phi_i^s = \sum_{h(\mathcal{J})|h \in \mathcal{A}} Q_h \breve{w}_h^s$
 5:     end for
 6:     $\mathcal{S}_i^{'} = \arg\max_{s \in \mathcal{S}} \{\Phi_i^s\}$
 7:     Choose a flow $s' \in \mathcal{S}_i^{'}$ randomly
 8:     if $\exists p_n \in \mathcal{Q}_i$, $n = 1..l$, from flow $s'$ then
 9:         Drop $p_n$
10:     else
11:         Drop $p_l$
12:     end if
13: end if
```

splitting, explained next.

The traffic splitting parameters $\alpha_h^{s,k}$ are found through the optimization problem in Eq. (4.16). Through numerical calculations, we made the following observation: each $\alpha_h^{s,k}$ converges to the percentage of time that packets from flow s are transmitted with the k-th network code over h at node i. At each packet transmission, we calculate the probability that a network code k over hyperarc h can be used for flow s, over a time window. The average calculated over this window gives a heuristic estimate of the traffic splitting parameter, $\breve{\alpha}_h^{s,k}$.

Packet Dropping (Alg. 2): When a node is congested, it decides which packet to drop. In order to eliminate the potential of rate mismatch between flows coded together, we propose that the node compares the number of all (coded and uncoded) packets of each flow, in queues where the flow is dominant ($m_h^{s,k} = 1$). This is motivated by the optimal rate control in Eq. (4.15). More specifically, for each flow s, we calculate $\Phi_i^s = \sum_{h(\mathcal{J})|h \in \mathcal{A}} Q_h \breve{w}_h^s$, where $\breve{w}_h^s = \sum_{k \in \mathcal{K}_h | s \in \mathcal{S}_k}$ and $H_h^{s,k} \breve{\alpha}_h^{s,k} \breve{m}_h^{s,k}$. Upon congestion, the Φ_i^s's are compared and a packet from the flow with the largest Φ_i^s is dropped, preferably the last uncoded packet. The choice of the last packet is to make it similar to DropTail. The choice of uncoded packet is so as to hurt only one flow, as opposed to several. If there is a tie in the Φ's between flows, one flow is randomly picked to drop a packet. If all packets from the selected flow are coded, a new coming packet(s) is dropped instead.

To estimate the dominance indicator $\check{m}_h^{s,k}$ needed in Alg. 4, we compute heuristically an estimate $\check{m}_h^{s,k}$ as follows. If $H_h^{s,k}\check{\alpha}_h^{s,k}Q_i^s < H_h^{s',k}\check{\alpha}_h^{s',k}Q_i^{s'}$ s.t. $\exists s' \in \mathcal{S}_k - \{s\}$, then $\check{m}_h^{s,k} = 0$. Otherwise, $\check{m}_h^{s,k} = (|\mathcal{S}_k^{max}|)^{-1}$ where $\mathcal{S}_k^{max} = \{s | s \in \mathcal{S}_k \wedge H_h^{s,k}\check{\alpha}_h^{s,k}Q_i^s = \max\{H_h^{s',k}\check{\alpha}_h^{s',k}Q_i^s \mid s' \in \mathcal{S}_k\}\}$.

Rate Control at the Sources

For logarithmic utility, we saw that the optimal rate control in Eq. (4.15) is $x_s = (\sum_{i\in\mathcal{P}_s} q_i^s)^{-1}$. q_i^s corresponds to the length of the network coded queue size of flow s at node i. The optimal rate x_s is inversely proportional to the sum of these queue sizes q_i^s across all nodes i on its path $\mathcal{P}_s$. This is essentially a generalization of standard optimal rate control [59], to account for network coding in the calculation of queue sizes.

When rate control is implemented, it is impractical to feed back to the source the full information $\sum_{i\in\mathcal{P}_s} q_i^s$, as required by the optimal control. Instead, when a queue is congested, a packet is dropped or marked [59]. The source uses this binary information as a signal to reduce its rate, mimicking the inverse relationship in the optimal control. The exact adaptation of the flow rate depends on the TCP version used. In the simulations, we used TCP-SACK without any modification. The only change we propose is the packet dropping scheme at the queue (Alg. 2), to take into account not only congestion but also network coding. Essentially, TCP still reacts to drops but these drops are caused when the flow is dominant in at least one network coded queue along the path.

Example 9 Let us re-visit the example in Fig. 4.9. There is only one network coded flow over $h = (I, \{A_2, B_2\})$ and assume that link transmission rates are the same. Then the two flows are always coded together and their traffic splitting parameters approach to 1. The network coded queue sizes are $\Phi_I^1 = Q_h\check{m}_h^1$ and $\Phi_I^2 = Q_h\check{m}_h^2$, where Q_h is the size of the h-queue for $h = (I, \{A_2, B_2\})$, and $\check{m}_h^1$ and $\check{m}_h^2$ are the dominance indicators for the two flows.

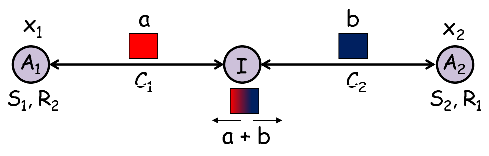
Since Q_h is constant, Φ_I^1, Φ_I^2 depend on $\check{m}_h^1$ and $\check{m}_h^2$, *i.e.,* on which flow has more packets in the output queue. Upon congestion, a packet from the first is dropped if it has more packets in the queue. Then, S_1 will reduce its rate by transmitting less packets, while flow S_2 keeps increasing its rate, thus decreasing the probability that there is no packet from the second flow for coding at node I. More generally, the interaction of our queue management (NCAQM) mechanism and TCP tends to eliminate the rate mismatch of the flows coded together. □

Scheduling

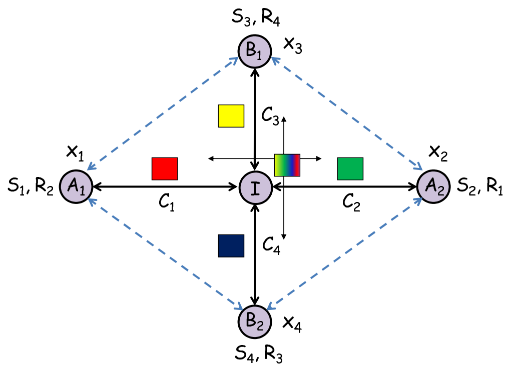
The scheduling part in Eq. (4.17) has two parts: intra- and inter-scheduling that determine which packet to transmit from a node and which node should transmit, respectively. Both have difficulties in practice. Intra-scheduling causes packet reordering at TCP receivers. Inter-scheduling requires centralized knowledge and it is NP hard and hard to approximate [60]. Given these difficulties and our original goal to make minimal changes to protocols related to congestion control, we limit our proposed modifications to the queue management. We do not propose new scheduling and we use FIFO scheme for packet transmission and standard 802.11 as wireless MAC.

4.4.3 Performance Evaluation

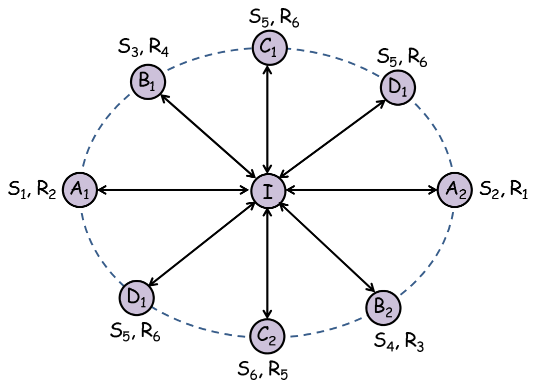
In this sub-section, we evaluate the throughput of TCP over our proposed scheme (NCAQM) in various topologies and traffic scenarios. We compare it to TCP over the following baseline schemes: no network coding (noNC), which uses FIFO without network coding; COPE [10], which stores native packets in a FIFO and decides which packets to code together at each transmission opportunity; and the optimal control.



(a) Alice-and-Bob Topology



(b) Cross Topology



(c) Wheel Topology

Figure 4.10: (a) Alice-and-Bob Topology. Two unicast flows, $S_1 - R_1$, and $S_2 - R_2$, meeting at intermediate node I. (b) Cross topology. Four unicast flows, $S_1 - R_1$, $S_2 - R_2$, $S_3 - R_3$, and $S_4 - R_4$, meeting at intermediate node I. (c) Wheel topology. Multiple unicast flows $S_1 - R_1$, $S_2 - R_2$, etc., meeting at intermediate node I. I opportunistically combine the packets and broadcast.

Simulation Setup

We used the GloMoSim simulator [77], which is well suited for wireless. We implemented from scratch the modules for one-hop network coding over wireless mesh networks (COPE) as well as for our proposed scheme (NCAQM).

Topologies We simulated four illustrative topologies shown in Fig. 4.9, Fig. 4.10, and Fig. 4.11. In X and Alice-and-Bob topologies, shown in Fig. 4.9 and Fig. 4.10(a), two

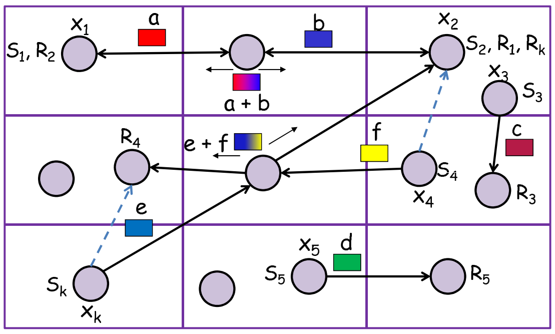


Figure 4.11: Grid topology. Multiple unicast flows $S_1 - R_1$, $S_2 - R_2$, etc., meeting at different intermediate points.

unicast flows $S_1 - R_1$ and $S_2 - R_2$ meet at intermediate node I. In the cross topology, shown in Fig. 4.10(b), four unicast flows $S_1 - R_1$, $S_2 - R_2$, $S_3 - R_3$, and $S_4 - R_4$ are transmitted via the relay I. In the wheel topology, shown in Fig. 4.10(c), multiple unicast flows such as $S_1 - R_1$, $S_2 - R_2$, $S_3 - R_3$, $S_4 - R_4$, and etc. are combined at the intermediate node I. Note that the wheel topology is the generalized version of the cross topology shown in Fig. 4.10(b). In all these topologies node I; (i) performs network coding, and (ii) is placed in the center of a circle with $90m$ radius over $200m \times 200m$ terrain and all other nodes are placed around the circle. Finally, we considered the grid topology shown in Fig. 4.11, in which nodes are distributed over a $300m \times 300m$ terrain, divided into 9 cells of equal size. 15 nodes are divided into sets consisting of 1 or 2 nodes and each set is assigned to a different cell. Nodes in a set are randomly placed within their cell. If both the transmitter and the receiver are in the same cell or in neighboring cells, there is a direct transmission; otherwise, a node in a neighboring cell acts as a relay. If there are more than one neighboring cells, one is chosen at random. In all topologies, a single channel is used for both uplink and downlink transmissions.

MAC: In the MAC layer, we simulated IEEE 802.11 with RTS/CTS enabled and with the following modifications for network coding. First, we need a broadcast medium, which is hidden by the 802.11 protocol. We used the pseudo-broadcasting mechanism of [10]: packets are XOR-ed in a single unicast packet, an XOR header is added for all nodes that should receive that packet, and the MAC address is set to the address of one of the receivers. A receiver knows whether a packet is targeted to it from the MAC address or the XOR header.

Wireless Channel: We used the two-ray path loss model and Rayleigh fading in Glomosim. We set the average loss rate to 15%. In our simulations 15% loss rate is medium loss rate, and residual loss rate after MAC re-transmissions is less than 1%.[5]

TCP Traffic We consider FTP/TCP traffic on top of the wireless network. In the Alice-and-Bob, X, cross, and wheel topologies, TCP flows, between the pairs of nodes described above, start at random times within the first $5sec$ and live until the end of the simulation. In the grid topology, TCP flows arrive according to a Poisson distribution with average 6 flows per $30sec$. The sender and the receiver of a TCP flow are chosen randomly. If the same node is chosen, the random selection is repeated.

Simulation Results

In this section, we present simulation results for the Alice-and-Bob, X, cross, wheel, and grid topologies. We compare to: (i) TCP over NCAQM (TCP+NCAQM), (ii) TCP over COPE (TCP+COPE), (iii) the optimal solution (optimal rate control in Eq. (4.15) working together with the optimal queue management in Eq. (4.18)). We report the average throughput of each scheme as % improvement over the throughput of the baseline TCP+noNC. In addition, we report transport level throughput. All throughput results reported in this section are averaged over $1min$ simulation duration first, then over 10 simulations with different seeds.

Table 4.5 presents the results for the following parameters: the buffer size at each interme-

[5]When channel loss rate increases, there are two problems. First, the residual loss rate after MAC re-transmissions increases. Therefore, TCP is not able to utilize the medium effectively and benefit of network coding reduces. Second, network coding decision at intermediate nodes becomes erroneous, because intermediate nodes do not know which packets are overheard correctly. These issues are out of scope of this chapter, and we have analyzed them separately in the next chapter.

diate node is 10 packets[6]; the packet size is $500B$; the channel capacity is $1Mbps$. In this scenario, TCP+NCAQM has two advantages: (i) it stores network coded packets instead of the uncoded ones, thus uses the buffer more effectively, and (ii) it drops packets so that network coding opportunities increase. Thus, our scheme (TCP+NCAQM) significantly improves throughput as compared to TCP+COPE in all four topologies. It is also seen from the table that there is still a gap between our scheme and the optimal improvement due to the very limited buffer size for multiple flows at the relay. Yet, even in this challenging scenario, TCP+NCAQM significantly improves over TCP+COPE: it doubles the throughput improvement of TCP+COPE.

In Table 4.5, the improvement of TCP+NCAQM and TCP+COPE in Alice-and-Bob topology is slightly smaller as compared to X topology, although Alice-and-Bob and X topologies have the same optimal improvement (33%). In Alice-and-Bob topology, source nodes are also receiver nodes, *i.e.*, S_1-R_2 and S_2-R_1 pairs are the same nodes; A_1, A_2, respectively. Therefore, transport level data and ACK packets share the same buffers at these source/receiver nodes. Due to the limited buffer size, some packets are dropped at the source/receiver nodes, and this reduces TCP throughput. It is also seen that the improvement in cross and grid topologies is larger as compared to Alice-and-Bob and X topologies, for the following reasons: (i) in cross topology, four flows (*i.e.*, four packets) are combined at the intermediate node (I) instead of two flows, and (ii) in grid topology, we have observed that, during a part of $1min$ simulation duration, four or more flows are combined at intermediate nodes.

Fig. 4.12 presents the cumulative distributed function (CDF) of throughput improvement for the Alice-and-Bob, X, cross, and grid topologies and the same setup. The CDFs are calculated over 30 seeds. One can see that the CDF of TCP+NCAQM is shifted to significantly higher throughput levels compared to TCP+COPE in all four topologies. For example, TCP+NCAQM improves the throughput more than 20% and 40% in more than 60% of the

[6]Note that 10 packet buffer size corresponds to bandwidth-delay product (BDP) in our simulation scenario. We also present simulation results for larger buffer sizes later in this section.

Table 4.5: Average throughput improvement compared to noNC.

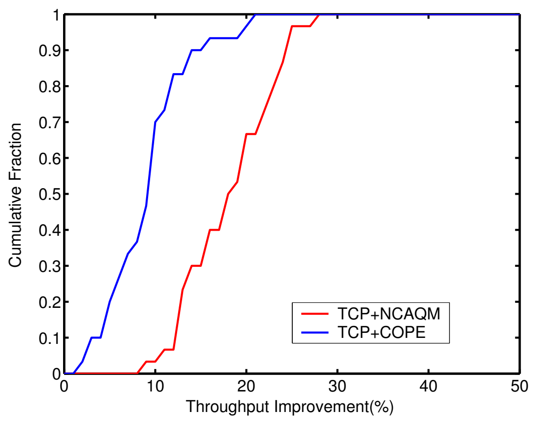
	Optimal	TCP+NCAQM	TCP+COPE
Alice-and-Bob Topology	33%	18%	8%
X Topology	33%	19%	9%
Cross Topology	60%	39%	21%
Grid Topology	-	35%	18%

realizations in Alice-and Bob and cross topologies, respectively. In contrast to Alice-and-Bob and cross topologies, we also observe that the CDF of TCP+NCAQM is shifted to higher throughput levels compared to the CDF of TCP+COPE in the cross and grid topologies. In the cross and grid topologies, it is possible to code more than two flows together, and when the number of flows coded together increases, the way that TCP+NCAQM uses buffers and balances the rates becomes more important. Thus, we see larger improvement in the cross and grid topologies.

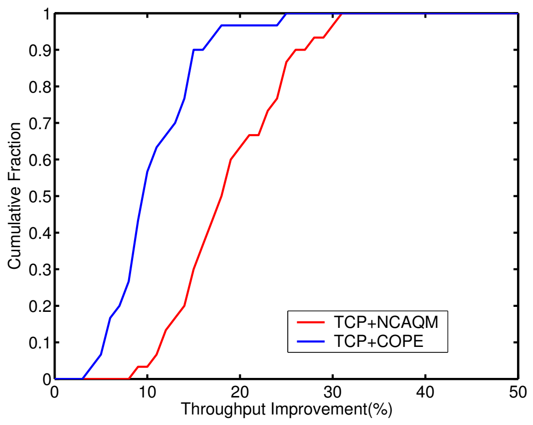
Fig. 4.13 shows the average transport-level throughput versus the buffer size, for the Alice-and-Bob, X, cross, and grid topologies. Packet size is $500B$, and channel capacity is $1Mbps$. Our observations from Fig. 4.13 are in the following.

The throughput improvement of TCP+noNC for different buffer sizes is negligible in all topologies. The reason is that 10 packet buffer size is already matched to bandwidth-delay product (BDP) and TCP utilizes wireless medium effectively for almost all buffer sizes when network coding is not used (TCP+noNC). However, for network coding schemes (TCP+NCAQM and TCP+COPE), the throughput increases significantly with increasing buffer size. This shows the importance of active queue management in coded networks.

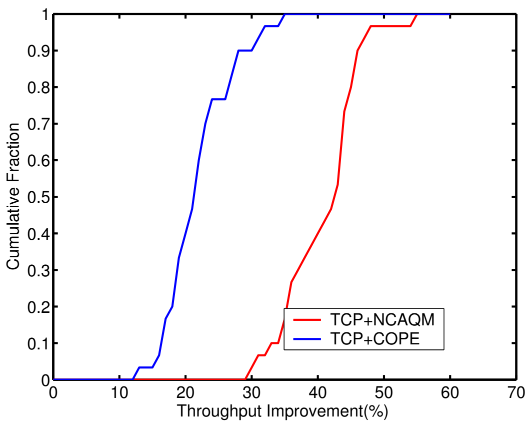
When buffer sizes are small, the improvement of TCP+NCAQM over TCP+noNC is significantly larger than that of TCP+COPE. This is for the same reason explained earlier: TCP+NCAQM stores network coded, instead of uncoded packets, thus using buffer more effectively, and it drops packets so that network coding opportunities increase. Thus, our



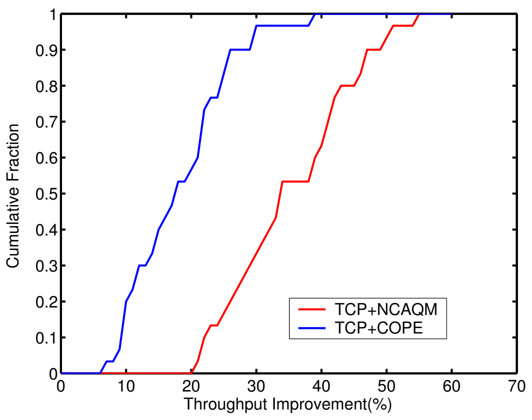
(a) Alice-and-Bob topology



(b) X topology



(c) Cross topology



(d) Grid topology

Figure 4.12: Cumulative distribution function (CDF) of throughput improvement for Alice-and-Bob (shown in Fig. 4.10(a)), X (shown in Fig. 4.9), cross (shown in Fig. 4.10(b)), and grid (shown in Fig. 4.11) topologies. Buffer size is 10 packets, packet sizes are $500B$, and the channel capacity is $1Mbps$. The distributions are generated over 30 seeds.

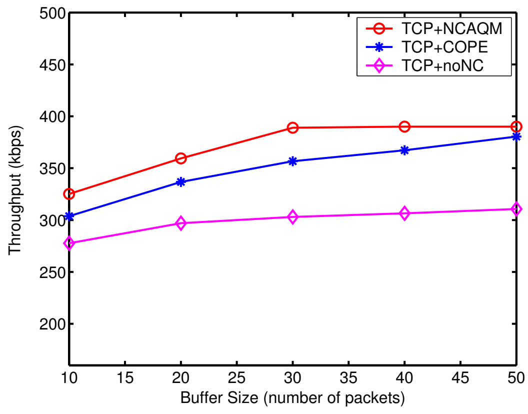
scheme (TCP+NCAQM) significantly improves throughput as compared to TCP+COPE in all four topologies.

The throughput of TCP+COPE increases when buffer sizes increase, which is intuitively expected. The problem addressed in this section was the mismatch between rates of flows coded together, due to the bursty nature of TCP, which reduces coding opportunities. However, when buffer sizes increase, there are more packets available in queues for coding. Thus, TCP+COPE exploits coding opportunities at larger buffers and its throughput increases. However, even at the large buffer sizes, TCP+NCAQM improves throughput more than TCP+COPE. For example, TCP+NCAQM improves throughput 7% more than TCP+COPE in X topology when buffer size is 50 packets. Fig. 4.13 demonstrates that our scheme is particularly beneficial in harsh buffer size conditions.

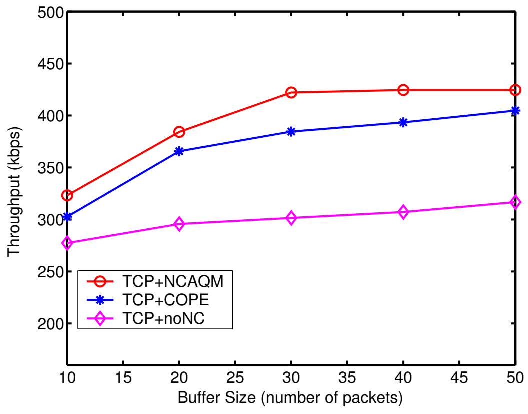
The improvement of TCP+NCAQM over TCP+noNC exceeds the optimal throughput at some buffer sizes. *E.g.*, the improvement of TCP+NCAQM over TCP+noNC is around 40% in the X topology when the buffer size is set to 30 packets (although the optimum improvement is 33%). The reason is that since TCP+NCAQM uses the buffer more effectively by storing network coded packets instead of uncoded packets, TCP can utilize the medium more effectively, thus the TCP rate increases beyond the network coding benefit.

Fig. 4.14 shows the average transport-level throughput versus the number of flows in the wheel topology shown in Fig. 4.10(c). The buffer size is 30 packets, the packet size is $500B$, and channel capacity is $1Mbps$. One can see from the figure that the throughput of TCP+noNC reduces with increasing number of flows. This is expected, because when the number of flows increases, all flows share the same queue at the intermediate node I. As a result, the round trip time of each flow increases, and thus the TCP rate decreases. On the other hand, the throughput of TCP+NCAQM and TCP+COPE increases with the number of flows, because when the number of flows increases, there are more network coding opportunities and more packets can be combined together (*i.e.*, it is possible to combine 8 packets when the number of flows is 8). TCP+NCAQM significantly improves over TCP+COPE for all number of flows, especially when the number of flows is large. This is intuitive, because when the number of flows increases, network coding opportunities increases, and TCP+NCAQM exploits these opportunities effectively.

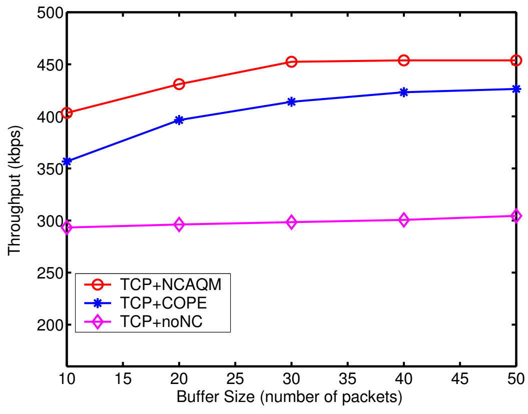
Fig. 4.15 presents the average transport-level throughput versus channel capacity for the Alice-and-Bob, X, cross, and grid topologies. The buffer size is 30 packets, and the packet



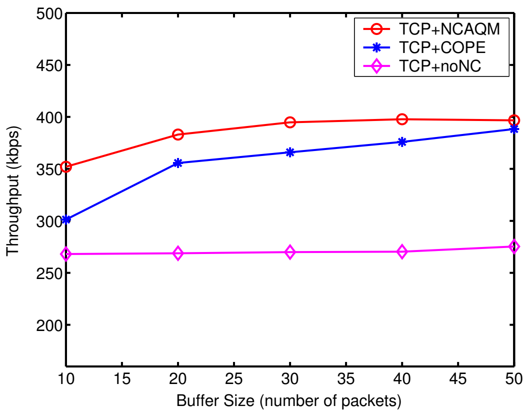
(a) Alice-and-Bob topology



(b) X topology



(c) Cross topology



(d) Grid topology

Figure 4.13: Average throughput (averaged in $1min$ simulation first, then over 10 seeds) versus buffer size for Alice-and-Bob (shown in Fig. 4.10(a)), X (shown in Fig. 4.9), cross (shown in Fig. 4.10(b)), and grid (shown in Fig. 4.11) topologies. Packet size is $500B$, and channel capacity is $1Mbps$.

size is $500B$. One can see from the figure that when the channel capacity increases, the gap between TCP+NCAQM and TCP+COPE increases. Therefore, while the improvement of TCP+NCAQM over TCP+noNC increases with increasing channel capacity, it decreases for TCP+COPE. Namely, the improvement of TCP+NCAQM increases from 40% to 42%, while the improvement of TCP+COPE decreases from 27% to 16% in X the topology. The improvement of TCP+NCAQM is quite significant; more than double the improvement of TCP+COPE at $11Mbps$ channel capacity. The reason is that when the channel capacity

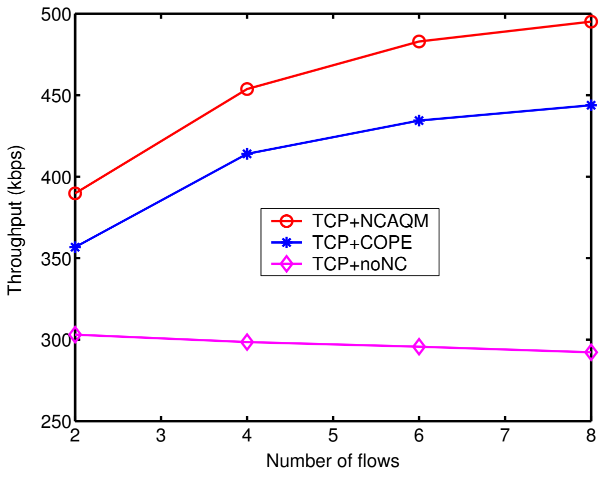


Figure 4.14: Average throughput (averaged in $1min$ simulation first, then over 10 seeds) versus the number of flows in wheel topology shown in Fig. 4.10(c). Buffer size is 30 packets, packet size is $500B$, and the channel capacity is $1Mbps$.

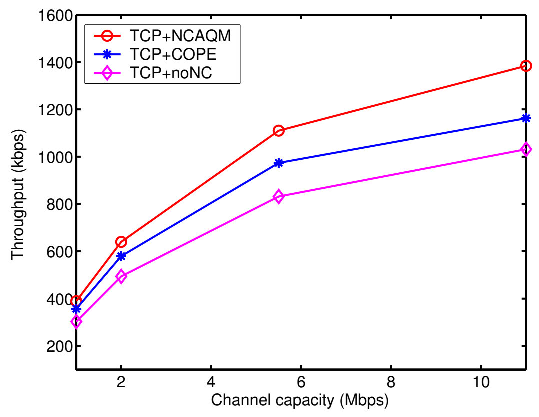
increases, more packets share buffer at intermediate node. TCP+NCAQM can improve the throughput by using the shared buffers more effectively, and by dropping packets so as to increase network coding opportunities.

4.4.4 Multi-Hop Network Coding

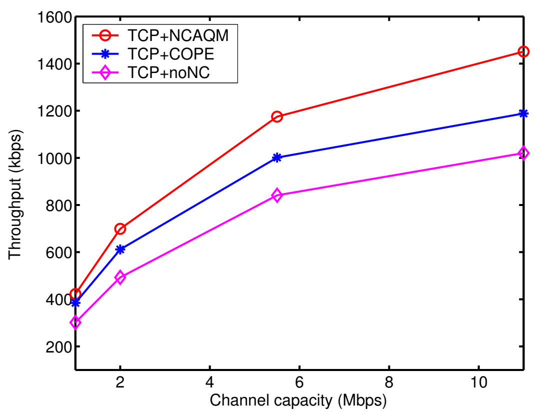
In this sub-section, we extend our framework from one-hop to multi-hop network coding. We note that our framework can accommodate any given multi-hop network coding scheme, but we use BFLY [33] in our simulations, as an example.

System Model

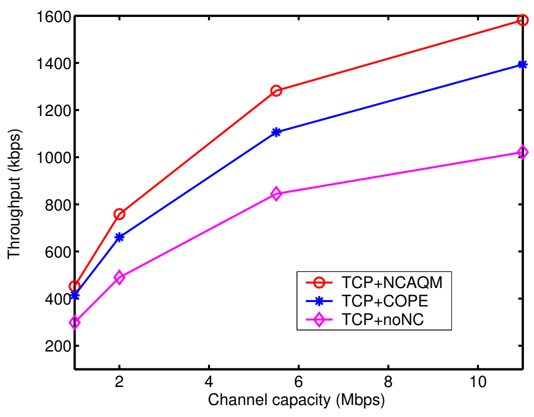
We consider the same system model as in Section 4.1, with the difference of multi-hop, as opposed to one-hop, network coding. A flow s can be network coded and decoded several times over its path $\mathcal{P}_s$. The network coded flow may be transmitted over multiple (M) hops, which we call M-hop network coding. M-hop network coding is implemented by COPE [10]



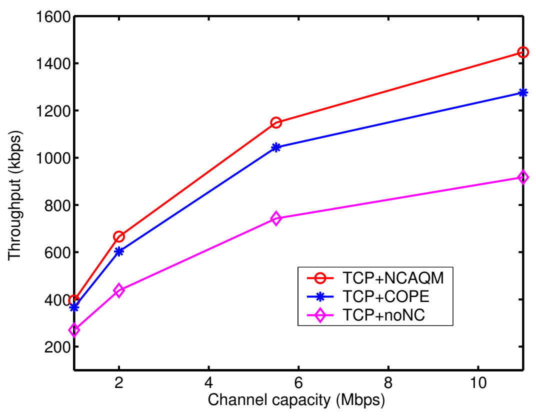
(a) Alice-and-Bob topology



(b) X topology



(c) Cross topology



(d) Grid topology

Figure 4.15: Average throughput (averaged in $1min$ simulation first, then over 10 seeds) versus channel capacity for Alice-and-Bob (shown in Fig. 4.10(a)), X (shown in Fig. 4.9), cross (shown in Fig. 4.10(b)), and grid (shown in Fig. 4.11) topologies. Buffer size is 30 packets, and packet size is $500B$.

for $M = 1$, BFLY [33] for $M = 2$, or other network coding schemes for $M > 2$. We assume that a flow s cannot be network coded if it (or a part of it) is already coded. This assumption allows us to divide the path $\mathcal{P}_s$ to F_s intermediate paths which we call network coding paths. Over its f-th network coding path, where $f \in \{1, \dots, F_s\}$, flow s can be network coded with $\Gamma_f^s \in \{0, 1, \dots, |\mathcal{S} - \{s\}|\}$ other flows. Without loss of generality, we can assume that a flow may be transmitted over the f-th network coding path without network coding; *i.e.*, $\Gamma_f^s = 0$. A flow s can be divided into network coded and non-network coded parts over a network

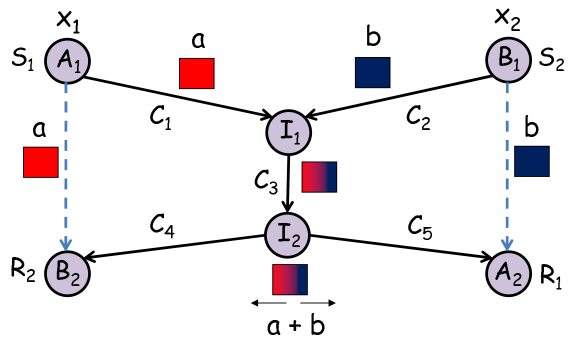


Figure 4.16: Butterfly topology. Source S_1 transmits a flow with rate x_1 to receiver R_1 and source S_2 transmits a flow with rate x_2 to receiver R_2, over the intermediate nodes I_1 and I_2. Nodes A_1 and B_1 transmit their packets a and b, in two time slots, and node I_1 receives them. Node B_2 overhears a and A_2 overhears b, because $A_1 - B_2$ and $B_1 - A_2$ are in the same transmission range and they can overhear each other. In the next time slot, I_1 transmits the network coded packet $a \oplus b$ to node I_2. Finally, I_2 broadcast $a \oplus b$ over hyperarc $(I_2, \{A_2, B_2\})$. Since A_2 and B_2 have overheard b and a, they can decode their packets a and b, respectively.

coding path f, where $\mathcal{Z}_s^f$ is the set of partitions of flow s over its f-th network coding path. Each partition $z \in \mathcal{Z}_s^f$ transmitted over hyperarc h has one-to-one mapping with the k-th network code over h such that $k \in \mathcal{K}_h$, *i.e.*, $z = \eta(k)$ over h where η is an injective function.

Example 10 The example shown in Fig. 4.16 illustrates the problem with 2-hop network coding. The flow from source S_1 is transmitted over the link $A_1 - I_1$ without network coding and it is network coded over the links $I_1 - I_2$ and $I_2 - A_2$. Over the network coding path, including the set of nodes I_1, I_2, A_2, the flow rate x_1 is partitioned into a network coded and a non-network coded part. The network coded part is combined with the corresponding part of the flow from source S_2, transmitted over $I_1 - I_2$, and broadcast over $(I_2, \{A_2, B_2\})$. The other part is transmitted over $I_1 - I_2$ and $I_2 - A_2$ without network coding. Similar to the one-hop network coding in Example 8, if there is a mismatch between the rates x_1, x_2 of the two flows, network coding benefit is not fully exploited. The goal is to solve this problem, assuming a given multi-hop network coding scheme. $\square$

Problem Formulation

We consider the following NUM problem;

$$
\begin{aligned}
&\max_{\boldsymbol{x},\boldsymbol{\alpha},\boldsymbol{\tau}} \sum_{s\in\mathcal{S}} U_s(x_s) \\
&\text{s.t.} \sum_{k\in\mathcal{K}_h} \max_{s\in\mathcal{S}_k}\{H_h^{s,k}\alpha_h^{s,k}x_s\} \le R_h\tau_h,\ \forall\ h\in\mathcal{A} \\
&\sum_{z\in\mathcal{Z}_s^f} \beta_f^{s,z} = 1,\ \forall\ s\in\mathcal{S}, f=1,...,F_s \\
&\alpha_h^{s,k} = \begin{cases} \beta_f^{s,z}, & \exists\ z=\eta(k), z\in\mathcal{Z}_s^f, f=1,...,F_s \\ 0, & \text{otherwise.} \end{cases} \\
&\sum_{h\in\mathcal{C}_q} \tau_h \le \tau,\ \forall\ \mathcal{C}_q\subseteq\mathcal{A}
\end{aligned}
\tag{4.19}
$$

The NUM problem in Eq. (4.19) is similar to the one in Eq. (4.11), in terms of the objective functions, capacity and interference constraints. We only need to update the flow conservation constraint (the second constraint) and add the third constraint, as explained below.

We introduce a new traffic splitting parameter $\beta_f^{s,z}$ which represents the percentage of the flow rate x_s allocated to the z-th partition of flow s over its f-th network coding path. The traffic splitting parameters should sum up to 1 according to the flow conservation constraint over each network coding path (the second constraint). Since there is a one-to-one mapping between the z-th partition and the k-th network code over h, the traffic splitting parameters, $\alpha_h^{s,k}$ and $\beta_f^{s,z}$ should be equal (the third constraint). This also implies the following equalities; $H_h^{s,k} = H_h^{s,z}$, $m_h^{s,k} = m_h^{s,z}$.

Solution

We use Lagrangian relaxation to solve the optimization problem in Eq. (4.19) by relaxing the capacity constraint with Lagrange multipliers q_h. We obtain the same Lagrange function in Eq. (4.12). The Lagrange function is decomposed into the same subproblems as in Eq. (4.13), Eq. (4.15), Eq. (4.17) and Eq. (4.18). The only different subproblem is the traffic splitting problem, which can be expressed as

$$\begin{aligned}
&\min_{\boldsymbol{\alpha}} \sum_{h\in\mathcal{A}} \sum_{k\in\mathcal{K}_h|s\in\mathcal{S}_k} q_h H_h^{s,k} \alpha_h^{s,k} (m_h^{s,k})^* \\
&\text{s.t.} \sum_{z\in\mathcal{Z}_s^f} \beta_f^{s,z} = 1, \ \forall\ s\in\mathcal{S}, f=1,...,F_s \\
&\qquad \alpha_h^{s,k} = \begin{cases} \beta_f^{s,z}, & \exists\ z=\eta(k), z\in\mathcal{Z}_s^f, f=1,...,F_s \\ 0, & \text{otherwise.} \end{cases}
\end{aligned} \tag{4.20}$$

The objective function in Eq. (4.20) can be expanded to be

$$\sum_{f=1}^{F_s} \sum_{h\in\mathcal{A}^f} \sum_{k\in\mathcal{K}_h|s\in\mathcal{S}_k} q_h H_h^{s,k} \alpha_h^{s,k} (m_h^{s,k})^*,$$

where $\mathcal{A}^f$ is the set of hyperarcs that originate from the nodes in the f-th network coding path of flow s. The two objective functions are equivalent considering the fact that the objective function in Eq. (4.20) is equal to zero for hyperarcs which are not originated from the nodes over the flow's network coding paths, because the indicator functions ($H_h^{s,k}$) are zero for those hyperarcs.

Now, let $\mathcal{Z}_s^{f,h}$ represent the set of partitions of the flow s over h in its f-th network coding path. Then, $\sum_{k\in\mathcal{K}_h|s\in\mathcal{S}_k}$ and $\sum_{z\in\mathcal{Z}_s^{f,h}}$ are equivalent, due to the one-to-one mapping between the z-th partition and k-the network code over h. Usage of $\sum_{z\in\mathcal{Z}_s^{f,h}}$ instead of $\sum_{k\in\mathcal{K}_h|s\in\mathcal{S}_k}$

implies the following changes; $\alpha_h^{s,k} = \beta_f^{s,z}$, $H_h^{s,k} = H_h^{s,z}$, and $m_h^{s,k} = m_h^{s,z}$. Then, the problem reduces to

$$\begin{aligned}
&\min_{\boldsymbol{\beta}} \sum_{f=1}^{F_s} \sum_{h \in \mathcal{A}^f} \sum_{z \in \mathcal{Z}_s^{f,h}} q_h H_h^{s,z} \beta_f^{s,z} (m_h^{s,z})^* \\
&\text{s.t.} \sum_{z \in \mathcal{Z}_s^f} \beta_f^{s,z} = 1, \ \forall \ s \in \mathcal{S}, f = 1, ..., F_s
\end{aligned} \tag{4.21}$$

The objective in Eq. (4.21) is expressed as $\sum_{f=1}^{F_s} \sum_{z \in \mathcal{Z}_s^f} \sum_{h \in \mathcal{A}^{f,z}} q_h H_h^{s,z} \beta_f^{s,z} (m_h^{s,z})^*$ where $\mathcal{A}^{f,z}$ which is the subset of $\mathcal{A}^f$ contains the hyperarcs over which the z-th partition of the f-th network coding path of flow s is transmitted. The two objective functions are equivalent, because the indicator functions $(H_h^{s,k})$ are zero for $h \notin \mathcal{A}^{f,z}$. Finally, the traffic splitting problem for $s \in \mathcal{S}, f = 1, ..., F_s$ is expressed as

$$\begin{aligned}
&\min_{\boldsymbol{\beta}} \sum_{z \in \mathcal{Z}_s^f} \beta_f^{s,z} \left(\sum_{h \in \mathcal{A}^{f,z}} q_h H_h^{s,z} (m_h^{s,z})^* \right) \\
&\text{s.t.} \sum_{z \in \mathcal{Z}_s^f} \beta_f^{s,z} = 1, \ \forall \ s \in \mathcal{S}, f = 1, ..., F_s
\end{aligned} \tag{4.22}$$

Similar to what we have done to solve Eq. (4.16), we use the proximal method [82] to solve this problem.

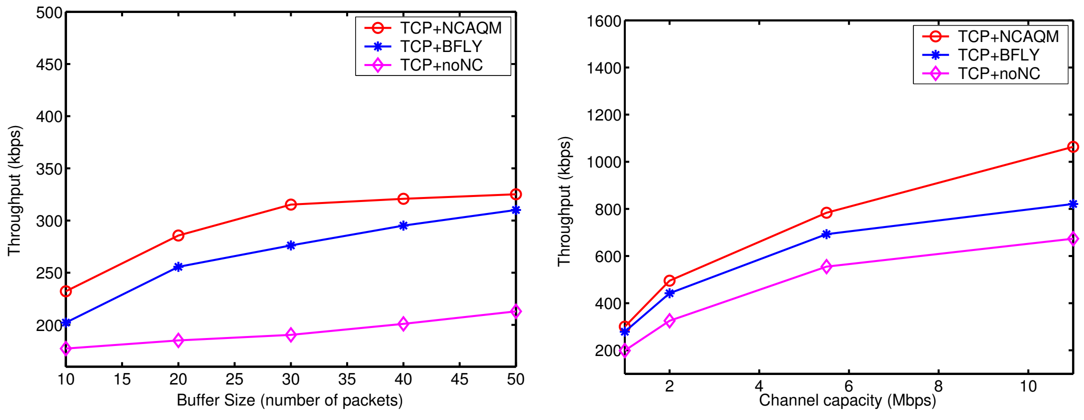
Simulation Results

We evaluate the throughput of TCP over NCAQM compared to TCP over the following baseline schemes: no network coding (noNC), which uses FIFO without network coding; BFLY [33], which utilizes knowledge of the local topologies by exchanging periodic messages that includes neighbors of nodes and source route information in the packet headers to exploit butterfly structures in wireless mesh networks. Similarly to COPE, BFLY stores native packets in a FIFO and decides which packets to code together at each transmission opportunity. We used the GloMoSim simulator [77] to implement the modules for two-hop network coding over wireless mesh networks (BFLY) as well as for our proposed scheme (NCAQM).

We simulate the butterfly topology shown in Fig. 4.16 in which two unicast flows S_1, R_1 and S_2, R_2 meet at intermediate node I_1. In this topology, nodes are placed over $300m \times 300m$ in butterfly like structure and a single channel is used for both uplink and downlink transmissions. We consider the same MAC update and wireless channel model as in Section 4.4.3. We consider FTP/TCP traffic over the wireless network. TCP flows, between the pairs of nodes described above, start at random times within the first $5sec$ and live until the end of the simulation.

Fig. 4.17(a) presents the average transport-level throughput vs. buffer size. Similarly to the simulation results in Section 4.4.3, TCP+NCAQM improves throughput much more than TCP+BFLY. Specifically, when buffer size is 10 packets, the improvement of TCP+BFLY over TCP+noNC is 13%, the improvement of TCP+NCAQM over TCP+noNC is 30%, while the optimum improvement is 50%. When buffer size increases, we see that TCP+NCAQM approaches and exceeds the optimum; *e.g.*, the improvement of TCP+NCAQM is 65% when buffer size is 30 packets, while it is 45% for TCP+BFLY. This shows that the advantages of TCP+NCAQM also apply to two-hop network coded wireless mesh networks.

Fig. 4.17(a) presents the average transport-level throughput vs. channel capacity. We can see that the improvement of TCP+NCAQM is larger than TCP+BFLY for all channel capacities and it is especially significant for large channel capacities, since TCP+NCAQM uses buffer more effectively and drops packets so that network coding opportunities increase.



(a) Buffer Size (b) Channel capacity

Figure 4.17: Average throughput (averaged in $1min$ simulation first, then over 10 seeds) in butterfly topology shown in Fig. 4.16. (a) Buffer size: packet size is $500B$, and the channel capacity is $1Mbps$. (b) Channel capacity: buffer size is 30 packets, and packet size $500B$.

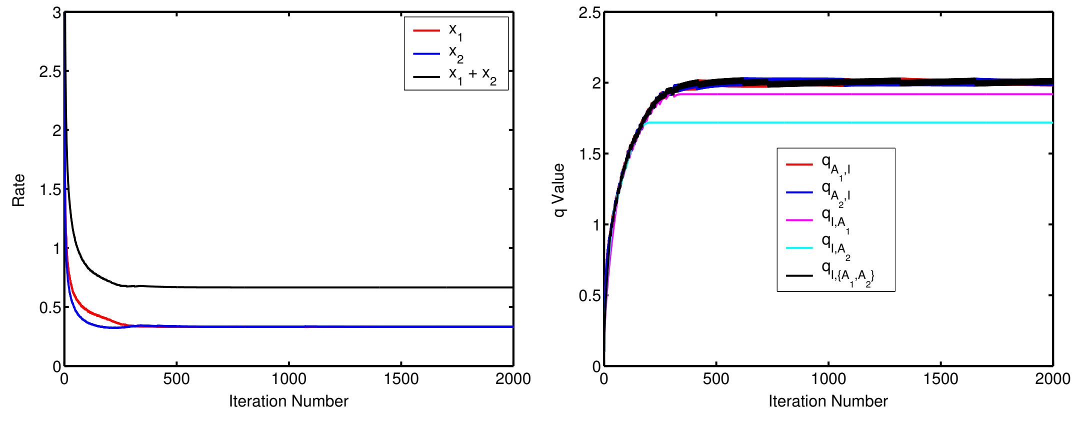
4.4.5 Numerical Results: Convergence

In this sub-section, we present numerical results that demonstrate the convergence of the solutions of NUM problems for one-hop and multi-hop network coding.

One-hop Network Coding

First, we consider the Alice-and-Bob topology presented in Fig. 4.10(a). We consider two cases for wireless channel capacities: (i) $C_1 = C_2 = 1$ units/transmission[7], and (ii) $C_1 = 1$, $C_2 = 4$. For the first case, the convergence of rates x_1, x_2, and $x_1 + x_2$ is presented in Fig. 4.18(a). One can see that the total rate $x_1 + x_2$ converges to 0.66 which is the optimal achievable rate when network coding is used for this scenario. Note that total achieved throughput is 0.50 for this scenario when network coding is not used. For the second case, it is seen in seen in Fig. 4.19(a) that the total throughput approaches to the

[7]We omit the units in the rest of the section for brevity.



(a) Rate (b) Lagrange multipliers

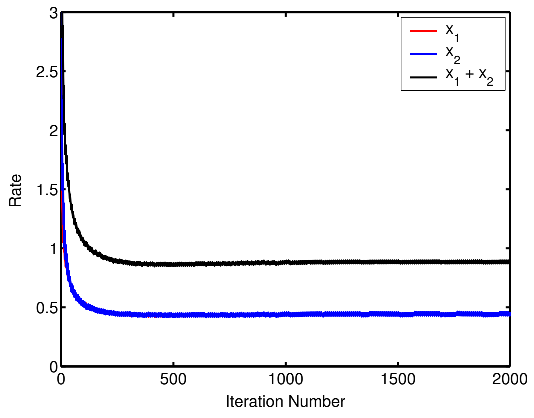
Figure 4.18: Convergence results for the Alice-and-Bob topology presented in Fig. 4.10(a). The total achieved rate approaches the optimum throughput 0.66. The optimum throughput is 0.50 when there is no network coding. $C_1 = C_2 = 1$.

optimal achievable rate of 0.88 when network coding is used. Note that the total achievable throughput is 0.80 when network coding is not used. We also present the convergence of Lagrange multipliers; $q_{A_1,I}$, $q_{A_2,I}$, q_{I,A_2}, q_{I,A_1}, and $q_{I,\{A_1,A_2\}}$ for both cases in Fig. 4.18(b) and Fig. 4.19(b), respectively.

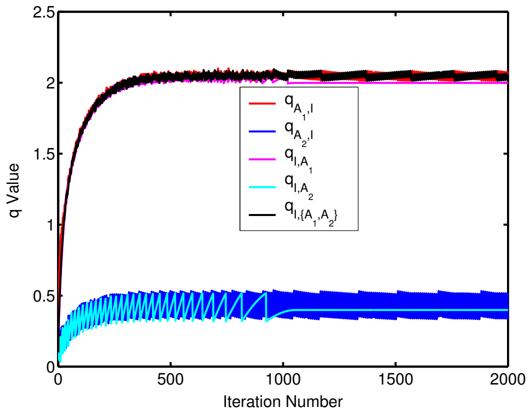
Second, we consider the X topology presented in Fig. 4.9. We consider two cases for wireless channel capacities: (i) $C_1 = C_2 = C_3 = C_4 = 1$, and (ii) $C_1 = C_4 = 1$, $C_2 = C_3 = 4$. In both cases the total rate $x_1 + x_2$ approaches to the optimum achievable rates; 0.66 and 1.3 as seen in Fig. 4.20(a) and Fig. 4.21(a). We also show results for the convergence of the Lagrange multipliers for both cases in Fig. 4.20(b) and Fig. 4.21(b).

Multi-Hop Network Coding

We consider the butterfly topology presented in Fig. 4.16. We consider two scenarios for the wireless channel capacities; (i) $C_1 = C_2 = C_3 = C_4 = C_5 = 1$ and (ii) $C_1 = C_2 = C_4 = C_5 =$

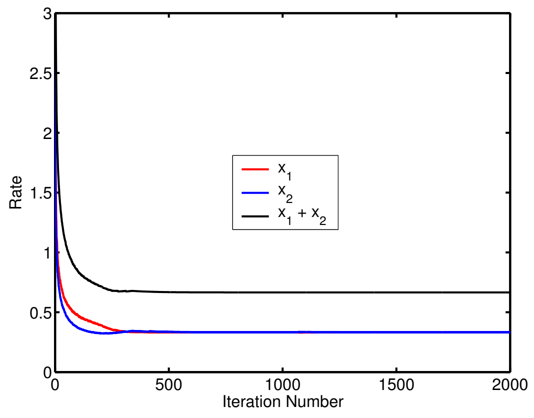


(a) Rate

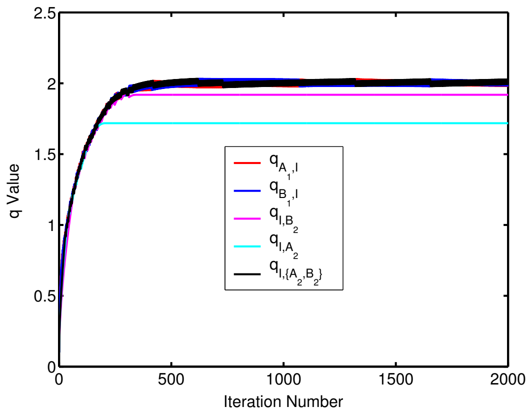


(b) Lagrange multipliers

Figure 4.19: Convergence results for the Alice-and-Bob topology presented in Fig. 4.10(a). The total achieved rate approaches the optimum throughput 0.88. The optimum throughput is 0.80 when there is no network coding. $C_1 = 1$, $C_2 = 4$.



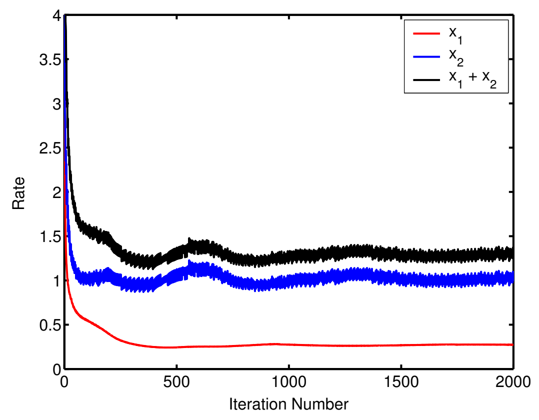
(a) Rate



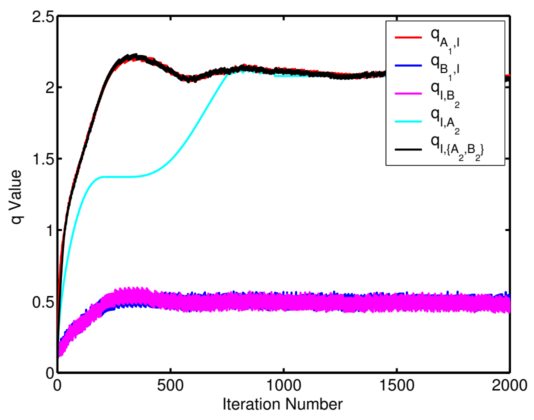
(b) Lagrange multipliers

Figure 4.20: Convergence results for the X topology presented in Fig. 4.9. The total achieved rate approaches to the optimum throughput 0.66. The optimum throughput is 0.50 when there is no network coding. $C_1 = C_2 = C_3 = C_4 = 1$.

4, $C_3 = 1$. The total rate approaches the optimal achievable rate in both scenarios: 0.5 for the first case as shown in Fig. 4.22(a), and 1.14 for the second case as shown in Fig. 4.23(a). We also show the convergence of the Lagrange multipliers, Fig. 4.22(b) and Fig. 4.23(b).

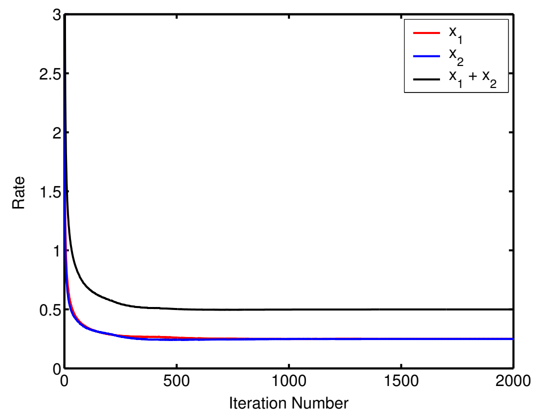


(a) Rate

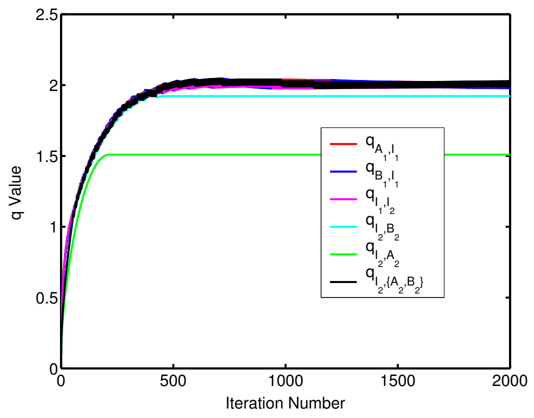


(b) Lagrange multipliers

Figure 4.21: Convergence results for the X topology presented in Fig. 4.9. The total achieved rate approaches the optimum throughput 1.3. The optimum throughput is 0.80 when there is no network coding. $C_1 = C_4 = 1$, $C_2 = C_3 = 4$.



(a) Rate

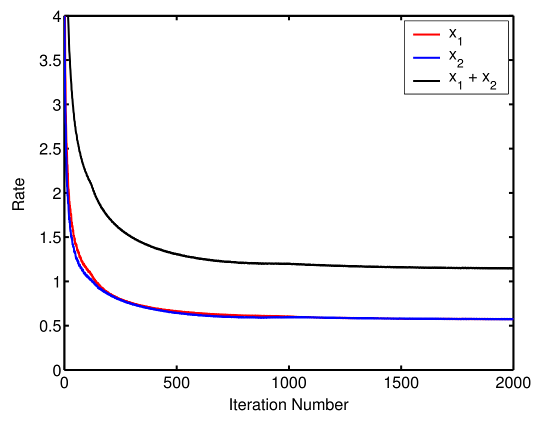


(b) Lagrange multipliers

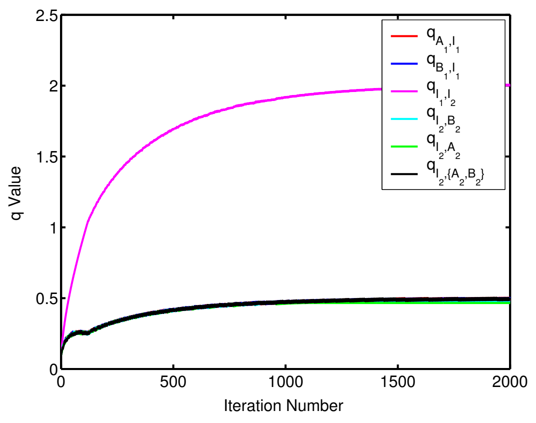
Figure 4.22: Convergence results for the butterfly topology presented in Fig. 4.16. The total achieved rate approaches the optimum throughput 0.50. The optimum throughput is 0.33 when there is no network coding. $C_1 = C_2 = C_3 = C_4 = C_5 = 1$.

4.5 Summary

In this chapter, we first argued in favor of making rate control and scheduling network-coding aware over coded wireless networks. We gave the main intuition and also compared NC-aware



(a) Rate



(b) Lagrange multipliers

Figure 4.23: Convergence results for the butterfly topology presented in Fig. 4.16. The total achieved rate approaches the optimum throughput 1.14. The optimum throughput is 0.66 when there is no network coding. $C_1 = C_2 = C_4 = C_5 = 4$, $C_3 = 1$.

vs. NC-unaware, both optimal and practical, schemes. In addition to the formulations, we presented simulation results for an illustrative example with two cross flows, which captures the main intuition and should be a building block of any large scenario with cross-flows and network coding. In general, the benefit from NC awareness depends on (i) the network coding opportunities (ii) the conflicts of network coded flows with other flows and (iii) the link rates where (i) and (ii) depend on the topology and traffic scenario.

Second, we showed that video rate control naturally fits in network utility maximization framework, with the additional complication that the time varying video rate and utility affect the network coding opportunities and the total achieved utility; to deal with time variability, we proposed video rate control over an appropriate time interval so as to optimize the utility vs. delay tradeoff. We formulated the problem, and showed that it has a distributed solution, and we provided insights into the interaction of rate control and network coding for video streaming.

Third, we showed how to improve the performance of TCP over coded wireless networks.

The key intuition was to eliminate the rate mismatch between flows that are coded together through a synergy of rate control and queue management. First, we formulated congestion control as a NUM problem and derived a distributed solution. Motivated by the structure of the solution, we proposed minimal modifications to queue management to make it network coding-aware, while TCP and MAC protocols remained intact. Simulation results show that the proposed NCAQM scheme doubles TCP performance compared to baseline schemes and achieves near-optimal performance. We have also extended the NUM formulation and solution to multi-hop network coding and we have confirmed convergence through numerical calculations.

Chapter 5

I^2NC: Intra- and Inter-Session Network Coding

The focus of this chapter is on the performance of network coding in the presence of non-negligible loss rates. Loss in wireless networks is a challenging problem: recent studies of IEEE 802.11b based wireless mesh networks [90], [91], have reported packet loss rates as high as 50% which typically translates to low application level goodput and high delay [92]. Furthermore, network coding over wireless networks amplifies this problem. The reasons are that (i) when a network coded packet is lost more than one packet is lost which should be considered separately, (ii) intermediate nodes in opportunistic network coding scheme (COPE, [10]) requires the knowledge of what their neighbors have overheard, in order to perform one-hop inter-session network coding. However, in the presence to medium-high loss rate, this information is limited, possibly corrupted, and/or delayed over wireless channels. The approach taken by [10] is to turn off network coding if loss rate exceeds a threshold with default value 20% [10]. However, this does not take full advantage of all the available network coding opportunities. To better illustrate this key point, let us discuss the following example.

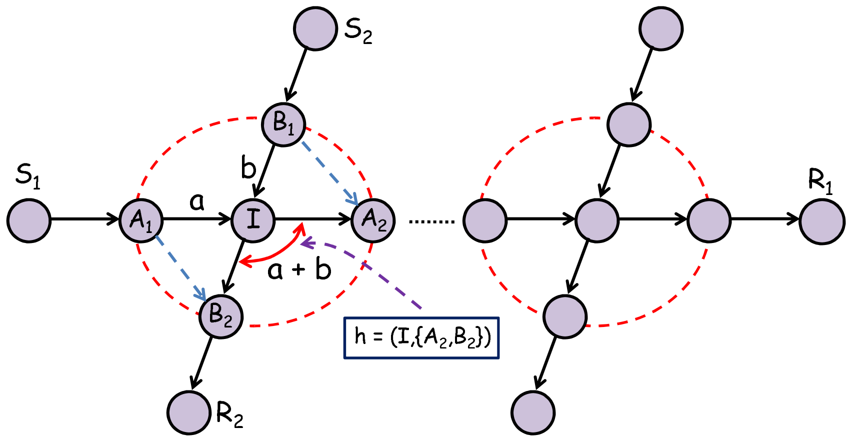


Figure 5.1: Example of a unicast flow (from S_1 to R_1) traversing multiple wireless hops. Each intermediate node performs one-hop (intra-session and inter-session) network coding. The neighborhood of one intermediate node I is shown here in detail. (Two unicast flows, $S_1 - R_1$ and $S_2 - R_2$, meeting at intermediate node I. I receives packets a, b from nodes A_1, B_1, respectively. It can choose to broadcast a, b or $a + b$ in a single transmission to both receivers. The next hops A_2, B_2 can decode $a + b$ because they overhear packets b and a transmitted from B_1, A_1, respectively.)

Example 11 Let us consider Fig. 5.1. For the moment, let us focus only on the neighborhood of intermediate node I, *i.e.,* only the packets transmitted via I, from A_1 to A_2 and from B_1 to B_2. This forms an X topology and is a well-known, canonical example of one-hop opportunistic network coding [9, 10]. In the absence of loss, throughput is improved by 33.3%, because I delivers two packets in three transmissions (with network coding), instead of four (without network coding). Let us consider this example when there is packet loss. Assume that there is loss only on the overhearing link $A_1 - B_2$, w.p. $\rho_{\{A_1,B_2\}} = 0.3$, and all other links have no loss. In this case, 70% of the packets can still be coded together, and throughput can be improved by 26%, which is still a significant improvement. Even at higher loss rate, *e.g.,* $\rho_{\{A_1,B_2\}} = 0.5$, inter-session coding still improves throughput by 16.6%. This is under the assumption that I knows the exact state of A_2, B_2, *i.e.,* what packets were overheard, and thus I is able to decide what packets to code together so as to guarantee decodability at the receivers. However, at high loss rates, coordination among nodes becomes difficult. This is why [10] turns off the coding functionality when loss rate is higher than a threshold with default value 20%, thus not taking full advantage of all coding opportunities. □

In this chapter, we propose a solution to this problem by introducing redundancy at intermediate nodes. As compared to previous work in [93], [72] which explore network coding benefit using redundancy and additional transmissions, we consider intra-session network coding to combine packets within the same flow and introduce parity packets to protect against loss. Then, we use inter-session network coding to combine packets from different (already intra-session coded) flows, as in [9, 10], and thus increase throughput. Our approach for combining intra- with inter-session network coding, which we refer to as I^2NC, has two key benefits. First, it can correct packet loss and still perform inter-session network coding, even in the presence of medium-high loss rates, thus improving throughput. Second, the use of intra-session network coding makes all packets in the session equally beneficial for the receiver. Thus, I^2NC eliminates the need to know the exact packets that have been overheard by the neighbors of intermediate node I.

We note that adding redundancy in the this setting is non-trivial, since a flow is affected not only by loss on its direct path, but also by loss on overhearing links that affect the decodability of coded packets. Therefore, we need to carefully determine how much redundancy to add and on which node.

Example 11 - continued Consider again the neighborhood of I in Fig. 5.1. Flow 2 is affected not only by loss on its own path $B_1 - I - B_2$, but also by loss on the overhearing link $A_1 - B_2$, which affects the decodability of coded packet $a + b$ at B_2. In order to protect flow 2 from high loss rate on the overhearing link $A_1 - B_2$, the intermediate node I may decide either to add redundancy on flow 2, or to not perform coding (as in [10]), or a combination of the two. On the other hand, I may also decide to add redundancy on flow 1, to correct loss on the overhearing link $A_1 - B_2$, thus helping B_2 to receive a and be able to decode $a + b$. $\square$

Therefore, a number of questions need to be addressed in the design of a system that combines both intra- and inter-session network coding. In particular:

Q1: *How to gracefully combine intra- and inter-session network coding* at intermediate nodes? We propose a generation-based design and specify the order in which we perform the two types of coding.

Q2: *How much redundancy to add in each flow?* We show how to adjust the amount of redundancy after taking into account the loss on the direct and overhearing links. We implement the intra-session network coding functionality as a thin layer between IP and transport layer.

Q3: *What percentage of flows should be coded together* and what parts should remain uncoded? We design algorithms that make this decision taking into account the loss characteristics on the direct and overhearing links. We implement this and other functionality (*e.g.,* queue management) performed with or after inter-session network coding as a layer between MAC and IP (similarly to [10]).

Q4: *What information do we need to know* in order to make these decisions? We propose two schemes: I^{2}NC-state, which needs to know the state (*i.e.,* overheard packets) of the neighbors; and I^{2}NC-stateless, which only needs to know the loss rate of links in the neighborhood.

Our approach is grounded on a network utility maximization (NUM) framework [60]. We formulate two variants of the problem, depending on what type of information is available (as in question Q4 above). The solution of each problem decomposes into several parts with an intuitive interpretation, such as rate control, network coding rate, redundancy rate, queue management, and scheduling. The structure of the optimal solution provides insight into the design of our two schemes, I^{2}NC-state and I^{2}NC-stateless.

We evaluate our schemes in a multi-hop setting and we consider their interaction with the transport layer, including TCP and UDP. We propose a thin adaptation layer at the interface between TCP and the underlying coding, to best match the interaction of the two. We perform simulations in `GloMoSim` [77] and we show that our schemes significantly improve throughput compared to the scheme proposed in [10].

The structure of the rest of the chapter is as follows. Section 5.1 gives an overview of the system model. Section 5.2 presents the NUM formulation and solution. Section 5.3 presents the design of the I^{2}NC schemes in detail. Section 5.4 presents simulation results. Section 5.6 summarizes the chapter.

5.1 System Model

We consider multi-hop wireless networks, where intermediate nodes perform intra- and inter-session network coding (I^{2}NC), as exemplified in Fig. 5.1. In what follows, we provide an overview of the system and highlight some of its key characteristics.

5.1.1 Notation and Setup

Sources and Flows

Let $\mathcal{S}$ be the set of unicast flows between source-destination pairs in the network. Each flow $s \in \mathcal{S}$ is associated with a rate x_s and a utility function $U_s(x_s)$, which we assume to be a strictly concave function of x_s.

Wireless Transmission

Packet transmissions from a source (*e.g.,* S_1 in Fig. 5.1) traverses potentially multiple wireless hops before being received by the receiver (*e.g.,* R_1). We consider the model for interference in wireless networks described in [81]: each node can either transmit or receive at the same time and all transmissions in the range of the receiver are considered as interfering.

We use the following terminology for the links involved in one-hop transmission. A hyperarc $(i, \mathcal{J})$ is a collection of links from node $i \in \mathcal{N}$ to a non-empty set of next-hop nodes $\mathcal{J} \subseteq \mathcal{N}$. A hypergraph $\mathcal{H} = (\mathcal{N}, \mathcal{A})$ represents a wireless mesh network, where $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ is the set of hyperarcs. For simplicity, $h = (i, \mathcal{J})$ denotes a hyperarc, $h(i)$ denotes node i and $h(\mathcal{J})$ denotes node $\mathcal{J}$, *i.e.*, $h(i) = i$ and $h(\mathcal{J}) = \mathcal{J}$. We use these representations interchangeably. Each hyperarc h is associated with a channel capacity R_h. Since h is a set of links, R_h is the minimum link capacity of the links in the hyperarc: $R_h = \min_{j \in h(J)}\{R_{i,j}\}$ s.t. $i \in \mathcal{N}$. In the example of node I in Fig. 5.1, $h = (I, \{B_2, A_2\})$ is one of the hyperarcs, it consists of links $I - B_2$ and $I - A_2$ and has capacity $\min\{R_{\{I,B_2\}}, R_{\{I,A_2\}}\}$.

Note that with both intra-session and inter-session network coding, it is possible to construct more than one code over a hyperarc $(i, \mathcal{J})$. Let $\mathcal{K}_{i,\mathcal{J}}$ be the set network codes over a hyperarc $(i, \mathcal{J})$. $\mathcal{S}_k \subseteq \mathcal{S}$ be the set of flows coded together using code $k \in \mathcal{K}_{i,\mathcal{J}}$ and broadcast over $(i, \mathcal{J})$[1].

Given a hypergraph $\mathcal{H}$, we can construct the conflict graph $\mathcal{C} = (\mathcal{A}, \mathcal{I})$, whose vertices are the hyperarcs of $\mathcal{H}$ and edges indicate interference between hyperarcs. A clique $\mathcal{C}_q \subseteq \mathcal{A}$ consists of several hyperarcs, at most one of which can transmit simultaneously without interference.

[1]Note that we consider constructive inter-session network coding, *i.e.*, network codes $k \in \mathcal{K}_{i,\mathcal{J}}$ as well as $h = (i, \mathcal{J})$ is determined at each node with periodic control packet exchanges or estimated through routing table.

Loss Model

Consider one-hop transmission. A flow s may experience loss in two forms: loss ρ_h^s over the direct transmission links; or loss $\rho_{h,k}^{s,s'}$ of antidotes[2] on overhearing links. These two types of losses have different impacts on network coded flows.

First, let us discuss loss on the direct links. A flow s transmitted over hyperarc h experiences loss w.p. ρ_h^s. This probability is different per flow s, even if several flows are coded and transmitted over the same hyperarc h, because different flows are transmitted to different next hops, thus see different channels. For example, in Fig. 5.1, $\rho_{(I,\{B_2,A_2\})}^{S_1}$ is equal to the loss probability over link $I - A_2$ and $\rho_{(I,\{B_2,A_2\})}^{S_2}$ is equal to the loss probability over link $I - B_2$.

Second, let us discuss the effect of lost antidotes on the overhearing link. Consider that flow s is combined with flow s' s.t. $s \neq s'$, and that some packets of flow s' are lost on the overhearing link to the next hop of s. Then, coded packets cannot be decoded at the next hop and flow s loses packets, with probability $\rho_{h,k}^{s,s'}$. For example, in Fig. 5.1, packets from flow S_1 cannot be decoded (hence are lost) at node A_2 due to loss of antidotes from flow S_2 on the overhearing link $B_1 - A_2$.

In our formulation and analysis, we assume that ρ_h^s and $\rho_{h,k}^{s,s'}$ are i.i.d. according to a uniform distribution. However, in our simulations, we also consider a Rayleigh fading channel model. The loss probabilities are calculated at each intermediate node as one minus the ratio of correctly received packets over all the packets in a time window. These loss probabilities are obtained by the upstream node (*e.g.,* I in Fig. 5.1) periodically via control packets from the downstream node on the link (*e.g.,* B_2, A_2 in Fig. 5.1).

[2]Following the poison-antidote terminology of [71], we call antidotes the packets of flows s' that are coded together with s, and thus are needed for the next hop of s to be able to decode. *E.g.*, in Fig.5.1, a is the antidote that B_2 needs to overhear over link $A_1 - B_2$, to decode $a + b$ and obtain b.

Routing

We assume that each flow $s \in \mathcal{S}$ follows a single path $\mathcal{P}_s \subseteq \mathcal{N}$ from the source to the destination, which is pre-determined by a routing protocol, *e.g.*, OLSR or AODV, and given as input to our problem. Note that several different hyperarcs may connect two consecutive nodes along the path. We define $H^{s,k}_{i,\mathcal{J}} = 1$ if s is transmitted through hyperarc $(i, \mathcal{J})$ using network code $k \in \mathcal{K}_{i,\mathcal{J}}$; and $H^{s,k}_{i,\mathcal{J}} = 0$, otherwise.

5.1.2 Intra- and Inter-session Network Coding

Next, we give an overview of how an intermediate node performs first intra-session network coding (to increase loss tolerance) and then inter-session network coding (to improve throughput). The implementation details are provided in Section 5.3.

Intra-session Network Coding (for Error Correction)

Consider the commonly used generation-based network coding [27]: packets from flow $s \in \mathcal{S}$ are divided into generations (note that we use generation and block terms interchangeably), with size G^s. At the source s, packets within the same generation are linearly combined (assuming large enough field size) to generate G^s network coded packets. Each intermediate node along the path of flow s adds P^s parity packets, depending on the loss rates of the links involved in this hop. At the next hop, it is sufficient to receive G^s out of $G^s + P^s$ packets. The same process is repeated at every intermediate node until the receiver receives G^s error-free packets, which can then be decoded and be passed on to the application.

There are many ways to generate those P^s parities in practice. In this work, we use intra-session network coding [94] for this purpose. Each intermediate node stores G^s packets of the same flow and generates P^s random linear combinations; w.h.p. any G^s out of $G^s +$

P^s are linearly independent, thus can be used to reconstruct G^s packets. Note that an intermediate node does not need to decode; it just combines packets in a generation and updates their global coefficients [27]. Although one could use various coding techniques, such as Reed-Solomon or Fountain codes, implementing error correction via intra-session network coding has several advantages. First, it has lower computational complexity than other error schemes. Second, in systems that already implement inter-session network coding, it is natural to incrementally add intra-session network coding functionality. Intra-session network coding can be implemented as a thin layer between IP and transport, as shown in Fig. 5.2.

Inter-session Network Coding (for Throughput)

After an intermediate node has added redundancy (P^s) to flow s, it treats all $(G^s + P^s)$ packets as indistinguishable parts of the same flow. Inter-session network coding is applied on top of the already intra-coded flows, as a thin layer between MAC and IP, shown in Fig. 5.2. We design two schemes, I^{2}NC-state and I^{2}NC-stateless, depending on the type of information they need about their neighbors. We define as state of a node the information about which exact packets have been overheard at that node; overheard packets always belong to flows that are of not interest to the node.

I^{2}NC-state: First, we assume that intermediate nodes uses one-hop opportunistic network coding proposed in [10]. Each node i listens all transmissions in its neighborhood, stores the overheard packets in its decoding buffer, and periodically advertises the content of this buffer to its neighbors. When a node i wants to transmit a packet, it checks or estimates the contents of the decoding buffer of its neighbors. If there is a coding opportunity, the node combines the relevant packets using simple coding operations (XOR) and broadcasts the combination to $\mathcal{J}$. The content of the decoding buffers needs to be exchanged, in order to make network coding decisions, via some protocol for state synchronization.

I^{2}NC-stateless: Second, we design an improved version, which no longer requires state synchronization. The key idea is to exploit the fact that the redundancy already introduced by intra-session coding makes all $G^s + P^s$ packets in a generation equally important[3]. In this improved scheme, each node i still listens to all transmissions in its neighborhood and stores the overheard packets[4]. The node periodically advertises the loss probability for each overheard flow, which is then provided as input to the intra-session network coding module in order to determine the amount of redundancy needed[5].

In summary, there is a nice synergy between intra- and inter-session network coding. Intra-session network coding makes the process sequence agnostic, which allows inter-session network coding to operate using only information about the loss rates, not about the identity of received packets. Conversely, the loss rates can be used as input for tuning the amount of redundancy in intra-session network coding. In terms of implementation, the two modules are separable: an intermediate node first performs intra-session network coding, then inter-session network coding.

[3]It no longer matters which exact packets a node has. As long as a node has any G^s out of $G^s + P^s$, it can decode. As long it knows the percentage of overheard received packets it can make coding decisions.

[4]As in [10], only original overheard packets are stored in the decoding buffer. Coded overheard packets are discarded.

[5]The loss rate over direct transmission links is calculated as a ratio of the number of lost packets at each hop vs. the total number of packets. The loss rate over overhearing links is calculated as effective loss rate. *E.g.,* in Fig. 5.1, the loss rate at node A_2 is calculated as follows. If $G^{S_1} +^P S_1$ packets are sent by B_1 and at least G^{S_1} packets are received at A_2, then the loss rate is set to 0. If $G^{S_1} - 1$ packets are received by A_2, then the loss rate is set to $1/G^{s_1}$.

5.2 Network Utility Maximization Formulation

5.2.1 I^{2}NC-state

Formulation

Our objective is to maximize the total utility function by optimally choosing the flow rates x_s at sources $s \in \mathcal{S}$, as well as the following variables at the intermediate nodes: the fraction $\alpha^s_{h,k}$ (or traffic splitting parameters, following the terminology of [66]) of flows inter-session coded using code $k \in \mathcal{K}_h$ over hyperarc h; and the percentage of time $\tau_{h,k}$ each hyperarc is used.

$$\begin{aligned}
&\max_{\boldsymbol{x},\boldsymbol{\alpha},\boldsymbol{\tau}} \sum_{s\in\mathcal{S}} U_s(x_s) \\
&\text{s.t. } \frac{H^s_{h,k}\alpha^s_{h,k}x_s}{1-\rho^s_h} + \sum_{s'\in\mathcal{S}_k-\{s\}} H^{s'}_{h,k}\alpha^{s'}_{h,k}x_{s'}\rho^{s,s'}_{h,k} \le R_h\tau_{h,k},\ \forall h\in\mathcal{A}, k\in\mathcal{K}_h, s\in\mathcal{S}_k \\
&\qquad \sum_{h(\mathcal{J})|h\in\mathcal{A}}\ \sum_{k\in\mathcal{K}_h|s\in\mathcal{S}_k} \alpha^s_{h,k} = 1,\ \forall s\in\mathcal{S}, i\in\mathcal{P}_s \\
&\qquad \sum_{h\in\mathcal{C}_q}\sum_{k\in\mathcal{K}_h} \tau_{h,k} \le \gamma,\ \forall \mathcal{C}_q \subseteq \mathcal{A}
\end{aligned} \tag{5.1}$$

The first constraint is the capacity constraint for each flow $s \in \mathcal{S}_k$. It is well-known, [65], that network coding allows flows that are coded together in code $k \in \mathcal{K}_h$, to coexist, *i.e.*, each have rate up to the rate allocated to that code k. The right hand side, $R_h\tau_{h,k}$, is the capacity of hyperarc h; $\tau_{h,k}$ is the percentage of time hyperarc h can be used for transmitting the k-th network code. $\tau_{h,k}$ is determined by scheduling in the third constraint, taking into account interference: all hypearcs in a clique interfere and should time-share the medium. Therefore, the sum of the time allocated to all hyperarcs in a clique should be less than an over-provisioning factor, $\gamma \le 1$. The second constraint is the flow conservation: at every node i on the path $\mathcal{P}_s$ of source s, the sum of $\alpha^{s,k}_h$ over all network codes and hyperarcs should be equal to 1. Indeed, when a flow enters a particular node i, it can be transmitted to its next hop j as part of different network coded and uncoded flows.

Intra-session Network Coding. The first constraint is key to our work because it determines how to deal with loss on the direct (ρ_h^s) and overhearing ($\rho_{h,k}^{s,s'}$) links and how large a fraction ($\alpha_h^{s,k}$) of flow rate (x_s) to code in the k-th code over hyperarc h. Let us discuss the left hand side in more detail[6].

The first term refers to the direct link of flow s. $H_{h,k}^s \alpha_{h,k}^s x_s$ is the fraction of flow rate x_s allocated to code k and hyperarc h. It is scaled by $1-\rho_h^s$ to indicate that we use intra-session redundancy to protect against loss that flow s experiences w.p. ρ_h^s. $(H_{h,k}^s \alpha_{h,k}^s x_s)/(1-\rho_h^s)$ is the total rate of flow s, including data and redundancy.

The second term refers to loss on the overhearing links. $\sum_{s' \in \mathcal{S}_k - \{s\}} H_{h,k}^{s'} \alpha_{h,k}^{s'} x_{s'} \rho_{h,k}^{s,s'}$ is the amount of redundancy (via intra-session coding) added by the intermediate node on flow(s) s' to protect flow s against loss of antidote packets. These antidotes come from other flows ($s' \in \mathcal{K}_h$) that are coded together with flow s, reach the next hop for flow s through the overhearing links, and are needed to decode inter-session coded packets.

Example 11 - continued. In Fig. 5.1, let us consider flow 2 from B_1 to B_2, as the flow of interest. The intermediate node I adds redundancy to S_2 to protect against loss rate $\rho_{(I,\{B_2,A_2\})}^{S_2}$ on the direct link $I-B_2$. It also adds redundancy to flow 1 to protect against loss rate $\rho_{(I,\{B_2,A_2\})}^{S_2,S_1}$ of antidotes coming to B_2 from flow 1 over the overhearing link $A_1 - B_2$[7]. $\square$

[6]Note that our formulation of has two novel aspects, compared to prior work, which allow us to better handle loss and parities. First, we allow for flows coded together to have different rates (*e.g.,* in the first constraint in Eq. (5.1)). Second, we allow for loss rates of each link to be specified separately, even for links in the same hyperarc.

[7]An alternative way to protect flow 2 from loss on the overhearing link $A_1 - B_2$ would be to have node A_1 itself to altruistically add redundancy. We choose not to implement this option, because realistically I is the node receiving the feedback from B_2.

Optimal Solution

By relaxing the capacity constraint in Eq. (5.1), we have:

$$L(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\tau}, \boldsymbol{q}) =$$
$$\sum_{s\in\mathcal{S}} U_s(x_s) - \sum_{h\in\mathcal{A}} \sum_{k\in\mathcal{K}_h} \sum_{s\in\mathcal{S}_k} q^s_{h,k}\Big(\frac{H^s_{h,k}\alpha^s_{h,k}x_s}{1-\rho^s_h} + \sum_{s'\in\mathcal{S}_k-\{s\}} H^{s'}_{h,k}\alpha^{s'}_{h,k}x_{s'}\rho^{s,s'}_{h,k} - R_h\tau_{h,k}\Big), \qquad (5.2)$$

where $q^s_{h,k}$ is the Lagrange multiplier, which can be interpreted as the queue size for k-th network code at hyperarc h for flow s. We define $\rho^{s,s'}_h = 0$ if $s = s', \forall s, s' \in \mathcal{S}$ and we rewrite $\sum_{k\in\mathcal{K}_h}\sum_{s\in S_k}$ as $\sum_{s\in\mathcal{S}}\sum_{k\in K_h|s\in\mathcal{S}_k}$. The Lagrange function is $L(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\tau}, \boldsymbol{q}) = \sum_{s\in\mathcal{S}}(U_s(x_s) - x_s\sum_{h\in\mathcal{A}}\sum_{k\in\mathcal{K}_h|s\in\mathcal{S}_k} H^s_{h,k}\alpha^s_{h,k}\Big((q^s_{h,k})/(1-\rho^s_h) + \sum_{s'\in\mathcal{S}_k} q^{s'}_{h,k}\rho^{s',s}_{h,k}\Big)) +$ $\sum_{h\in\mathcal{A}}\sum_{k\in\mathcal{K}_h}\sum_{s\in\mathcal{S}_k} q^s_{h,k}R_h\tau_{h,k}$. It can be decomposed into several intuitive problems (rate control, traffic splitting, scheduling, and queue update), each of which solves the optimization problem for one variable.

Rate Control. First, we solve the Lagrangian w.r.t x_s:

$$x_s = (U'_s)^{-1}\left(\sum_{i\in\mathcal{P}_s} Q^s_i\right), \qquad (5.3)$$

where $(U'_s)^{-1}$ is the inverse function of the derivative of U_s, and Q^s_i is the occupancy of flow s at node i and expressed as

$$Q^s_i = \sum_{h(J)|h\in\mathcal{A}} \sum_{k\in\mathcal{K}_h} H^s_{h,k}\alpha^s_{h,k}Q^s_{h,k}, \qquad (5.4)$$

where $Q^s_{h,k}$ is the queue size of flow s associated with hyperarc and network code pair $\{h, k\}$:

$$Q^s_{h,k} = \frac{q^s_{h,k}}{1-\rho^s_h} + \sum_{s'\in\mathcal{S}_k-\{s\}} q^{s'}_{h,k}\rho^{s',s}_{h,k} \qquad (5.5)$$

Traffic Splitting. Second, we solve the Lagrangian for $\alpha_h^{s,k}$. At each node i along the path (*i.e.*, $i \in \mathcal{P}_s$), the traffic splitting problem can be expressed as follows:

$$\begin{aligned}
&\min_{\boldsymbol{\alpha}} \sum_{h(J)|h\in\mathcal{A}} \sum_{k\in K_h|s\in\mathcal{S}_k} \alpha^s_{h,k} H^s_{h,k} Q^s_{h,k} \\
&\text{s.t.} \sum_{h(J)|h\in\mathcal{A}} \sum_{k\in K_h|s\in\mathcal{S}_k} \alpha_h^{s,k} = 1,\ \forall i \in \mathcal{P}_s. \qquad (5.6)
\end{aligned}$$

The structure of the optimal solution of Eq. (5.6) has the following interpretation: the higher the loss rate of antidotes on overhearing links $\rho_{h,k}^{s',s}$, the higher $Q^s_{h,k}$ in Eq. (5.5), and the smaller $\alpha^s_{h,k}$ in the minimization in Eq. (5.6). This means that flow s should code fewer packets with packets from flow(s) s' in code k, when antidotes from s' are likely to be lost, thus making it impossible for the next hop of flow s to decode.

Example 11 - continued: In Fig. 5.1, this means that I should combine fewer packets from the two flows if there is loss on the overhearing link $A_1 - B_2$. In the extreme case where loss rate is 1 over the link $A_1 - B_2$, inter-session coding should be turned off. At the other extreme, where there is no loss, the two flows should always be combined. □

Scheduling. Third, we solve the Lagrangian for $\tau_{h,k}$. This problem is solved for every hyperarc and every clique for the conflict graphs in the hypergraph. In our implementation, we solve this problem for every hypeararc originated from a node to determine which packets should be inter-session network coded and transmitted.

$$\begin{aligned}
&\max_{\boldsymbol{\tau}} \sum_{h\in\mathcal{A}} \sum_{k\in\mathcal{K}_h} \sum_{s\in\mathcal{S}_k} q^s_{h,k} R_h \tau_{h,k} \\
&\text{s.t.} \sum_{h\in\mathcal{C}_q} \tau_{h,k} \le \tau,\ \forall \mathcal{C}_q \subseteq A \qquad (5.7)
\end{aligned}$$

Queue Update. We find the Lagrange multipliers (queue sizes) $q^s_{h,k}$, using gradient descent:

$$q^s_{h,k}(t+1) = \{q^s_{h,k}(t) + c_t\{\frac{H^s_{h,k}\alpha^s_{h,k}x_s}{1-\rho^s_h} + \sum_{s'\in\mathcal{S}_k-\{s\}} H^{s'}_{h,k}\alpha^{s'}_{h,k}x_{s'}\rho^{s,s'}_{h,k} - R_h\tau_{h,k}\}\}^+ \tag{5.8}$$

where t is the iteration number, c_t is a small constant, and the $^+$ operator makes the Lagrange multipliers positive. $q^s_{h,k}$ is interpreted as the queue for flow s allocated for the k-th network code over hyperarc $\forall h \in \mathcal{A}$. Indeed, in Eq. (5.8), $q^s_{h,k}$ is updated with the difference between the incoming $(H^s_{h,k}\alpha^s_{h,k}x_s)/(1-\rho^s_h) + \sum_{s'\in\mathcal{S}_k-\{s\}} H^{s'}_{h,k}\alpha^{s'}_{h,k}x_{s'}\rho^{s,s'}_{h,k}$ and outgoing $R_h\tau_{h,k}$ traffic rates at h.

5.2.2 I^{2}NC-stateless

The second term in Eq. (5.1) describes the redundancy added by node i to protect s from loss of antidotes on the overhearing link. An implicit assumption was that node i knows what antidotes are available at the next hop and uses only those packets for inter-session coding. However, this knowledge can be imperfect, especially in the presence of loss. Here, we formulate a variation of the problem, where such knowledge is not necessary. Instead, node i needs to know only the loss rate on all the links to the next hop for flow s (*e.g.,* in Fig. 5.1 for flow 2 (S_2), these are links $I - B_2$ and $A_1 - B_2$).

We replace the capacity constraint in Eq. (5.1) with:

$$\frac{H^s_{h,k}\alpha^s_{h,k}x_s}{1-\rho^s_h} + \sum_{s'\in\mathcal{S}_k} \frac{H^{s'}_{h,k}\alpha^{s'}_{h,k}x_{s'}\rho^{s,s'}_{h,k}}{1-\rho^s_h} \le R_h\tau_{h,k} \tag{5.9}$$

and this is $\forall h \in \mathcal{A}, k \in \mathcal{K}_h, s \in \mathcal{S}_k$. The other constraints remain the same as in Eq. (5.1). The difference from Eq. (5.1) is in the second term, related to the overheard packets at the next hop. Any fraction of flow s' added as redundancy to flow s, as well as overheard packets from s' in the next hop, help to decode inter-session coded packets of s with flow s'. To protect transmissions of these "helping" fractions ($H^{s'}_{h,k}\alpha^{s'}_{h,k}x_{s'}\rho^{s,s'}_{h,k}$) from being lost on the direct link to the next hop of flow s (*e.g.,* from I to B_2), we add redundancy to match the loss rate of that direct link (ρ^s_h in general, $\rho^{S_2}_{\{I,B_2\}}$ in the example). This is why the term $H^{s'}_{h,k}\alpha^{s'}_{h,k}x_{s'}\rho^{s,s'}_{h,k}$ is divided by $1-\rho^s_h$.

The solution of this optimization problem also decomposes into rate control, traffic splitting, and scheduling problems, which correspond to Eq. (5.3), (5.6), and (5.7), respectively. $Q^s_{h,k}$ needs to be updated:

$$Q^s_{h,k} = \frac{q^s_{h,k}}{1-\rho^s_h} + \sum_{s'\in\mathcal{S}_k-\{s\}} \frac{q^{s'}_{h,k}\rho^{s',s}_{h,k}}{1-\rho^{s'}_h}. \tag{5.10}$$

The Lagrange multiplier is also updated as follows;

$$q^s_{h,k}(t+1) = \{q^s_{h,k}(t) + c_t\{\frac{H^s_{h,k}\alpha^s_{h,k}x_s}{1-\rho^s_h} + \sum_{s'\in\mathcal{S}_k-\{s\}} \frac{H^{s'}_{h,k}\alpha^{s'}_{h,k}x_{s'}\rho^{s,s'}_{h,k}}{1-\rho^s_h} - R_h\tau_{h,k}\}\}^+ \tag{5.11}$$

We have verified, through numerical calculations, the convergence of the optimal solutions which is presented in Section 5.5.

5.3 System Implementation

We propose practical implementations of the I^{2}NC-state and I^{2}NC-stateless schemes (Fig. 5.2), following the NUM formulation structure.

5.3.1 Operation of End-Nodes

There is an adaptation layer between transport and intra-session network coding (Fig. 5.2) at end nodes. This has two tasks: (i) to interface between application and intra-session

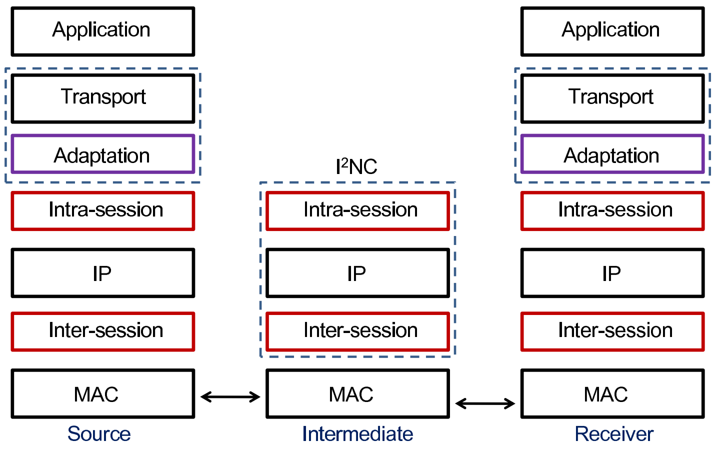


Figure 5.2: Operations taking place at end-points and intermediate nodes.

coded packets; and (ii) optimize the reliability mechanism at the transport layer.

Task (i): At the source, the adaptation layer sets the generation (block) size to G^s. G^s is set according to application (*e.g.*, media transmission) requirements (for UDP), or set equal to TCP congestion window (for TCP) and changes over time as proposed in [95]. The adaptation layer receives G^s original packets $a_1, a_2, ..., a_{G^s}$ from the transport layer of flow s and generates G^s intra-session coded packets; $p_1 = a_1$, $p_2 = a_1 + a_2$, ..., $p_{G^s} = a_1 + ...a_{G^s}$. We chose this iterative coding to avoid introducing coding delays. The intra-session header includes the block id, packet id, block size, and coding coefficients. At the receiver side, the reverse operations are performed.

Task (ii): To further optimize the interaction between I^{2}NC and transport, particularly TCP, we keep track and acknowledge the number of received packets in a generation, rather than their sequence numbers. This idea is similar to the use of end-to-end FEC that makes TCP sequence agnostic [96, 95]; and to intra-session network coding over TCP's window and acknowledging degrees of freedom in [44]. *E.g.*, if a receiver receives the first packet labeled with block id $g^s = 1$, then it generates an ACK with block id $g^s = 1$ and packet id $\eta^s = 1$. The uncoded packets, $a_1, a_2, ..., a_{G^s}$, must be stored in a buffer at the source for TCP ACK adaptation. *E.g.*, if an ACK for block id $g^s = 1$ and packet id $\eta^s = 1$ is received by the source, then the TCP adapter matches this ACK to packet a_1 and informs TCP that packet a_1 is ACK-ed. As long as TCP receiver transmits ACKs, the TCP clock moves, thus improving TCP goodput. After ACK with block and packet id is transmitted by TCP receiver, the packet is stored at the receiving buffer. When the last packet from a generation is received, then packets are decoded and passed to the application.

Note on the interaction of I^{2}NC and TCP. Although I^{2}NC can improve the performance of any traffic type transmitted over a wireless mesh, it has several characteristics that are particularly well suited for TCP. First, I^{2}NC masks wireless loss, which is well known to reduce TCP performance. It achieves this goal using local redundancy, which reduces the delay and thus TCP timeouts. Second, our queue management scheme is specifically tuned for TCP by balancing the rates of TCP flows coded together thus creating more network coding opportunities. Finally, the use of redundancy at intermediate node lends itself naturally to integration with end-to-end redundancy.

5.3.2 Operation of Intermediate Nodes

An intermediate node needs to take a number of actions when it receives (Alg. 5) or transmits (Alg. 6) a packet.

Receiving a packet and intra-session network coding

Buffer packets. A node i may receive a packet from higher layers or from previous hops. In the latter case, if the the received packet is inter-coded, it is decoded and the packet with destination this node is stored (or is passed to transport if it is the last hop). If it is not

Algorithm 5 Node i processes packet p_l from flow s.

```
 1: Read the information: packet $p_l$, from flow $s$, gen.size $G^s$
 2: Insert $p_l$ into the physical output queue $\mathcal{Q}_i$.
 3: Determine $\{h^*, k^*\}$ and label $p_l$ with $\{h^*, k^*\}$ pair and $s$
 4: Update virtual queue parameters: $q^s_{h^*,k^*}(t+1)$ and $q^{s'}_{h^*,k^*}(t+1)$
 5: Calculate $Q^s_{h^*,k^*}$ and $Q^s_i$
 6: $G^s_{h^*,k^*} = G^s_{h^*,k^*} + 1$
 7: if $G^s$ packets from flow $s$ are received at node $i$ then
 8:     Calculate the number of parities $P^{s,s}_{h,k}$, $P^{s',s}_{h,k}$
 9:     Create $P^{s,s}_{h,k}$ parities from $s$ and $P^{s',s}_{h,k}$ parities from $s'$
10:     Label all generated parities with $\{h, k\}$ pair and $s$
11: end if
```

the last hop, a packet $p_l \in \{p_1, p_2, ..., p_{G^s}\}$ is stored in the output queue $\mathcal{Q}_i$. In addition to the physical output queue $\mathcal{Q}_i$, the node i keeps track of several virtual queues $Q^s_{h,k}$ per (flow, hyperarc, code). The packet p_l is labeled with (h^*, k^*, s), which essentially indicates whether and how to code this packet according to the traffic splitting in Eq. (5.6): we pick $\{h^*, k^*\} = \arg\min_{h,k}\{H^s_{h,k}Q^s_{h,k}\}$, randomly breaking ties. Note that this labeling is local at the node, and does not introduce any transmission overhead.

Update Virtual Queue Sizes. Since packet p_l is selected to be transmitted with k^*-th network code over hyperarc h^*, the virtual buffers; $Q^s_{h^*,k^*}$ and $q^s_{h^*,k^*}$ should be updated.

$q^s_{h^*,k^*}$ is updated according to Eqs. (5.8) and (5.11). Specifically, for I^{2}NC-state: $q^s_{h^*,k^*}(t+1) = q^s_{h^*,k^*}(t) + 1/(1-\rho^s_{h^*})$ and $q^{s'}_{h^*,k^*}(t+1) = q^{s'}_{h^*,k^*}(t) + \rho^{s',s}_{h^*,k^*}$. For I^{2}NC-stateless: $q^s_{h^*,k^*}(t+1) = q^s_{h^*,k^*}(t) + 1/(1-\rho^s_{h^*})$ and $q^{s'}_{h^*,k^*}(t+1) = q^{s'}_{h^*,k^*}(t) + \rho^{s',s}_{h^*,k^*}/(1-\rho^{s'}_{h^*})$. Moreover, $Q^s_{h^*,k^*}$ is calculated according to Eq. (5.5) for I^{2}NC-state and Eq. (5.10) for I^{2}NC-stateless. Q^s_i is calculated according to Eq. (5.4). Then, the number of packets $G^s_{h^*,k^*}$ from the same generation that are allocated to h^*, k^* pair is incremented: $G^s_{h^*,k^*} = G^s_{h^*,k^*} + 1$. $G^s_{h,k}$ is set to 0 for each new generation.

Generate Parities. After G^s packets from a generation of flow s are received at node i, P^s parity packets are generated via intra-session network coding and labeled with information (s, h, k). There are two types of parities.

Algorithm 6 Node i transmits a packet.

```
1: Select $\{h^\dagger, k^\dagger\}$ pair that maximizes $R_h(\sum_{s\in\mathcal{S}_k} q_{h,k})$
2: Initialize: $\xi = \emptyset$
3: for $p_l \in \mathcal{Q}_i$ do
4:     if $p_l$ is labeled with $\{h^\dagger, k^\dagger\}$ AND flow id label of $p_l$ is different from $\forall p_{l'} \in \xi$ then
5:         if I$^2$NC-state AND $\xi \cup p_l$ is decodable OR I$^2$NC-stateless then
6:             Insert packet to $\xi$
7:         end if
8:     end if
9: end for
10: Network code (XOR) all packets in $\xi$
11: Broadcast the network coded packet over hyperarc $h^\dagger$
12: Update $q^s_{h^\dagger,k^\dagger}(t+1) = \{q^s_{h^\dagger,k^\dagger}(t) - 1\}^+$, $\forall s \in \mathcal{S}_k$
13: Re-calculate $Q_{h,k}$ and $Q^s_i$
```

- $P^{s,s}_{h,k} = \lceil G^s_{h,k}\rho^s_h/(1-\rho^s_h)\rceil$ parities are added on flow s's virtual queue to correct for loss during direct transmission to the next hop over hyperarc h.
- $P^{s',s}_{h,k} = \lceil G^s_{h,k}\rho^{s',s}_{h,k}, \forall s' \in \mathcal{S}_k\rceil$ parities are added on the virtual queues of other flows s' that are inter-session coded together with s. This is to to help the next hop for s' to decode despite losses on the overhearing link.

These numbers of parity packets are for I^{2}NC-state. For I^{2}NC-stateless $P^{s,s}_{h,k}$ is the same, but $P^{s',s}_{h,k} = \lceil G^s_{h,k}\rho^{s',s}_{h,k}/(1-\rho^s_h)\rceil$, *i.e.,* additional redundancy is used to protect parity packets from loss on the direct link.

Transmitting a packet and inter-session network coding

We consider the 802.11 MAC. When a node i accesses a channel, $\{h^\dagger, k^\dagger\}$ is chosen to maximize $R_h(\sum_{s\in\mathcal{S}_k} q_{h,k})$ according to Eq. (5.7), randomly breaking ties. Although the pair $\{h^\dagger, k^\dagger\}$ determines the hyperarc, code and flows to be coded together in the next transmission, the specific packets from those flows still need to be selected and coded. We call these packets the set ξ, and select them using the procedure specified in Alg. 6[8].

To achieve this, we first initialize the set of network coded packets $\xi = \emptyset$. For each packet

[8]The inter-session network coding header includes the number of coded packets together, next hop address, and the packet id's. Note that this header as well as the IP header of each packet is not coded, but included to the network coded packet as headers.

$p_l \in \mathcal{Q}_i$, check whether p_l is labeled with $\{h^\dagger, k^\dagger\}$. If it is, then we check whether its flow id label already exists in one of the packets in ξ, *i.e.,* another packet from the same flow has already been put in ξ. If not, there is one more check for I^{2}NC-state for decodability at the next hops of all packets in the network code, based on reports or estimates of overheard packets in the next hops, similarly to [10]. If the packet is decodable with some probability larger than a threshold (default value is 0.20) then, p_l is inserted to ξ. In I^{2}NC-stateless, the packet p_l is inserted to ξ without checking the decodability, which is ensured through the additional redundancy packets. This is the strength of I^{2}NC-stateless: it eliminates the need to exchange detailed state, which is costly and unreliable at high loss rates. After all packets in $\mathcal{Q}_i$ are checked, the labels (h, k, s) of the packets in ξ, inter-session network coding header is added, and coded (XORed) and broadcast over h.

After a coded packet is transmitted, the virtual queues are updated as; $q^s_{h^\dagger,k^\dagger}(t+1) = \{q^s_{h^\dagger,k^\dagger}(t) - 1\}^+$, $\forall s \in \mathcal{S}_k$. The queues $Q_{h^\dagger,k^\dagger}$ and Q^s_i are calculated according to Eqs. (5.5), (5.10), (5.4)[9].

Keeping Track and Exchanging State Information

For I^{2}NC-state, intermediate nodes need also to keep track and exchange information with each other, so as to enable the intra- and inter-session network coding modules to make their redundancy and coding decisions. An approach similar to COPE is used: ACKs are sent after the reception and successful decoding of a packet. Information about overheard packets is piggy-backed on the ACKs. With I^{2}NC-stateless, we only need neighbors to exchange information about the loss rates (through infrequent control packets) at the neighboring nodes.

[9]Note that I^{2}NC may cause reordering at the receiver, but since we already implemented intra-session network coding, and made TCP receiver sequence agnostic in this term, out of packet delivery is not a problem for TCP.

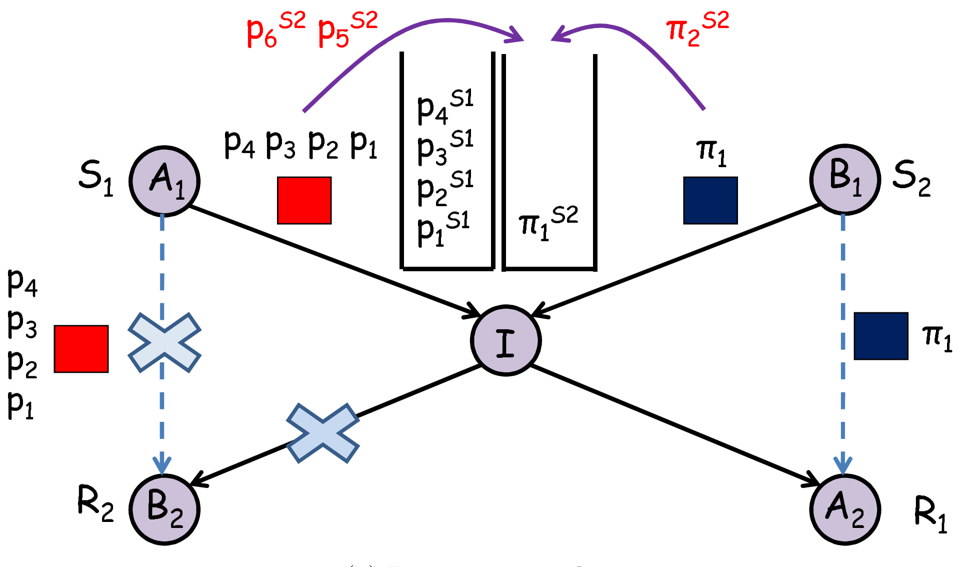
Congestion Control and Queue Management.

Upon congestion at node i, the per-flow queue sizes Q_i^s are compared and the last packet from flow s having the largest Q_i^s is dropped from the queue; in case of tie, an incoming packet is dropped. This eventually balances the rates of flows coded together, increases inter-session network coding opportunities, and improves TCP performance. For details on this network coding-aware queue management scheme, the reader is referred to [20].

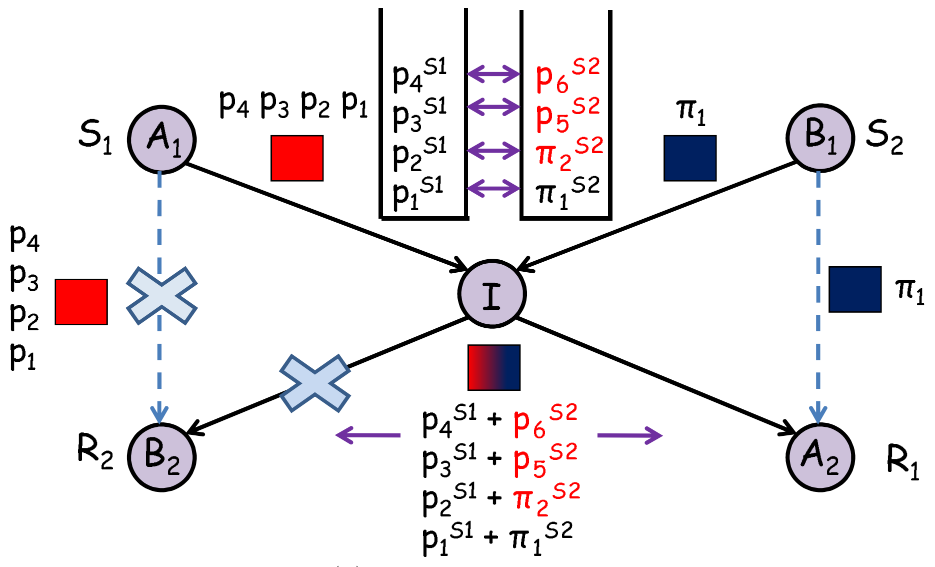
Example 12 Let us re-visit the X-topology from Fig. 5.1, shown again for convenience in Fig. 5.3, and illustrate how we perform intra- and inter-session network coding under scheme I^{2}NC-stateless. The loss probabilities over the direct $(I - B_2)$ and overhearing $(A_1 - B_2)$ links are assumed 0.5 and 0.25.

In Fig. 5.3(a), we describe intra-session network coding. Let us assume the generation size of S_1 is $G^{S_1} = 4$ and S_2 is $G^{S_2} = 1$. The packets transmitted by A_1, B_1 are p_1, p_2, p_3, p_4 and π_1 for flows S_1 and S_2, respectively. Note that there is only one option for inter-session network coding, *i.e.,* to XOR packets from the 2 flows, thus only one possible network code $k = 1$ over hyperarc $h = (I, \{B_2, A_2\})$. All packets are labeled with this information and their flow ids. The labeled packets are $p_1^{s_1}, p_2^{s_1}, p_3^{s_1}, p_4^{s_1}$ and $\pi_1^{s_2}$. Parities are generated as follows. Since $G_{I,\{B_2,A_2\}}^{S_1} = 4$ and $G_{I,\{B_2,A_2\}}^{S_2} = 1$, the number of parities is $P_{I,\{B_2,A_2\}}^{S_1,S_1} = 0$, $P_{I,\{B_2,A_2\}}^{S_2,S_1} = 0$, $P_{I,\{B_2,A_2\}}^{S_2,S_2} = 1$ (thus generating one parity from flow S_2 and labeling it with s_2, *i.e.*, $\pi_2^{S_2}$), and $P_{I,\{B_2,A_2\}}^{S_1,S_2} = 2$ (thus generating two parities from flow S_1 and labeling them with S_2, *i.e.*, $p_5^{S_2}, p_6^{S_2}$).

In Fig. 5.3(b), we describe inter-session network coding. Node I performs inter-session network coding and transmits packets according to Alg. 6: it XORs packets from the two queues, for S_1, S_2, and broadcasts over the hyperarc $(I, \{B_2, A_2\})$. In particular, it transmits the following packets: $p_1^{s_1} \oplus \pi_1^{s_2}$, $p_2^{s_1} \oplus \pi_2^{s_2}$, $p_3^{s_1} \oplus p_5^{s_2}$, and $p_4^{s_1} \oplus p_6^{s_2}$. Node A_2 receives and decodes all the packets. Node B_2 receives 3 packets on the average over overhearing link



(a) Intra-session coding



(b) Inter-session coding

Figure 5.3: Example of coding (under scheme I^{2}NC-stateless) at intermediate node I in the X-topology. There is loss only on two links: the direct link $I - B_2$ (w.p. 0.5) and the overhearing link $A_1 - B_2$ (w.p. 0.25).

$A_1 - B_2$ and receives 2 packets over direct link $I - B_2$. Five received packets allows B_2 to decode all five packets $p_1, p_2, p_3, p_4, \pi_1$, so π_1 is successfully decoded. $\square$

5.4 Performance Evaluation

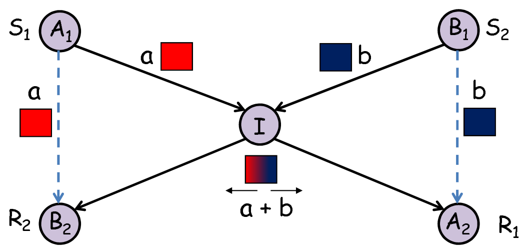
5.4.1 Simulation Setup

We used the `GloMoSim` simulator [77], which is well suited for simulating wireless environments. We considered various *topologies*: the classic X topology, shown in part of Fig. 5.1 and repeated in Fig. 5.4(a); the classic cross-topology with 4 end-nodes generating bi-directional traffic, with one relay shown in Fig. 5.4(b); the wheel topology shown in Fig. 5.4(c); and the multi-hop topology shown in Fig. 5.1. In X, cross, and wheel topologies, the intermediate node I is placed in a center of of circle with radius $90m$ over $200m \times 200m$ terrain and all other nodes A_1, B_1 and etc. are placed around the circle. In the multi-hop topology of Fig. 5.1, two X topologies are cascaded and the distance between consecutive nodes is set to $90m$. The topology is over a $800m \times 300m$ terrain.

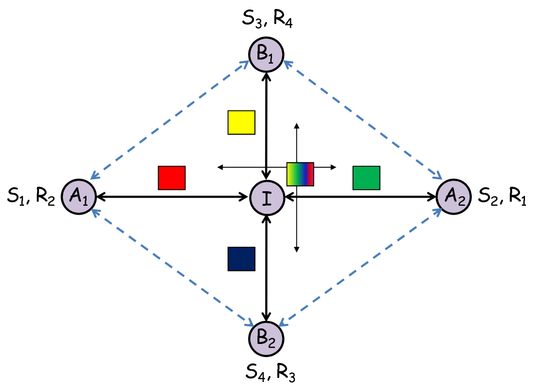
We also considered various *traffic scenarios*: FTP/TCP and CBR/UDP[10]. IEEE 802.11b is used in the *MAC layer*, with the addition of the pseudo-broadcasting mechanism, as in COPE [10]. In terms of *wireless channel*, we simulated the two-ray path loss model and a Rayleigh fading channel with average channel loss rates $0, 20, 30, 40, 50$ %. We have repeated each $60sec$ simulation for 10 seeds. Channel capacity is $1Mbps$, the buffer size at each node is set to 100 packets, packet sizes are set to $500B$, and the block size for UDP flows is set to 15 packets.

We compare our schemes (I^{2}NC-state and I^{2}NC-stateless) to no network coding (*noNC*), and

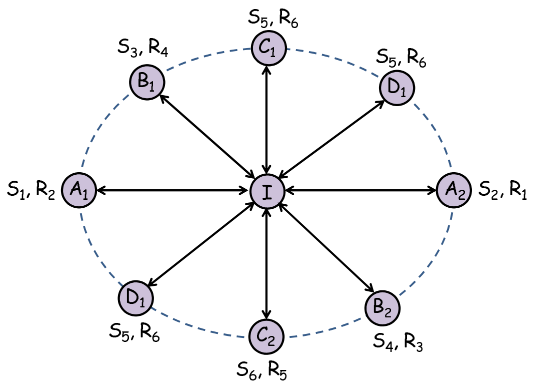
[10]TCP and CBR flows start at random times within the first $5sec$ and are on until the end of the simulation which is $60sec$.



(a) X topology



(b) Cross topology



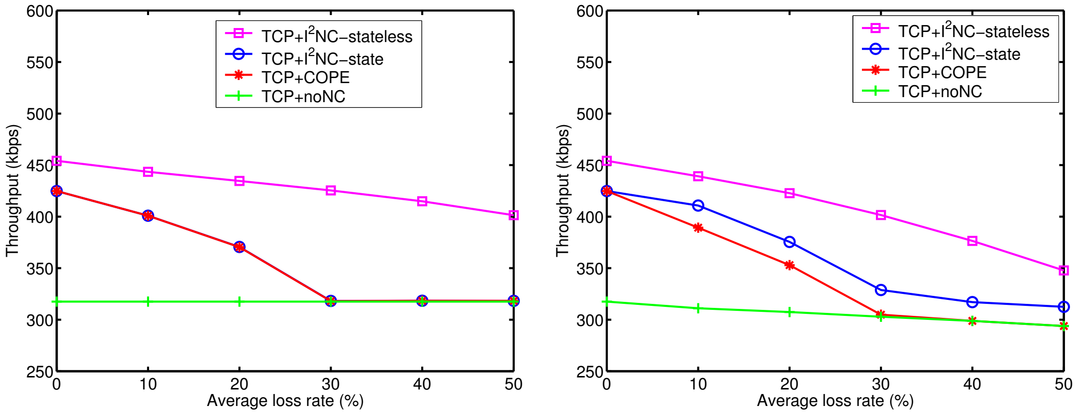
(c) Wheel topology

Figure 5.4: Topologies under consideration. (a) X topology. Two unicast flows, S_1, R_1, and S_2, R_2, meeting at intermediate node I. (b) Cross topology. Four unicast flows, S_1, R_1, S_2, R_2, S_3, R_3, and S_4, R_4, meeting at intermediate node I. I receives packets A_1, A_2, B_1, B_2. (c) Wheel topology. Multiple unicast flows S_1, R_1, S_2, R_2, etc., meeting at intermediate node I. I opportunistically combine the packets and broadcast.

COPE [10], in terms of total transport-level throughput (added over all flows).

5.4.2 Simulation Results

X topology - loss on some links. In Fig. 5.5, we present results for 2 TCP flows over Example 11, in order to illustrate the key intuition of our approach. Consider, for the moment, that loss occurs only on one link, either (a) the overhearing link $A_1 - B_2$ or (b) the direct



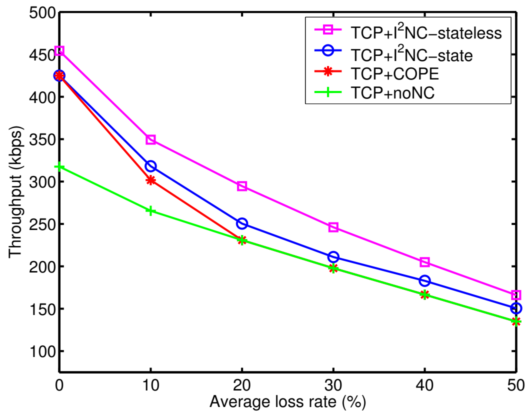
(a) Loss only on overhearing link (b) Loss only on direct link

Figure 5.5: X topology in Fig. 5.4(a). We show the total TCP throughput (added over two flows) vs. link loss rate, for two specific loss patterns. Loss happens only on one link, either: (a) the overhearing link $A_1 - B_2$ or (b) the direct link $I - B_2$. All other links are lossless.

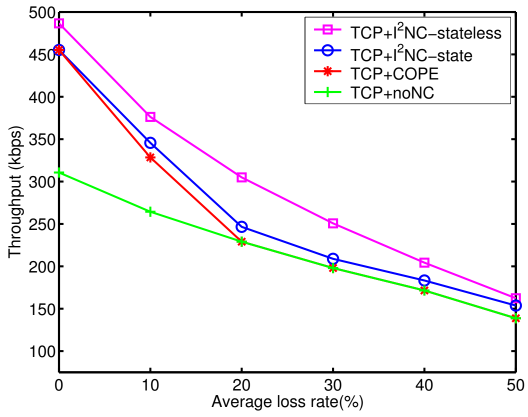
link $I - B_2$.

The first case is depicted in Fig. 5.5(a). Loss on the overhearing link does not affect the uncoded streams, thus TCP+noNC. TCP+I^{2}NC-state and TCP+COPE are identical: I^{2}NC-state introduces redundancy for loss on the direct link and does inter-session network coding as with COPE. TCP+I^{2}NC-stateless outperforms other schemes over the entire loss range. For example, if there is no loss, I^{2}NC-stateless still brings the benefit of eliminating ACK packets, thus using the medium more efficiently. When the loss rate increases, the improvement of I^{2}NC-stateless becomes significant, reaching up to 30%. The reason is that at high loss rates, I^{2}NC-state and COPE do not have reliable knowledge of the decoding buffers of their neighbors and cannot do network coding efficiently. In contrast, I^{2}NC-stateless does not rely on this information, but on the loss rate of the overhearing link to code together part of the flows. In the discussion of Example 11, we mentioned that even at 50% loss rate, 16.6% improvement can be achieved via network coding. Here we see this improvement (around 13%) as well as the the additional benefit of eliminating ACK packets (around 12%). Note that the total improvement is around 25%.

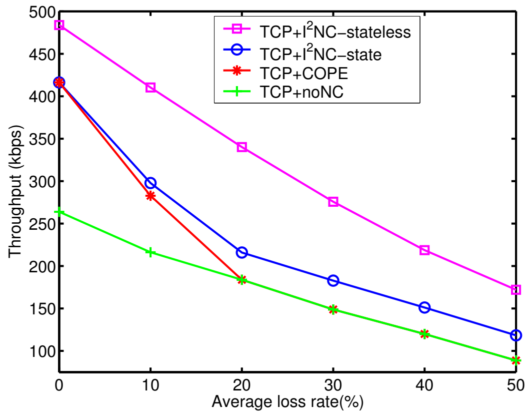
The second case is depicted in Fig. 5.5(b). TCP+I^{2}NC-stateless significantly outperforms



(a) X topology (shown in Fig. 5.4(a))



(b) Cross topology (shown in Fig. 5.4(b))



(c) Multi-hop topology (shown in Fig.5.1)

Figure 5.6: Total TCP throughput vs. average loss rate (for ease of presentation, the same loss rate is assumed on all links) in three different topologies.

all alternatives again. TCP+I^{2}NC-state outperforms TCP+COPE in this case, because it corrects errors on the direct link, reduces the number of re-transmissions and uses the channel more efficiently.

Various topologies - loss on all links. Fig. 5.6 presents simulation results for TCP traffic over various topologies (X, cross, and the multi-hop topology), considering the more general case where loss happens on all links. For ease of presentation, here we report only the results when all links have the same loss probability.

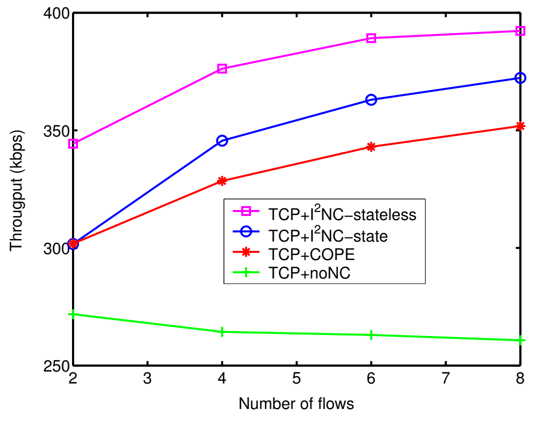


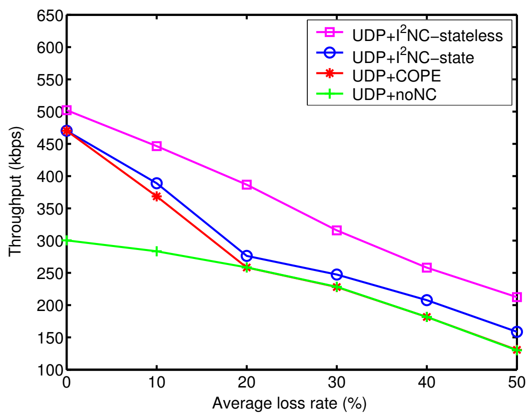
Figure 5.7: Wheel topology shown in Fig. 5.4(c) with increasing number of flows. Loss rate on all links is set to 10%.

Fig. 5.6(a) shows the results for the X topology. At low-medium loss rates (10%-30%), there is only a moderate gain from I^{2}NC-stateless. The reason is that I^{2}NC-state and COPE are still able to do some network coding, in conjunction with link level ARQ. At higher loss rates, network coding opportunities reduce. I^{2}NC-stateless brings significant benefit thanks to both network coding and eliminating the ACKs.

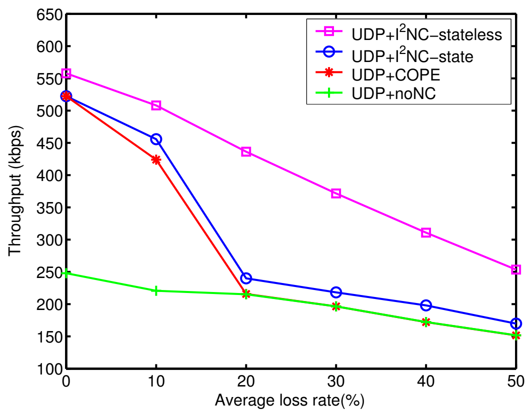
Fig. 5.6(b) shows the results for the cross topology. The improvement of TCP+I^{2}NC-stateless is higher compared to the X topology. This is because, here, there are more network coding opportunities for I^{2}NC-stateless to exploit.

Fig. 5.6(c) presents the results for the multi-hop topology in Fig. 5.1. The improvement of TCP+I^{2}NC-state is higher than in the X and cross topologies, especially at higher loss rates. This is because intra-session coding, employed by I^{2}NC-state, reduces the dependency on link level ARQ. Especially in this multi-hop topology, the end-to-end residual loss rate increases with the number of hops. Intra-session network coding overcomes this, thus increasing TCP throughput. The improvement of I^{2}NC-stateless is even more significant for this topology, because the benefit of eliminating ACKs is more pronounced with larger number of hops.

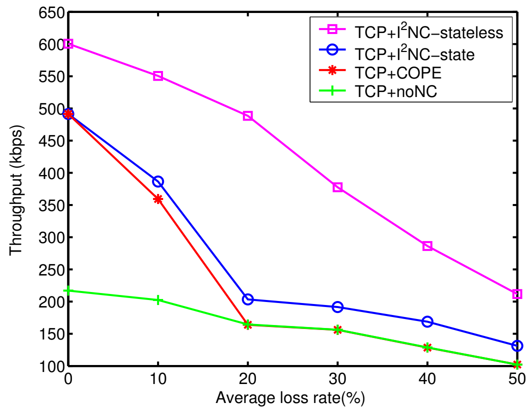
We also performed simulations with increasing number of flows, *i.e.,* nodes in wheel topology in Fig. 5.4(c). It is seen in Fig. 5.7 that the total throughput achieved by network coding



(a) X topology (shown in Fig. 5.4(a))



(b) Cross topology (shown in Fig. 5.4(b))



(c) Multi-hop topology (shown in Fig.5.1)

Figure 5.8: Total UDP throughput vs. average loss rate (the same loss rate is assumed on all links) in three different topologies.

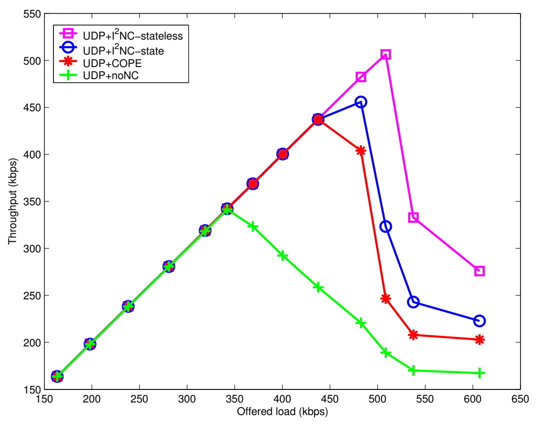


Figure 5.9: Cross topology shown in Fig. 5.4(b). Loss rate on all links is set to 10%.

schemes increases with the increasing number of flows. When the number of flows increases, the probability of network coding at the intermediate node I increases. More network coding opportunities leads to higher throughput.

UDP traffic. We repeated the simulations for the three topologies for the case that there is loss over all links. The results are presented in Fig. 5.8.

Fig. 5.8(a) presents the results for the X topology. The improvement of UDP+I^{2}NC-stateless is up to 60% as compared to UDP+noNC. This is significantly higher than the improvement of TCP+I^{2}NC-stateless and the optimal scheme (in which the improvement is 33.3%). The reason is the MAC gain as explained in [10]. We present the results for the load at which the system saturates. At this load, UDP+noNC is already saturated, several packets are dropped from the buffers, and they do not arrive to their receivers. This reduces the throughput of noNC, while network coding schemes still handle the traffic created by the load. Notice that even at 50% loss rate, UDP+I^{2}NC-stateless improves over UDP+noNC by 40%, which is significant.

Fig. 5.8(b) presents the results for the cross topology. In this topology, the improvement of network coding is very large. When there is no loss, the improvement is around 250%. The effectiveness of UDP+I^{2}NC-stateless is also significant in this topology: at 50% loss rate the improvement of UDP+I^{2}NC-stateless over UDP+noNC is 70%.

Fig. 5.8(c) presents the results for multi-hop topology, and shows similar results.

Fig. 5.9 presents the throughput vs. the offered load in cross topology shown in Fig. 5.4(b). At low loads, all schemes perform equally, because even if network coding is possible, offered load is limited and the transmission medium stays idle at some instants, and network coding benefit is not seen. When the offered load increases, noNC saturates earlier than network coding schemes which handle higher loads better. Specifically, network coding schemes reduce queue sizes as compared to noNC by increasing throughput (hence drain rate seen by each queue). After the saturation point, the throughput decreases with increasing load, because many packets are dropped from buffers, especially from I, and the time slots that are used to transmit packets which are dropped by I are wasted. It is seen that I^{2}NC-state and I^{2}NC-stateless arrive to their saturation points at higher loads than *COPE*, because our algorithms handle load better than COPE with their error correction mechanisms.

5.4.3 Numerical Results

We consider the X and cross topologies shown in Figs. 5.4(a) and 5.4(b). In the X topology, A_1 transmits packets to A_2 via I with rate x_1, and B_1 transmits packets to B_2 via I with rate x_2. In the cross topology, A_1 transmits packets to A_2 with rate x_1, A_2 transmits packets to A_1 with rate x_2, B_1 transmits packets to B_2 with rate x_3, and B_2 transmits packets to B_1 with rate x_4. All transmissions are via I. In both topologies, the data rate of each link is set to 1 packet/transmission. We compare our schemes I^{2}NC-state and I^{2}NC-stateless with noNC which is also formulated in a NUM framework without any network coding constraints.

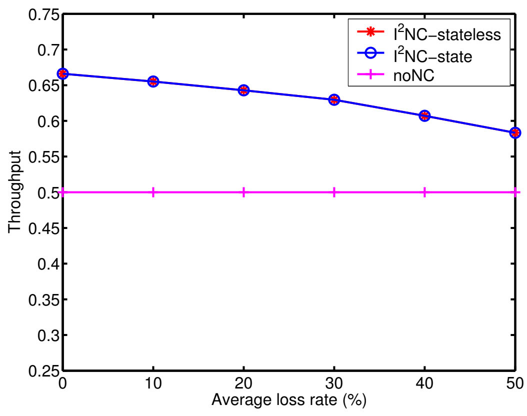
Fig. 5.10 shows the total throughput; $x_1 + x_2$ for I^{2}NC-state, I^{2}NC-stateless and noNC for X topology. Fig. 5.10(a) shows the results when there is loss only on the overhearing link $A_1 - B_2$. It is seen that the throughput of noNC is flat with increasing loss rate, because it is not affected by the loss rate on the overhearing link. I^{2}NC-state and I^{2}NC-stateless improve over noNC, because they exploit network coding benefit. When the loss rate increases, the improvement reduces, because B_2 overhears only part of the data transmitted by A_1. Although the improvement decreases with increasing loss rate, it is still significant, *e.g.,* 16.6% at 50% loss rate. Note that Fig. 5.10(a) is the counterpart of the simulation results presented in Fig. 5.5(a). It is seen that TCP+I^{2}NC-stateless in Fig. 5.5(a) shows similar performance as I^{2}NC-stateless in Fig. 5.10(a). This shows the effectiveness of I^{2}NC-stateless in a realistic simulation environment. On the other hand, the throughput of TCP+I^{2}NC-stateless and

TCP+COPE decreases with increasing loss rate in Fig. 5.5(a), while I^{2}NC-state improves throughput significantly even at higher loss rates Fig. 5.10(a). I^{2}NC-state and COPE need to know the state of neighbors which gets corrupted and/or delayed at high loss rates in simulations. Therefore, the throughput improvement of I^{2}NC-state and COPE is limited at high loss rates in Fig. 5.5(a).

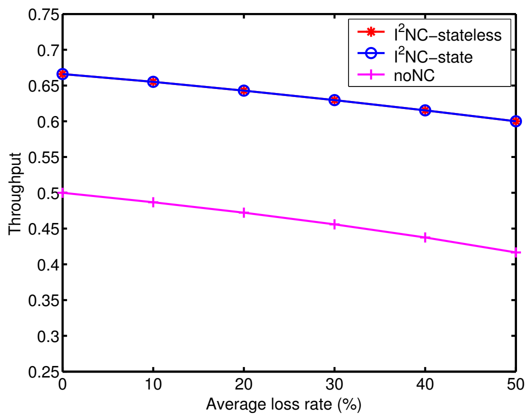
Fig. 5.10(b) shows the results when there is loss only on the direct link $I - B_2$. It is seen that I^{2}NC-state and I^{2}NC-stateless improve over noNC significantly at all loss rates. It is also interesting to note that at 50% loss rate, I^{2}NC-state and I^{2}NC-stateless improve over noNC by 44% which is even higher than in the no loss case (33%). In the optimal solution, the throughput values are $x_1 = 0.4$ and $x_2 = 0.2$. In this case, in the downlink $I - B_2$, data part of x_2 with rate 0.2 and the parity part with rate 0.2 (considering loss rate 50%) are combined with x_1. This means that our schemes combine both parity and data parts of a flow with other flows and this improves the throughput significantly. This is one of the important contributions of our NUM formulations.

Fig. 5.10(c) shows the results when there is loss on links $A_1 - B_2$ and $I - B_2$. It is seen that I^{2}NC-state improves the throughput significantly while the improvement of I^{2}NC-stateless reduces to 0 with increasing loss rate. The reason is that, I^{2}NC-stateless is a more conservative scheme as compared to I^{2}NC-state in the sense that it eliminates the perfect knowledge on antidotes. Yet, it still improves the throughput significantly, *e.g.,* it improves over noNC by 22% at 30% loss rate.

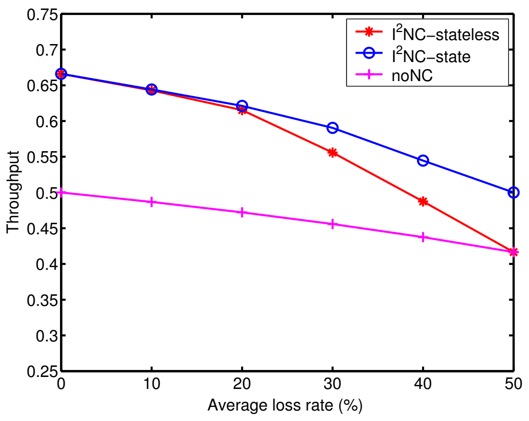
Fig. 5.10(d) shows the results when there is loss on all links. It is seen that I^{2}NC-state and I^{2}NC-stateless improve over noNC significantly at all loss rates. Note that throughput of I^{2}NC-stateless reduces to that of noNC at 50% loss rate in Fig. 5.10(c). The reader might wonder why we do not see such behavior in Fig. 5.10(d). The reason is that since there is loss over link $A_1 - I$ as well as $A_1 - B_2$, the number of parities added by A_1 to correct losses over link $A_1 - I$ also increases the number of overheard packets at B_2. Therefore, I^{2}NC-



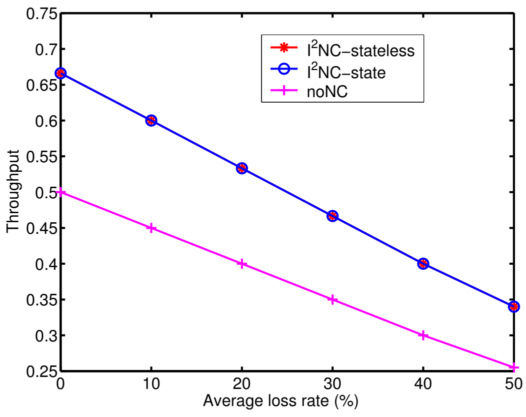
(a) Loss only on overhearing link $A_1 - B_2$



(b) Loss only on direct link $I - B_2$



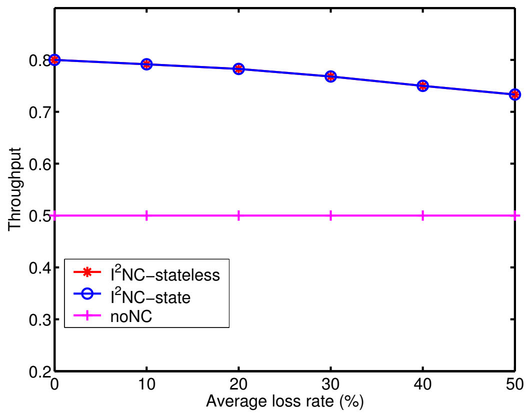
(c) Loss on links $A_1 - B_2$ and $I - B_2$



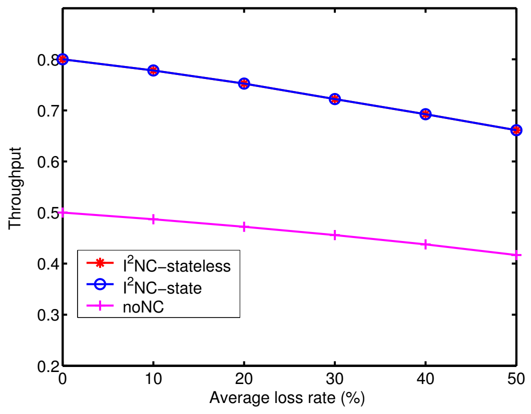
(d) Loss on all links

Figure 5.10: X topology. Throughput vs. loss rate (the same loss rate is assumed on $A_1 - B_2$ and $I - B_2$ in (c), and the same loss rate is assumed on all links in (d)).

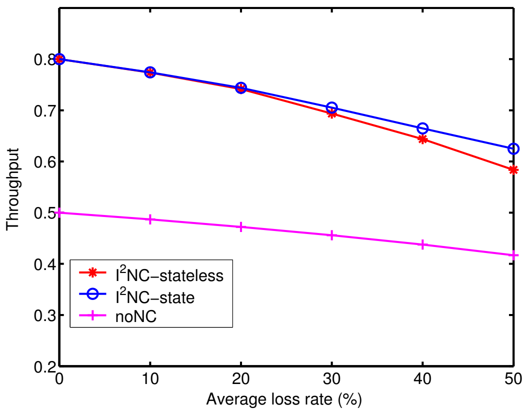
stateless does not add redundancy at node I for both $A_1 - B_2$ and $I - B_2$ as in Fig. 5.10(c), but adds redundancy only for loss on link $I - B_2$. This improves the performance of I^{2}NC-stateless . Note that the counterpart of these results are presented in Fig. 5.6(a). It is seen that the throughput improvement of I^{2}NC-stateless over noNC at 50% loss rate is around 30% in Fig. 5.10(d). As compared to this, the improvement of TCP+I^{2}NC-stateless over noNC is limited in Fig. 5.6(a), because, in simulations, the block size is limited and fixed, and the scheduling is not perfect (we consider IEEE 802.11). Yet, the throughput improvement of TCP+I^{2}NC-stateless over noNC is around 20% in Fig. 5.6(a), which is significant.



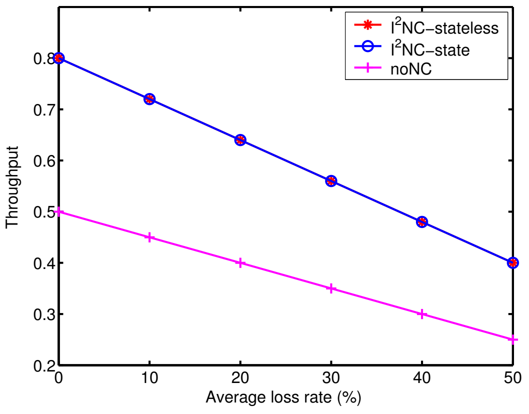
(a) Loss only on overhearing link $A_1 - B_2$



(b) Loss only on direct link $I - B_2$



(c) Loss on links $A_1 - B_2$ and $I - B_2$



(d) Loss on all links

Figure 5.11: Cross topology. Throughput vs. loss rate (the same loss rate is assumed on $A_1 - B_2$ and $I - B_2$ in (c), and the same loss rate is assumed on all links in (d)).

Fig. 5.11 shows the total throughput; $x_1 + x_2 + x_3 + x_4$ for I^{2}NC-state, I^{2}NC-stateless and noNC for the cross topology shown in Fig. 5.4(b) for different loss patterns. It is seen that the results are similar to the ones in Fig. 5.10. One difference is that the throughput improvement of network coding schemes is higher, *i.e.*, up to 80%, because there are more network coding opportunities in the cross topology.

5.5 Numerical Results: Convergence

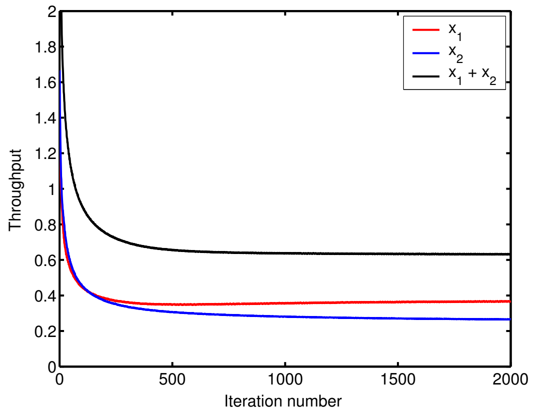
We consider again the X and cross topologies shown in Figs. 5.4(a) and 5.4(b). In the X topology, A_1 transmits packets to A_2 via I with rate x_1, and B_1 transmits packets to B_2 via I with rate x_2. In the cross topology, A_1 transmits packets to A_2 with rate x_1, A_2 transmits packets to A_1 with rate x_2, B_1 transmits packets to B_2 with rate x_3, and B_2 transmits packets to B_1 with rate x_4. All transmissions are via I. In both topologies, the data rate of each link is set to 1 packet/transmission and the loss rate is set to 30%.

In Figs. 5.12 and 5.13, we present the throughput vs. the iteration number for the X topology at different loss patterns for I^{2}NC-state and I^{2}NC-stateless, respectively. Each figure shows the convergence of x_1, x_2, and $x_1 + x_2$ to their optimum values. *E.g.*, $x_1 + x_2$ converges to 0.59 in Fig. 5.12(c) and $x_1 + x_2$ converges to value 0.55 in Fig. 5.13(c).

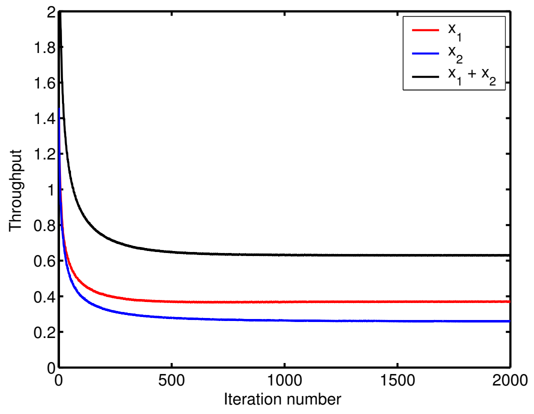
Fig. 5.14 and 5.15 present the throughput vs. the iteration number for the cross topology at different loss patterns for I^{2}NC-state and I^{2}NC-stateless, respectively. We see similar convergence results. Specifically, each flow rate, x_1, x_2, x_3, x_4, and the total throughput, $x_1 + x_2 + x_3 + x_4$ converge to their optimum values.

5.6 Summary

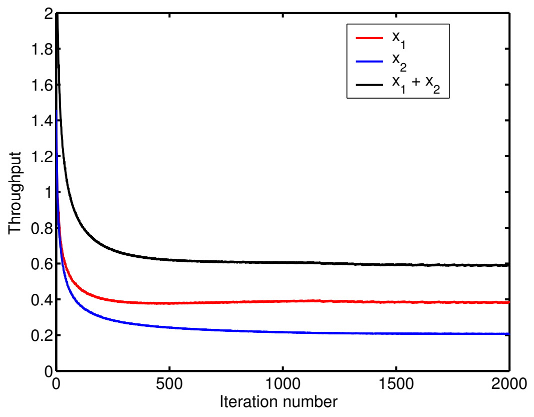
In this chapter, we proposed I^{2}NC: a one-hop intra- and inter-session network coding approach for wireless networks. I^{2}NC is built to improve network coding performance over lossy wireless networks. I^{2}NC has two critical improvements: it is resilient to loss and it does not need to rely on exact the knowledge of the state of the neighbors. Our design is grounded on a NUM formulation and its solution. Simulations in GloMoSim demonstrate significant throughput gain of our approach compared to baselines.



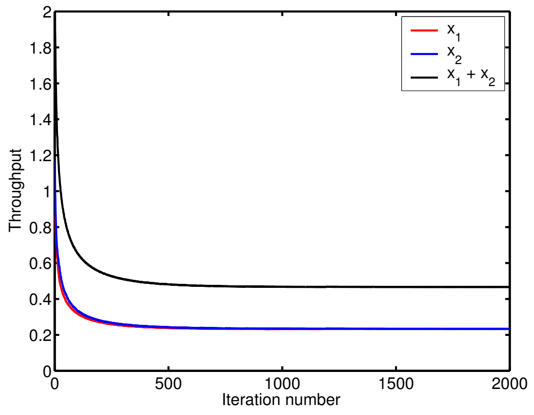
(a) Loss only on overhearing link $A_1 - B_2$



(b) Loss only on direct link $I - B_2$

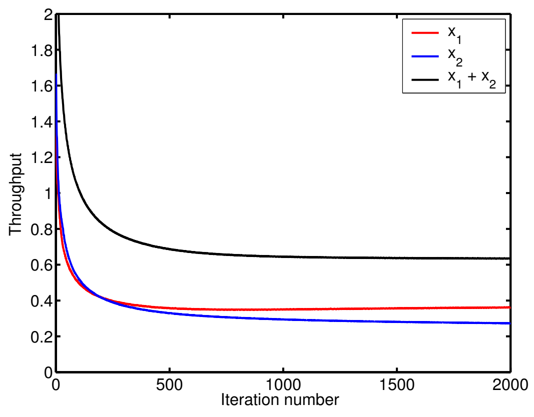


(c) Loss on links $A_1 - B_2$ and $I - B_2$

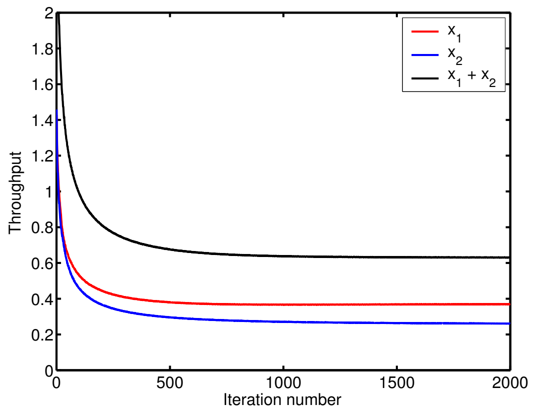


(d) Loss on all links

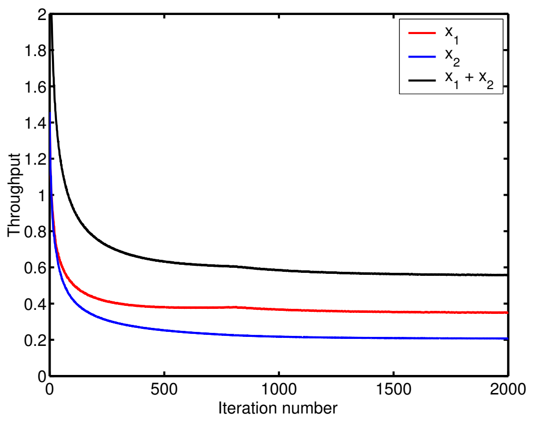
Figure 5.12: X topology. Convergence of x_1, x_2, and $x_1 + x_2$ for I^{2}NC-state. Loss rate is 30%.



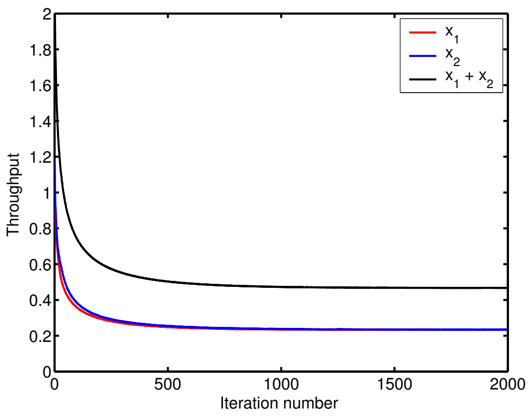
(a) Loss only on overhearing link $A_1 - B_2$



(b) Loss only on direct link $I - B_2$

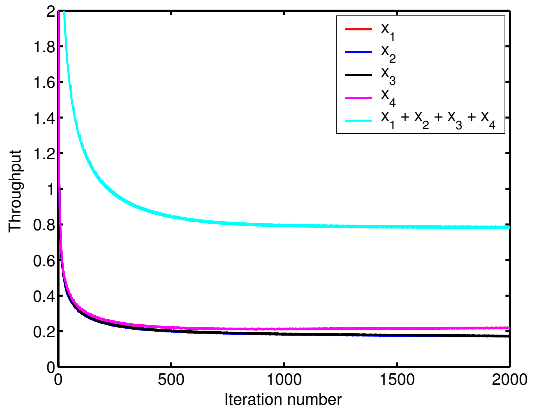


(c) Loss on links $A_1 - B_2$ and $I - B_2$

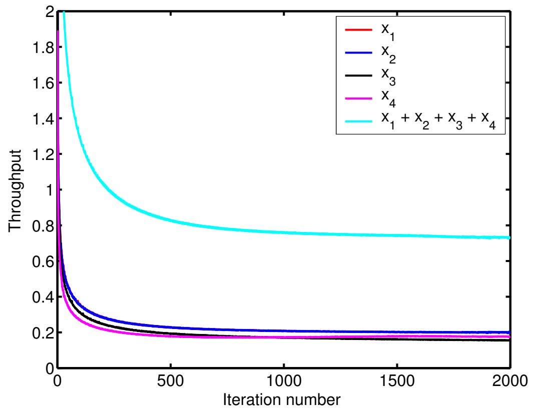


(d) Loss on all links

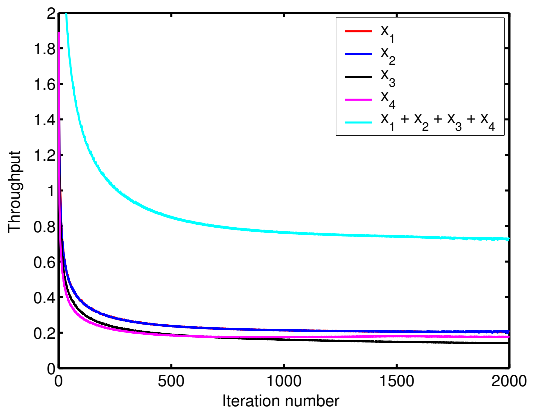
Figure 5.13: X topology. Convergence of x_1, x_2, and $x_1 + x_2$ for I^{2}NC-stateless. Loss rate is 30%.



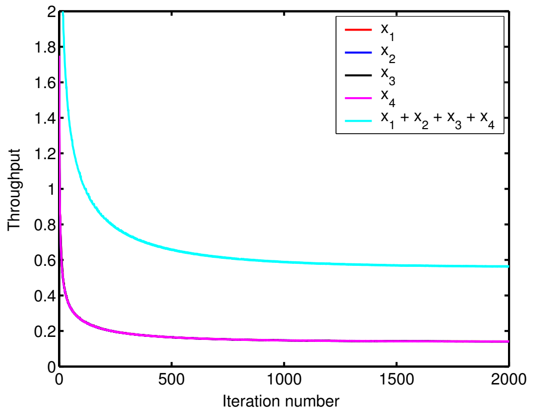
(a) Loss only on overhearing link $A_1 - B_2$



(b) Loss only on direct link $I - B_2$

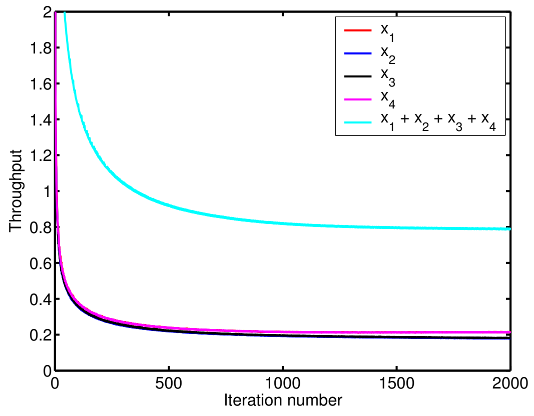


(c) Loss on links $A_1 - B_2$ and $I - B_2$

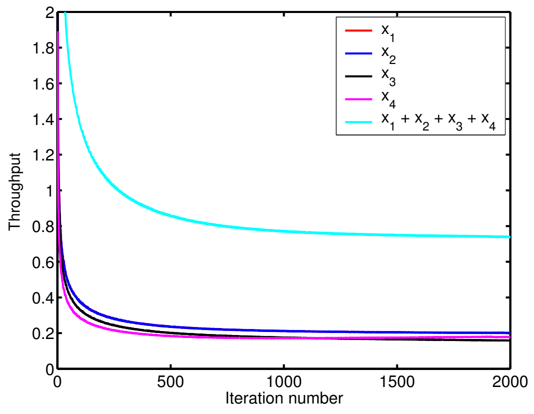


(d) Loss on all links

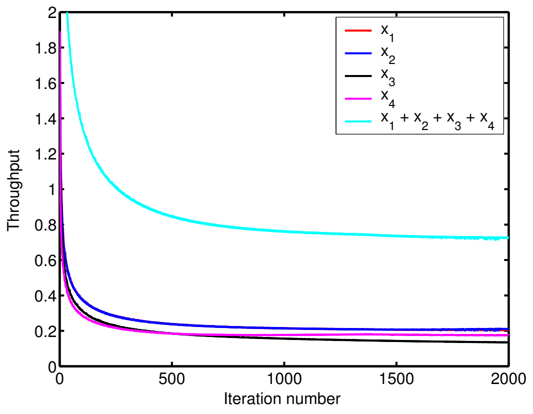
Figure 5.14: Cross topology. Convergence of x_1, x_2, x_3, x_4, and $x_1 + x_2 + x_3 + x_4$ for I^{2}NC-state. Loss rate is 30%.



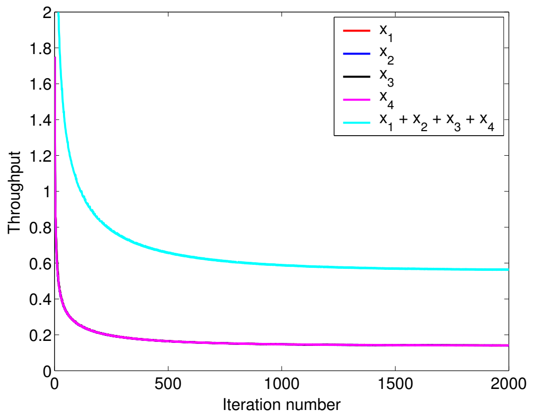
(a) Loss only on overhearing link $A_1 - B_2$



(b) Loss only on direct link $I - B_2$



(c) Loss on links $A_1 - B_2$ and $I - B_2$



(d) Loss on all links

Figure 5.15: Cross topology. Convergence of x_1, x_2, x_3, x_4, and $x_1 + x_2 + x_3 + x_4$ for I^{2}NC-stateless. Loss rate is 30%.

Chapter 6

Conclusions

In this chapter, we conclude the dissertation by summarizing our contributions and presenting our final remarks.

6.1 Contributions

Below, we highlight the contributions of the thesis.

6.1.1 Video Streaming over Coded Wireless Networks

Network Coding-Aware Video Streaming Algorithms: We designed video-aware network coding schemes that take into account both the decodability of network codes by several receivers and the unequal importance of video packets. We formulated the problem in a rate-distortion optimization framework and proposed distributed solution (NC-RaDiO). We also proposed video-aware network coding algorithms (NCVD and NCV) which have near optimal performance and have less computational complexity.

In our design, we first considered the fact that different packets in the same multimedia stream may have different contributions to video quality. One important challenge with network coding is that it has been agnostic to the content of the packets that are coded together. Our network coding-aware video streaming algorithms (NC-RaDiO, NCVD, and NCV) take into account importance of video packets, importance of video flows, and strict delay requirements of video streaming. In a sense, our schemes combine two orthogonal aspects of packet scheduling: (i) network coding to mix packets from different flows and increase throughput and (ii) radio-distortion optimized streaming of packets within the same stream to maximize video quality.

Performance Evaluation: We evaluated the performance of the proposed schemes (NC-RaDiO, NCVD, and NCV) in terms of video quality and network throughput in a wide range of scenarios. Simulation results showed that the proposed schemes improve video quality up to $5dB$ compared to baseline schemes. Furthermore, they significantly improve the application-level throughput while achieving the same or similar levels of MAC throughput.

6.1.2 Rate Control and Scheduling over Coded Wireless Networks

Necessity of Network Coding-Aware Rate Control and Scheduling over Coded Wireless Networks: We argued the necessity of making rate control and scheduling network-coding aware over coded wireless networks. We gave the main intuition and also compared NC-aware vs. NC-unaware, both optimal and practical schemes. In addition to the formulations, we presented simulation results for an illustrative example with two cross flows, which captures the main intuition and should be a building block of any large scenario with cross-flows and network coding. In general, the benefit from the network coding awareness depends on (i) the network coding opportunities (ii) the conflicts of network coded flows with other flows and (iii) the link rates where (i) and (ii) depend on the topology and traffic scenario.

Video: We showed that video rate control naturally fits in network utility maximization framework, with the additional complication that time varying video rate and utility affect the network coding opportunities and the total achieved utility; to deal with time variability, we proposed video rate control over an appropriate time interval so as to optimize the utility vs. delay tradeoff. We formulated the problem, and showed that it has a distributed solution and provided insights on the interaction of rate control and network coding for video streaming.

TCP: We showed that network coding performance of TCP over coded wireless networks is improved with cross-layer design. The key intuition was to eliminate the rate mismatch between flows that are coded together through a synergy of rate control and queue management. We formulated congestion control as a NUM problem and derived a distributed solution.

Protocol Design: Motivated by the structure of the NUM solution, we proposed minimal modifications to queue management to make it network coding-aware, while TCP and MAC protocols remained intact. To the best of our knowledge, our work is the first, to take the step from theory (optimization) to practice (protocol design), specifically for the problem of congestion control over inter-session network coding. We proposed implementation changes, which have a number of desired features: they are justified and motivated by analysis, they perform well (double the throughput in simulations), and they are minimal (only queue management is affected, while TCP and MAC remain intact).

Performance Evaluation: Simulation results showed that the proposed NCAQM scheme doubles TCP performance compared to baseline schemes and achieves near-optimal performance. We also extended the NUM formulation and solution to multi-hop network coding and we confirmed convergence through numerical calculations.

6.1.3 Performance of Network Coding over Lossy Wireless Networks

I^2NC: Combining inter- and intra-session network coding: We proposed I^2NC: a one-hop inter- and intra-session network coding scheme for wireless networks. I^2NC builds on one-hop opportunistic network coding [10] and improves it in two aspects: it is resilient to loss and it does not need to rely on exact the knowledge of the state of the neighbors. Our design is grounded on a NUM formulation and its solution. In this scheme, we considered how to gracefully combine intra- and inter-session network coding at intermediate nodes.

Adding redundancy: We showed how to adjust the amount of redundancy after taking into account the loss on the direct and overhearing links. We implemented the intra-session network coding functionality.

Inter-session network coding: We designed algorithms that make the decision of what percentage of of flows should be coded together by taking into account the loss characteristics on the direct and overhearing links. We implemented this and other functionalities (e.g., queue management) performed with or after inter-session network coding.

Synchronization: We proposed two schemes: I^2NC-state, which needs to know the state (i.e., overheard packets) of the neighbors; and I^2NC-stateless, which only needs to know the loss rate of links in the neighborhood.

Performance Evaluation: Simulations in GloMoSim and numerical results demonstrated significant throughput gain of our proposed schemes; I^2NC-state and I^2NC-stateless as compared to baselines schemes.

6.2 Final Remarks

In this thesis, we studied the design and optimization of algorithms and network protocols to solve some practical problems in coded wireless networks. We first studied video streaming over coded wireless networks and developed video-aware network coding algorithms. Second, we showed the importance of network coding-awareness in rate control and scheduling over coded wireless networks, and we optimized rate control protocols for video streaming and TCP. Finally, we studied the performance of network coding over lossy wireless networks and developed loss-aware network coding algorithms along with optimal rate control protocols.

We believe that our algorithms and protocol designs are viable and practical to fully exploit network coding benefit and and they also take into account the application and protocol requirements thanks to the cross-layer design. We believe that the main ideas of our work can be extended to further optimize algorithms and protocols and eventually bridge the gap between the theory of network coding and its practical applications. This understanding and cross-optimization is essential for making the case for deploying network coding in practical networked systems.

Bibliography

[1] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.

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

[3] R. Koetter and M. Medard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 11(5):782–795, 2003.

[4] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong. A random linear network coding approach to multicast. *IEEE Transactions on Information Theory*, 52(10):4413–4430, 2006.

[5] R. W. Yeung, S. Y. Li, and N. Cai. *Network Coding Theory*. now Publishers Inc., 2005.

[6] C. Fragouli and E. Soljanin. *Network Coding Fundamentals*. now Publishers Inc., 2007.

[7] C. Fragouli and E. Soljanin. *Network Coding Applications*. now Publishers Inc., 2007.

[8] T. Ho and D. S. Lun. *Network Coding: An Introduction*. Cambridge University Press, 2008.

[9] Y. Wu, P. A. Chou, and S. Y. Kung. Information exchange in wireless network coding and physical layer broadcast. In *Proceedings of IEEE CISS*, Baltimore, MD, March 2005.

[10] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. Xors in the air: Practical network coding. In *Proceedings of ACM SIGCOMM*, Pisa, Italy, September 2006.

[11] S. Katti, D. Katabi, W. Hu, H. Rahul, and M. Medard. The importance of being opportunistic: Practical network coding for wireless environments. In *Proceedings of Allerton Conference*, University of Illinois at Urbana-Champaign, September 2005.

[12] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *Proceedings of Infocom*, Miami, FL, March 2005.

[13] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. In *Proceedings of Infocom*, Anchorage, AK, May 2007.

[14] J. K. Syndararajan, D. Shah, M. Medard, M. Mitzenmacher, and D.Barros. Tcp meets network coding. In *Proceedings of Infocom*, Rio de Janeiro, Brazil, April 2009.

[15] N. Thomos and P. Frossard. Network coding: from theory to media streaming. *Journal of Communications, Special issue on Multimedia Communications, Networking, and Application*, 4(9):628–639, 2009.

[16] H. Seferoglu and A. Markopoulou. Opportunistic network coding for video streaming over wireless. In *Proceedings of Packet Video*, Lausanne, Switzerland, November 2007.

[17] H. Seferoglu and A. Markopoulou. Video-aware opportunistic network coding over wireless networks. *IEEE JSAC*, 27(5):713–728, 2009.

[18] H. Seferoglu, A. Markopoulou, and U. C. Kozat. Network coding-aware rate control and scheduling in wireless networks. In *Proceedings of ICME*, New York, NY, June 2009.

[19] H. Seferoglu and A. Markopoulou. Distributed rate control for video streaming over wireless networks with intersession network coding. In *Proceedings of Packet Video*, Seattle, WA, May 2009.

[20] H. Seferoglu and A. Markopoulou. Network coding-aware queue management for unicast flows over coded wireless networks. In *Proceedings of NetCod*, Toronto, Canada, June 2010.

[21] H. Seferoglu and A. Markopoulou. Network coding-aware queue management for tcp flows over coded wireless networks. *arXiv:cs.NI:1002.4885*, February 2010.

[22] H. Seferoglu, A. Markopoulou, and K. K. Ramakrishnan. I^2nc: Intra- and inter-session network coding for unicast flows in wireless networks. In *Proceedings of Infocom*, Shanghai, China, April 2011.

[23] H. Seferoglu, A. Markopoulou, and K. K. Ramakrishnan. I^2nc: Intra- and inter-session network coding for unicast flows in wireless networks. *arXiv:cs.NI:1008.5217*, August 2010.

[24] A. Reibman and M. T. Sun. *Compressed Video over Networks.* New York: Marcel Dekker, 1 edition, 2001.

[25] M. Wang and B. Li. R2: Random push with random network coding in live peer-to-peer streaming. *IEEE JSAC*, 25(9):1655–1666, 2007.

[26] P. A. Chou, Y. Wu, and K. Jain. Practical network coding. In *Proceedings of Allerton Conference*, University of Illinois at Urbana-Champaign, September 2003.

[27] P. A. Chou and Y. Wu. Network coding for the internet and wireless networks. *IEEE Signal Processing Magazine*, 24(5):77–85, 2007.

[28] C. Fragouli, J. Widmer, and J. Y. LeBoudec. Network coding: an instant primer. In *Proceedings of ACM SIGCOMM CCR*, Pisa, Italy, September 2006.

[29] Z. Li and B. Li. Network coding: The case for multiple unicast sessions. In *Proceedings of Allerton Conference*, University of Illinois at Urbana-Champaign, September 2004.

[30] T. Ho and R. Koetter. Online incremental network coding for multiple unicasts. In *DIMACS WG on Network Coding*, Piscataway, NJ, January 2005.

[31] S. Deb, M. Effros, T. Ho, D. R. Karger, R. Koetter, D. S. Lun, M. Medard, and N. Ratnakar. Network coding for wireless applications: A brief tutorial. In *Proceedings of IWWAN '05*, London, UK, May 2005.

[32] A. Ramamoorthy, J. Shi, and R. Wesel. On the capacity of network coding for wireless networks. In *Proceedings of Allerton Conference*, University of Illinois at Urbana-Champaign, September 2003.

[33] S. Omiwade, R. Zheng, , and C. Hua. Butterflies in the mesh: lightweight localized wireless network coding. In *Proceedings of NetCod*, Hong Kong, January 2008.

[34] A. Khreishah, C. C. Wang, and N. B. Shroff. Cross-layer optimization for wireless multihop networks with pairwise inter-session network coding. *IEEE JSAC*, 27(5):606–621, 2009.

[35] M. Effros, T. Ho, and S. Kim. A tiling approach to network code design for wireless networks. In *Proceedings of ITW*, Punta del Este, Uruguay, March 2006.

[36] Q. Dong, J. Wu, W. Hu, and J. Crowcroft. Practical network coding in wireless networks. In *Proceedings of MobiCom*, Montreal, Canada, September 2007.

[37] P. Chaporkar and A. Proutiere. Adaptive network coding and scheduling for maximizing througput in wireless networks. In *Proceedings of ACM Mobicom*, Montreal, Canada, September 2007.

[38] J. Le, J. Lui, and D. M. Chiu. How many packets can we encode? - an analysis of practical wireless network coding. In *Proceedings of Infocom*, Phoenix, AZ, April 2008.

[39] S. Sengupta, S. Rayanchu, and S. Banarjee. An analysis of wireless network coding for unicast sessions: The case for coding-aware routing. In *Proceedings of Infocom*, Anchorage, AK, May 2007.

[40] F. Zhao and M. Medard. On analyzing and improving cope performance. In *Proceedings of ITA*, San Diego, CA, February 2010.

[41] Y. Huang, M. Ghaderi, D. Towsley, and W. Gong. Tcp performance in coded wireless mesh networks. In *Proceedings of SECON*, San Francisco, CA, June 2008.

[42] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein. Growth codes: Maximizing sensor network data persistence. In *Proceedings of ACM SIGCOMM*, Pisa, Italy, September 2006.

[43] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading structure for randomness in wireless opportunistic routing. In *Proceedings of ACM SIGCOMM*, Kyoto, Japan, August 2007.

[44] J. K. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher, and J. Barros. Network coding meets tcp. In *Proceedings of Infocom*, Rio de Janeiro, Brazil, April 2009.

[45] B. Girod, J. Chakareski, M. Kalman, Y. J. Liang, E. Setton, and R. Zhang. Advances in network-adaptive video streaming. In *Proceedings of IWDC 2002*, Capri, Italy, September 2002.

[46] S. Shankar, Z. Hu, and M. Van der Schaar. Cross layer optimized transmission of wavelet video over ieee 802.11a/e wlans. In *Proceedings of Packet Video*, Irvine, CA, December 2004.

[47] M. Van der Schaar and S. Shankar. Cross-layer wireless multimedia transmission: Challenges, principles, and new paradigms. *IEEE Wireless Communication Magazine*, 12(4):50–58, 2005.

[48] E. Setton, T. Yoo, X. Zhu, A. Goldsmith, and B. Girod. Cross-layer design of ad hoc networks for real-time video streaming. *IEEE Wireless Communication Magazine*, 12(4):59–65, 2005.

[49] G. M. Su, Z. Han, M. Wu, and R. Liu. Multi-user cross-layer resource allocation for video transmission over wireless networks. *IEEE Network Magazine*, 20(2):21–27, 2006.

[50] P. Chou and Z. Miao. Rate-distortion optimized streaming of packetized media. *IEEE Transactions on Multimedia*, 8(2):390–404, 2006.

[51] J. Chakareski and P. Frossard. Rate-distortion optimized distributed packet scheduling of multiple video streams over shared communication resources. *IEEE Transactions on Multimedia*, 8(2):207–218, 2006.

[52] J. Chakareski and P. Chou. Radio edge: Rate-distortion optimized proxy-driven streaming from the network edge. *IEEE/ACM Transactions on Networking*, 14(6):1302–1312, 2006.

[53] M. Kalman and B. Girod. Rate-distortion optimized video streaming with multiple deadlines for low latency applications. In *Proceedings of Packet Video*, Irvine, CA, December 2004.

[54] H. Seferoglu, O. Gurbuz, O. Ercetin, and Y. Altunbasak. Rate-distortion based real-time wireless video streaming. *Signal Processing: Image Communication*, 22(6):529–542, 2007.

[55] H. Seferoglu, A. Markopoulou, U. C. Kozat, M. R. Civanlar, and J. Kempf. Dynamic fec algorithms for tfrc flows. *IEEE Transactions on Multimedia*, 12(8):1–17, 2010.

[56] A. Markopoulou and H. Seferoglu. Network coding meets multimedia: Opportunities and challenges. *IEEE MMTC E-Letter*, 4(1):12–15, 2009.

[57] D. Nguyen, T. Nguyen, and X. Yang. Multimedia wireless transmission with network coding. In *Proceedings of Packet Video*, Lausanne, Switzerland, November 2007.

[58] F. Kelly, A. Maulloo, and D. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of Operational Research Society*, 49(3):237–252, 1998.

[59] R. Srikant. *The Mathematics of Internet Congestion Control (Systems and Control: Foundations and Applications)*. SpringerVerlag, 2004.

[60] M. Chiang, S. T. Low, A. R. Calderbank, and J. C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, 2007.

[61] X. Lin, N. B. Schroff, and R. Srikant. A tutorial on cross-layer optimization in wireless networks. *IEEE JSAC*, 24(8):1452–1463, 2006.

[62] X. Lin and N. B. Shroff. Joint rate control and scheduling in multihop wireless networks. In *Proceedings of Decision and Control*, Paradise Island, The Bahamas, December 2004.

[63] A. L. Stolyar. Greedy primal dual algorithm for dynamic resource allocation in complex networks. *Queuing Systems*, 54(3):203–220, 2006.

[64] U. Akyol, M. Andrews, P. Gupta, J. Hobby, I. Saniee, and A. Stolyar. Joint scheduling and congestion control in mobile ad-hoc networks. In *Proceedings of Infocom*, Phoenix, AZ, April 2008.

[65] D. S. Lun, N. Ratnakar, M. Medard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao. Minimum-cost multicast over coded packet networks. *IEEE Transactions on Information Theory*, 52(6):2608–2623, 2006.

[66] L. Chen, T. Ho, S. Low, M. Chiang, and J. C. Doyle. Optimization based rate control for multicast with network coding. In *Proceedings of Infocom*, Anchorage, AK, May 2007.

[67] B. Radunovic, C. Gkantsidis, P. Key, P. Rodriguez, and W. Hu. An optimal framework for practical multipath routing in wireless mesh networks. In *Proceedings of Infocom*, Phoenix, AZ, April 2008.

[68] J. Yuan, Z. Li, W. Yu, and B. Li. A cross-layer optimization framework for multi-hop multicast in wireless mesh networks. *IEEE JSAC*, 24(11):2092–2103, 2006.

[69] Z. Li, B. Li, and M. Wang. Optimization models for streaming in multihop wireless networks. In *Proceedings of ICCCN*, Honolulu, HI, August 2007.

[70] T. Cui, L. Chen, and T. Ho. Energy efficient opportunistic network coding for wireless networks. In *Proceedings of Infocom*, Phoenix, AZ, April 2008.

[71] D. Traskov, N. Ratnakar, D. S. Lun, R. Koetter, and M. Medard. Network coding for multiple unicasts: An approach based on linear optimization. In *Proceedings of ISIT*, Seattle, WA, July 2006.

[72] J. Y. Lee, W. J. Kim, J. Y. Baek, and Y. J. Suh. A wireless network coding scheme with forward error correction code in wireless mesh networks. In *Proceedings of Globecom*, Honolulu, HI, December 2009.

[73] Y. Liang, J. Apostolopoulos, and B. Girod. Analysis of packet loss for compressed video: Does burst-length matter? In *Proceedings of ICASSP*, Hong Kong, April 2003.

[74] M. Wang and M. van der Schaar. Rate distortion modeling for wavelet video coders. In *Proceedings of ICASSP*, Philadelphia, PA, March 2005.

[75] A. Eichhorn. Modeling dependency in multimedia streams. In *Proceedings of ACM Multimedia*, Santa Barbara, CA, October 2006.

[76] H. Shiang and M. van der Schaar. Multi-user video streaming over multi-hop wireless networks: A distributed, cross-layer approach based on priority queuing. *IEEE JSAC*, 25(4):770–785, 2007.

[77] Glomosim. *Global Mobile Information Systems Simulation Library - GloMoSim 2.0.*

[78] H.264. Advanced video coding for generic audiovisual services, 2005.

[79] H.264. *H.264/AVC Reference Software Version JM 8.6*, 2006.

[80] V. V. Vazirani. *Approximation Algorithms.* Springer-Verlag Berlin Heidelberg, 1 edition, 2003.

[81] P. Gupta and P. R. Kumar. The capacity of wireless network. *IEEE Transactions on Information Theory*, 34(5):910–917, 2000.

[82] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods.* Prentice Hall, 1989.

[83] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice Hall, 1982.

[84] E. Leonardi, F. Neri, and M. A. Marsan. On the stability of input-queued switches with speed-up. *IEEE/ACM Transactions on Networking*, 9(1):104–118, 2001.

[85] J. P. Wagner and P. Frossard. Playback delay optimization in scalable video streaming. In *Proceedings of ICME*, Amsterdam, The Netherlands, July 2005.

[86] Y. Li, A. Markopoulou, J. Apostolopoulos, and N. Bambos. Content aware playout and packet scheduling for video streaming over wireless links. *IEEE Transactions on Multimedia*, 10(5):885–895, 2008.

[87] X. Zhu and B. Girod. Distributed rate allocation for video streaming over wireless networks with heterogeneous link speeds. In *Proceedings of IWCMC*, Honolulu, HI, August 2007.

[88] T. Ozcelebi, A. M. Tekalp, and M. R. Civanlar. Delay-distortion optimization for content-adaptive video streaming. *IEEE Transactions on Multimedia*, 9(4):826–836, 2007.

[89] K. Stuhlmuller, N. Farber, M. Link, and B. Girod. Analysis of video transmission over lossy channels. *IEEE JSAC*, 18(6):1012–1032, 2000.

[90] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *Proceedings of ACM SIGCOMM*, Portland, OR, September 2004.

[91] C. Steger, P. Radosavljevic, and J. P. Frantz. Performance of ieee 802.11b wireless lan in an emulated mobile channel. In *Proceedings of VTC*, Orlando, FL, October 2003.

[92] V. Sharma, S. Kalyanaraman, K. Kar, K. K. Ramakrishnan, and V. Subramanian. Mplot: A transport protocol exploiting multipath diversity using erasure codes. In *Proceedings of Infocom*, Phoenix, AZ, April 2008.

[93] S. Rayanchu, S. Sen, J. Wu, S. Banerjee, and S. Sengupta. Loss-aware network coding for unicast wireless sessions: Design, implementation, and performance evaluation. In *Proceedings of ACM Sigmetrics*, Annapolis, MD, June 2008.

[94] R. Gowaikar, A. F. Dana, B. Hassibi, and M. Effros. A practical scheme for wireless network operation. *IEEE Transactions on Communications*, 55(3):463–476, 2007.

[95] V. Sharma, K. K. Ramakrishnan, K. Kar, and S. Kalyanaraman. Complementing tcp congestion control with forward error correction. In *Proceedings of Networking*, Aachen, Germany, May 2009.

[96] O. Tickoo, V. Subramanian, S. Kalyanaraman, and K. K. Ramakrishnan. Lt-tcp: End-to-end framework to improve tcp performance over networks with lossy channels. In *Proceedings of IWQoS*, Passau, Germany, June 2005.