PRIORITY-BASED DATA TRANSMISSION IN WIRELESS NETWORKS USING NETWORK CODING

A Dissertation
Submitted to
the Temple University Graduate Board

in Partial Fulfillment
of the Requirements for the Degree of
DOCTOR OF PHILOSOPHY

by

Pouya Ostovari
December, 2015

Examining Committee Members:

Dr. Jie Wu, Dept. of Computer and Information Sciences (Advisor)
Dr. Bo Ji, Dept. of Computer and Information Sciences (Committee chair)
Dr. Avinash Srinivasan, Dept. of Computer and Information Sciences
Dr. Li Bai, Dept. of Electrical and Computer Engineering (External reader)

ABSTRACT

PRIORITY-BASED DATA TRANSMISSION IN WIRELESS NETWORKS USING NETWORK CODING

By

Pouya Ostovari

Temple University, 2015

With the rapid development of mobile devices technology, they are becoming very popular and a part of our everyday lives. These devices, which are equipped with wireless radios, such as cellular and WiFi radios, affect almost every aspect of our lives. People use smartphone and tablets to access the Internet, watch videos, chat with their friends, and etc. The wireless connections that these devices provide is more convenient than the wired connections. However, there are two main challenges in wireless networks: error-prone wireless links and network resources limitation.

Network coding is widely used to provide reliable data transmission and to use the network resources efficiently. Network coding is a technique in which the original packets are mixed together using algebraic operations. In this dissertation, we study the applications of network coding in making the wireless transmissions robust against transmission errors and in efficient resource management. In many types of data, the importance of different parts of the data are different. For instance, in the case of numeric data, the importance of the data decreases from the most significant to the least significant bit. Also, in multi-layer videos, the importance of the packets in different layers of the videos are not the same. We propose novel data transmission methods in wireless networks that considers the unequal importance of the different parts of the data. In order to provide robust data transmissions and use the limited resources efficiently, we use random linear network coding technique, which is a type of network coding.

In the first part of this dissertation, we study the application of network coding in resource management. In order to use the the limited storage of cache nodes efficiently, we propose to use triangular network coding for content distribution. We also design a scalable video-on-demand system, which uses helper nodes and network coding to provide users with their desired video quality. In the second part, we investigate the application of network coding in providing robust wireless transmissions. We propose symbol-level network coding, in which each packet is partitioned to symbols with different importance. We also propose a method that uses network coding to make multi-layer videos robust against transmission errors.

Keywords: Network coding, unequal data protection, wireless networks, performance evaluation, triangular network coding.

To my mother, brothers, and the soul of my father

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Dr. Jie Wu, the chair of the computer and information science department and my advisor. He guided me in my research, provided me an excellent environment at the Center for Networked Computing, and taught me how to think about the research problems. I learned many things from him, and I am confident that they will be valuable assets for my future career and life. I am very grateful for being his student.

My special thank goes to Dr. Abdallah Khreishah, who helped me a lot in my research. Working with him was a great opportunity for me to learn network coding and optimization techniques. He always had time for me to discuss the research challenges. I will never forget his support.

I would like to thank Dr. Bo Ji, Dr. Avinash Srinivasan, and Dr. Li Bai for being my committee members. I sincerely appreciate their valuable comments and help to complete my dissertation.

I would like to thank Professor Christopher Pascucci. It was my pleasure to be his teaching assistant for about four years. I deeply appreciate his help and support.

I thank my labmates at the Center for Networked Computing for the good moments that we had together and their help and support. To all of the staff of the computer and information science department, thanks for their help.

Finally, I wish to thank my family, Fereshteh Rajabi, Arash Ostovari, Babak Ostovari, Armita Ghasemieh, and Banafsheh Hosseini. Thanks for encouraging and supporting me. I could not finish my research without their support.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

CHAPTER 1

INTRODUCTION

Using wireless radios is a convenient way to connect mobile devices together. However, unreliability of the links and limited bandwidth (resource limitations in general) are two major challenges in wireless networks. Network coding, in which data packets are mixed together using algebraic operation, is a promising technique in making the transmissions robust against transmission errors and in using the resources efficiently.

Network coding has many applications in wired and wireless networks, such as in reliable data transmission, enhancing network throughput, content distribution, and security. The focus of this dissertation is on the applications of network coding in enhancing the robustness of wireless data transmissions and in using the limited network resources, such as bandwidth and storage, efficiently. Specifically, we focus on the applications of network coding in unequal data protection and in multi-layer video transmission.

1.1 Background

In recent days, we are witnessing a fast development in the hardware technology of mobile sireless devices, such as tablets and smartphones. As a result of these advances, the mobile wireless devices are becoming very popular. Using wireless connections is more convenient than the wired connections. Broadcasting schemes are widely used for disseminating data and control messages in wireless networks. Also, mobile wireless devices are widely used to watch videos that require a high

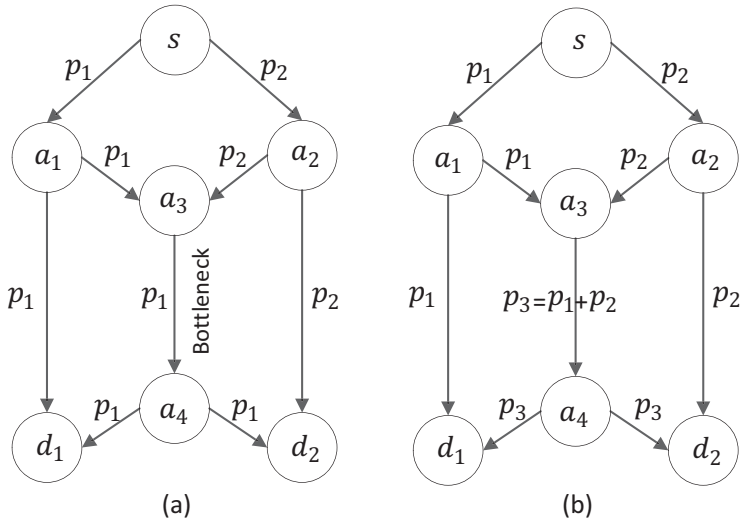


Figure 1.1: Network coding in butterfly network.

bandwidth. However, error-prone wireless links and resource limitation are two major challenges in wireless networks.

Network coding is an effective technique that can be used as a tool to provide reliable transmissions and use the limited network resources such as bandwidth and storage efficiently. In random linear network coding [1], which is a special type of network coding, the original packets (data) are mixed linearly together using random coefficients. Assuming that k packets are coded together, we can potentially generate an infinite number of coded packets. Any k linearly independent coded packets out of the generated coded packets are sufficient to decode the coded packets and retrieve the original packets. Using random linear network coding, a transmitter can keep transmitting coded packets until its destination nodes receive k linearly independent coded packets.

Network coding has received a lot of attention from the research community. The first time network coding was proposed for wired networks to solve the bottleneck problem and achieve the capacity of multicast problem. Consider the example in Figure 1.1(a). The source node s has 2 packets p_1 and p_2 to transmit to nodes d_1 and d_2. Assume that the capacity of each link equals one packet. A possible transmission

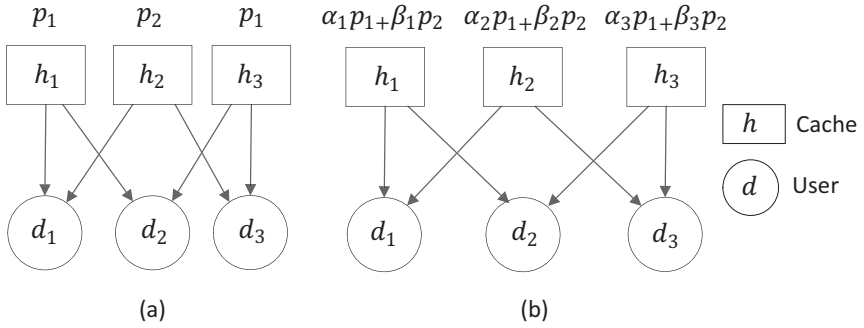


Figure 1.2: Application of network coding in resource management.

scenario is shown in Figure 1.1(a). In this case, node d_1 does not receive packet p_2, which is due to the bottleneck between nodes a_3 and a_4. In Figure 1.1(b), network coding has been applied on the received packets by node a_3, and a linear combination of the packets is transmitted. In this case, each of the destination nodes receive two linear independent coded packets. As a result, they are able to use Gaussian elimination and retrieve packets p_1 and p_2.

To see another application of network coding consider the example in Figure 1.2. We have three caches h_1, h_2, and h_3, which are connected to three users as shown in the figure. Let us assume that the capacity of each storage is one packet, and we have two packets to store on the caches. Figure 1.2(a) shows a possible packet distribution on the caches. In this case, user d_2 does not have access to packet p_2. In Figure 1.2(b), network coding has been applied on the original packets. In this case, each user have access to two coded packets. As a result, all of the users are able to decode the coded packets and retrieve the original packets.

1.2 Challenges

Motivated by the examples on the applications of network coding, in this dissertation, we study two major challenges in wireless networks that can be addressed using network coding.

The first challenge that we address in this dissertation is the limited network resources, e.g. storage capacity and bandwidth. In order to use the limited network resources optimally, we should consider that different data might have different importance. In many types of data, the importance of different parts of the data are not equal. For example, in the case of numeric data, e.g., the captured information by sensor nodes, the importance of the data (numbers) decreases from left (most significant bit) to right (least significant bit). Another type of data with unequal importance is multi-layer video [2,3], in which a video is divided to video layers with different qualities. The base layer is required to watch the video and the other layers can enhance the quality of the received video. In Chapters 3 and 4, we study distribution of multi-layer videos over caches with limited storage. We show the benefit of network coding in simplifying content distribution and using the limited resources efficiently.

The second challenge is unreliability of wireless links. The easiest approach to provide reliable transmissions is using feedback messages. Howsoever, the overhead of feedback messages make them infeasible for multicast and broadcast applications. Another common approach to protect the transmissions against errors is to add redundancy through different types of codes. In this way, a subset of coded packets is sufficient to retrieve the original data or partial original data. In order to enhance the robustness of data transmissions, it is important to consider the unequal importance of these types of data. In Chapters 5, 6, and 7, we study the application of network coding in protecting multi-layer videos and the general type of data against transmission errors.

1.3 Major Contributions

From the above discussion, the contributions of this dissertation fall in the field of priority-based data transmission, which can be summarized as follows:

- We investigate the benefit of network coding in multi-layer video transmission. We use a special type of network coding, called triangular network coding, for content distribution on cache nodes. We show that using triangular network coding, the limited storage of caches can be used efficiently.

- We show the benefit of joint inter- and intra-layer network coding in designing a scalable video-on-demand system. In order to make the system scalable, we use cache nodes that can help to reduce the load on the servers. In order to use the limited storage and bandwidth of the helpers efficiently, we propose to use joint inter- and intra-layer networks coding. We also propose a distributed mechanism to find the optimal distribution of the videos on the cache nodes.

- The problem of robust multi-layer video broadcast using network coding is investigated. For the first time, we propose a 2-dimensional triangular network coding. We illustrate the effectiveness of our proposed methods by applying our coding scheme on real video traces.

- We study the problem of symbol-level data transmission, in which different parts of the packets to be transmitted have different importance. We propose efficient unequal-error protection methods to make the data transmission robust against transmission errors. In order to enhance the data transmission, we benefit from symbol-level network coding in our proposed method.

- In order to evaluate our symbol-level network coding method, we implemented it on universal software radio peripheral (USRP) devices. In order to make our symbol-level network coding method robust against bursty transmission errors, we propose modified versions of our basic algorithm.

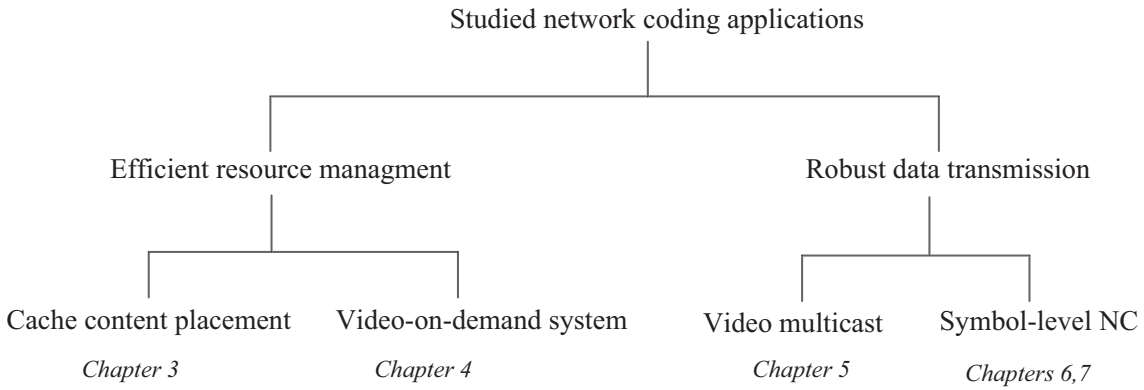


Figure 1.3: Dissertation overview.

1.4 Dissertation Overview

The rest of this dissertation is organized as follows. We review the relevant research work in Chapter 2. Chapters 3, 4, and 5 are on the applications of network coding in multi-layer video transmission. We study the benefit of a special type of network coding in video caching in Chapter 3. In Chapter 4, we use helper nodes and network coding to design a scalable video-on-demand system. We propose a promising network coding-based robust video transmission scheme in Chapter 5. In Chapters 6 and 7, we study the problem of symbol-level network coding. Chapter 8 concludes this dissertation and identifies the potential future work. Figure 1.3 depicts the overview of the network coding applications in dissertation and their corresponding chapters.

CHAPTER 2

RELATED WORK AND BACKGROUND

In this chapter, we provide a background on network coding. We also review the related work on reliable transmissions, video-on-demand systems, and unequal error protection.

2.1 Reliable Transmission

Certain mechanisms, such as feedback messages, can be applied in error-prone wireless networks to provide reliability. Automatic Repeat reQuest (ARQ) is one of the most frequently used approaches for addressing this challenge [4]. Nevertheless, ARQ imposes overhead, since it requires transmitting many feedback messages, especially for the case of multi destination nodes. Hybrid-ARQ approaches [5, 6], which combine FEC (Forward Error Correction) with ARQ, are proposed to solve this problem. The RMDP approach, which is a complex method, [6] uses Vandermonde [7] code and ARQ to ensure reliability.

Using rateless (fountain) codes [8–10] is an efficient way to provide reliability without using feedback messages. In these schemes, the source node can generate and transmit an unlimited number of encoded packets until each destination node receives enough encoded packets to retrieve the original packets. In this scheme, the destination nodes need to collect a sufficient number of encoded packets, regardless of which packets have been lost. Assuming that the number of original packets is k, the number of sufficient coded packets that need to be received is $(1 + \Omega)k$ [8], where Ω is a small number and shows the overhead of the rateless codes. Note that

Ω is independent of the reliability of the links. It can be shown that as $k \to \infty$, the overhead goes to zero [11]. Therefore, rateless codes are very efficient for transmitting a large number of packets, but are inefficient for transmitting a small number of packets. As a result, rateless codes are not appropriate for delay-sensitive applications which needs small batches of packets.

In order to combat with the bit errors during wireless transmission, forward error correction (FEC) codes [12–14] are widely used at the physical layer. A FEC code converts a packet of n bits to a packet of $n + k_b$ bits. FEC codes can detect and correct at most k_b bit errors. The rate of a FEC code is defined as the ratio of the original packet size to the packet size after adding the FEC bits. The works in [15,16] studied the problem of designing efficient FEC codes.

The authors in [17, 18] study the problem of optimal redundancy allocation between packet-level erasure codes and physical-layer channel coding in the case of single destination node. In order to evaluate the synergy between FEC and erasure codes, the authors in [19] perform an experiment in a low power wireless sensor. They use random linear network coding as an erasure code, and combine it with physical-layer FEC. The authors' motivation was the hostile wireless environment which low power sensors operate in, in which there is significant interference from nearby nodes. The results of the paper show the effectiveness of random linear network coding, outperforming physical-layer FEC, in the case of high packet erasure rate. However, the authors do not propose any method to find the optimal transmission strategy.

2.2 Network Coding

Network coding (NC) [20–26] is introduced in [27] for wired networks, to solve the bottleneck problem in single multicast problem. It is shown in [28] that linear network coding achieves the capacity for the single multicast session problem. The authors in [29] provide a useful algebraic representation of the linear network coding problem.

Random linear network coding is proposed in [1], and it is shown that randomly selecting the coefficients of the coded packets, achieves the capacity asymptotically, with respect to the finite field size.

In random linear network coding, coded packets are the random linear combination of the original packets over a finite field. The coded packets are in the form of $\sum_{i=1}^{k} \alpha_i \times p_i$, where p and α are the packets and random coefficients, respectively. Using random linear network coding, the source node generates and transmits random coded packets and their respective random coefficient vector. The destination nodes are able to decode the coded packets once they receive k linearly independent coded packets. The decoding process is done using Gaussian elimination for solving a system of linear equations. Using this scheme, the destination nodes can send just one acknowledgment message to stop the source node from sending more coded packets once they are able to decode the coded packets.

The work in [30–33] address the problem of reliable one-hop broadcasting. In order to provide reliability, the source node needs to retransmit the lost packets by the destination nodes. The source node uses the benefit of network coding in the retransmission phase to improve the transmission efficiency. In order to reduce the number of required retransmissions, these methods combine the packets that have not been received correctly by different receiver nodes. Assume that source node s sends packets p_1 and p_2, and destination nodes d_1 and d_2 only receive packets p_1 and p_2, respectively. As a result, the source node needs to retransmit both of the packets. However, using network coding, the source node can mix the packets to send a single packet $p_1 + p_2$. If nodes d_1 and d_2 receive the coded packets, they can retrieve their respective packets p_2 and p_1, by performing $(p_1 + p_2) - p_1$ and $(p_1 + p_2) - p_2$, respectively.

Symbol-level network coding for wireless mesh networks is introduced in [34], and it is shown that its throughput is more than that of the packet-level network

coding. The insight behind the symbol-level coding is that, even in the case that a node does not receive a coded packet correctly, some of the symbols that form the packet might be received without any error. As a result, if instead of coding the packets together we code the symbols, the successfully received symbols do not need to be retransmitted, and transmitting the remaining symbols is sufficient; therefore, symbol-level transmission reduces the transmission cost.

The authors in [35, 36] use the symbol-level coding to propose a method for distributing data and multimedia in vehicular networks. They show that the symbol-level network coding outperforms the packet level network coding for content distribution in Vehicular Ad-Hoc Networks (VANET). The goal in [36] is to efficiently designate live streaming multimedia to the mobile nodes in a specific region of a road, called an area of interest.

Network coding can be classified into local and global coding. In local network coding, each relay node decodes the received coded packets, and it mixes the native (non-coded) packets, such that its neighbors can decode the coded packet using the packets in their buffer. This means that the next hops can decode the received coded packets immediately, and they do not need to wait to receive further packets to be able to decode the coded packets. On the other hand, in global network coding, the intermediate nodes do not perform decoding; they just code the coded packets again without considering the status of their neighbors. In this approach, when a receiver node receives a coded packet, it cannot decode the packet immediately, and it has to wait to receive a sufficient number of packets to be able to decode the coded packet. Usually, local network coding protocols use XOR coding, and global protocols perform random linear coding.

From another perspective, network coding can be classified into intra- and inter-session network coding. Intra-session network coding uses the diversity of the wireless links, and codes packets from the same sessions, to address the packet loss problem

and to provide reliability. In contrast, inter-session network coding mixes the packets from different sessions (sources) to solve the bottleneck problem and reduce the number of transmissions.

The work in [37] uses intra-session network coding for the dissemination of data from a source node to the sensor nodes in wireless sensor networks. The R-Code method [38], constructs a minimum spanning tree, based on the reliability of the links, in which each non-leaf node works as a relay node. In R-Code, each parent node is responsible for delivering a sufficient number of intra-session linear coded packets to its children nodes, until they are all capable of decoding the coded packets..

The COPE method is a practical forwarding architecture for multiple unicast sessions [20,39]. The nodes in COPE opportunistically listen to the transmissions and the relay nodes that are intersection of unicast flows use the advantage of inter-session network coding to reduce the number of transmissions. The authors in [40] show that, for fixed networks, inter-session network coding can, at most, offer a constant factor of benefits in terms of energy efficiency. They also propose a probabilistic network coding-based broadcasting algorithm. The work in [41] combines the partial dominant pruning (PDP) forwarding approach [42], which is a deterministic approach, with network coding. The algorithm uses local, two-hop topology information, and makes use of opportunistic listening to reduce the number of transmissions. Using network coding with directional antennas is considered in [43]. The work is based on the deterministic forwarding approach that uses directional CDS. It can be noted that the works that use the deterministic approach with network coding limit coding to XOR operations, and exploit only local coding opportunities.

2.3 Video-on-Demand System with Network Coding

The authors in [44] study rate allocation problem in Peer-to-peer (P2P) VoD streaming. They propose a distributed rate allocation algorithm, which can reduce

the unfriendly traffic to the Internet service providers (ISP), such as inter-ISP traffic, without much increase on the server load. In [45], distributed bandwidth allocation in live P2P streaming is studied. The challenges and the design issues of a large-scale P2P-VoD system are studied in [46]. The authors argue that less synchrony in the video contents shared by the users in VoD streaming makes the problem of reducing server load and maintaining streaming performance hard. In order to resolve this problem, each peer needs to contribute a small amount of storage. The paper proposes content replication, content discovery, and peer scheduling schemes.

In [47], the authors study live streaming of a single video in a helper-assisted P2P system. In their proposed method, each helper downloads one coded packet of the currently streamed segment. The simulation results show a significant increase in the streaming bitrate. The authors in [48] use helpers in a P2P VoD system to stream a single video, and propose a distributed bandwidth allocation algorithm for the helpers.

In [49], the role of coding in the design of a large-scale Video-on-Demand (VoD) system is studied. The authors show that NC can convert a combinatorial problem into a tractable problem. In [50], a P2P Video-on-Demand (VoD) system using helpers is proposed. The objective of the paper is to minimize the server load in the case of limited helpers' bandwidth and storage. The authors formulate the problem as an LP optimization, and propose a distributed algorithm to solve it. However, their distributed scheme oscillates among different solutions. As a result, it does not converge to the optimal solution. The other problem with the oscillation is that it can result in delay oscillation, which might cause playback lags.

2.4 Unequal Error Protection

Unequal error protection (UEP) has been widely employed on multi-resolution videos. The authors in [51] proposed a UEP scheme by exploiting the unequal impor-

tance of the temporal and quality layers. They use a genetic algorithm to distribute the redundancy to different layers. In [52], a performance metric is proposed to measure the importance of the quality and temporal layers, which are based on the error propagation of a packet loss in a given layer. The authors use the hill climbing method and their performance metric to efficiently assign redundancy to different layers.

The advantage of combining multi-resolution coding with network coding has been studied by [53, 54]. In [53], the authors propose a video multicast with joint network coding and video interleaving. They put the temporal layers of different GoPs that have the same importance to the same partition, and perform network coding among the layers of the same partition. The amount of FEC assigned to each partition depends on the importance of that partition. The authors in [54] study the cases of non-coding, random linear coding, and triangular coding, and show that the gain in the case of triangular network coding is more than that of the other cases. They also propose a triangular coding scheme for multi-hop networks.

The work in [55] combines ARQ with the triangular coding scheme and rate control. The authors assume that each user requests the desired number of layers from the source node. In [56], the authors apply multi-generation coding, which is similar to triangular coding. In their proposed method, they distribute the redundant transmissions between different coded layers evenly.

The authors in [57] show that the performance of the previously proposed triangular inter-layer coding schemes are poor, and they use the estimated number of decodable layers as a measure to find how many layers should be coded to enhance the coding performance. In order to find the optimal triangular coding strategy (transmissions distribution) in the case of multiple users, the authors create a reference table which contains the number of decodable layers for a given delivery rate and triangular coding strategy. They also propose a set of optimization techniques to reduce the time complexity of the search for the optimal solution. These optimizations

include controlling the number of packets that should be assigned to different layers in each round of the algorithm run (granularity), and checking the decodability of a set of packets without using Gaussian elimination. Using the reference table, the source node can search for the optimal solution given the delivery rates of the receivers.

CHAPTER 3

CACHE CONTENT PLACEMENT USING TRIANGULAR NETWORK CODING

In this chapter, we study the application of network coding in efficient resource allocation. In more details, we study the application of network coding in content distribution on a set of caches with limited storage.

3.1 Setting

We consider a single cell, equipped with a base station (such as 4G), which serves n users, $d_1, .., d_n$ with the help of m independent caches, $h_1, .., h_m$. The cache nodes are placed in fixed positions in the cell, and each cache node can cover a subset of the users based on their locations. There are e video layers, $l_1, .., l_e$ on a base station where $e < m$. The capacity of the caches is equal to the size of the video layers, and if a layer is not available on any adjacent caches to a user, the user will contact the central server to download the layer. However, the cost of downloading from the server is more than downloading from the cache nodes.

The key point is that, if there is enough content reuse, such that many users are requesting the same video layers, caching can replace the communication between the users and the base station. We assume that in order to use the layer l_i, all of the layers $l_j \ \forall j, 0 \leq j \leq i$ should be available to the user. We represent the availability of the video layers to the users with matrix $[z]_{ij}$. If the layers l_1 to l_i are available to user d_j, then $z_{ij} = 1$; otherwise, $z_{ij} = 0$. The set of symbols used in this chapter is summarized

Table 3.1: The set of symbols used in this chapter.

Notation	Definition
n	The number of users
m	The number of caches
e	The number of video layers
d_i	The i-th user
h_j	The j-th cache
l_k	The k-th layer
b_i	The degree of the ith user
ϕ_i	The number of filled out adjacent caches to the ith user
r_i	The rank of the coded video layers on the filled out adjacent caches to the ith user
s_i	The number of triangular codes of degree i that are available to a user
ϕ	The fairness
ϕ'	The unfairness
v	The average number of available video layers to the users
q_i	The number of available video layers to the ith user

in Table 3.1. Our goal is to minimize the number of contacts to the central server for layers retrieval. In other words, we want to maximize the total number of video layers that are available to the users. Therefore, our objective becomes:

$$\max \sum_{i=1}^{e} \sum_{j=1}^{n} z_{ij} \tag{3.1}$$

The solution is straightforward when the cache nodes are not dense and each user is connected to a single cache. In this case, each cache should store the first video layer (with the smallest index), since other layers are not useful without accessing to this layer. However, if the cache nodes deployment is dense enough, each user will have more than one cache node that is a part of a distributed caching system. In this case, the content placement problem becomes more complicated because of the correlation among the cache nodes. It is shown in [58] that the uncoded distributed caching problem is NP-complete.

3.2 Content Placement Approach Using Network Coding

In the problem of content placement using network coding, each video layer can be coded with any other layers. Therefore, there are $2^e - 1$ possibilities for coding the video layers, where e is the number of layers on the server (the empty set should not be considered). On the other hand, we have m caches in the network; in total, there are $(2^e - 1)^m$ different possible placements. To calculate the number of video layers that are available to the users in each placement, we need to calculate the layers that can be retrieved by each user. Note that we cannot use the Gaussian Elimination directly on the coded video layers that are available to a user nor compute the rank of the available layers. This is because by using Gaussian Elimination, or the overall rank of the coded video layers, we can just find out if all of the original layers are decodable or not. Therefore, we need to calculate the rank of each subset of the coded packets that are available to a user or use the Gaussian Elimination to decode the coded video layers. As a result, the total complexity becomes $(2^h-1)^m n \sum_{i=1}^{n}(2^{b_i}-1)$.

One approach to decrease the complexity of the problem is to use triangular network coding [59], which is a kind of random linear network coding. In triangular network coding, the encoded video layers are in the form of $\sum_{j=1}^{k} \alpha_j l_j$, where $1 \leq k \leq h$ and α_j are random coefficients. In other words, each coded layer is a random linear combination of the first k original layers. Therefore, there are just e possibilities for coding e original layers. Figures 3.1 (b) and (c) show the possible coded layers of the original layers in Figure 3.1 (a) using the general form of network coding and triangular coding, respectively. As shown in Figures 3.1 (b), there are $7 = 2^3 - 1$ choices for coding 3 original packets in the case of the general form of random linear network coding, but just 3 possibilities for the case of triangular network coding. The coefficients are not shown in the figures for simplicity. For example, $l_1 + l_2$ means $\alpha_1 l_1 + \alpha_2 l_2$, where α_1 and α_2 are two random coefficients.

We prefer using triangular network coding over the general form of network coding

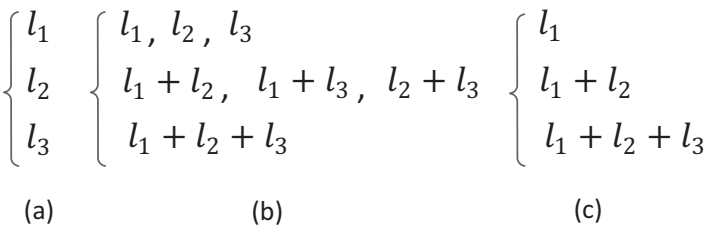


Figure 3.1: (a) Original video layers. (b) General form of random linear network coding. (c) Triangular network coding.

for three reasons. First, it limits the coding space of the coding problem. Second, based on our setting, the i-th video layer is not useful without accessing to the layers with a smaller index. This assumption has the same nature as the coding rule in triangular network coding. Third, in addition to the lesser complexity of triangular coding, compared with linear coding, we do not need to use Gaussian Elimination to check the number of decodable video layers at each user. This checking can be done by a simple algorithm in order of $O(e^2)$ [57]. Assume that s_k represents the number of coded packets in the form of $\sum_{j=1}^{k} \alpha_j l_j$ that are available to a user. In this scheme, the user can decode the first i layers if $\sum_{j=i-k}^{i} s_j \geq k+1, \forall k \in [0, i-1]$. The condition implies that the rank of the coefficients matrix of the available encoded video layers from coding any of the first i layers is at least i. Therefore, Gaussian Elimination will be able to decode the first i layers. For example, assume that three packets, $\alpha_1 l_1$, $\alpha_2 l_1 + \alpha_3 l_2$, and $\alpha_4 l_1 + \alpha_5 l_2 + \alpha_6 l_3$, are available to a user. In this example, $s_3 \geq 1$, $s_2 + s_3 \geq 2$, and $s_1 + s_2 + s_3 \geq 3$, so all three layers can be decoded by the user.

Lemma 3.1. *the complexity of the content placement using triangular network coding is* $e^{m+2}m$.

Proof. In this case, there are just e different coding possibilities since $1 \leq k \leq h$. Therefore, the total number of different placements will be e^m. As mentioned before, the number of decodable original video layers at each user can be calculated in order on $O(e^2)$. Therefore, the total complexity of checking all possible triangular coding

Algorithm 1 Checking the number of decodable packets in triangular coding

```
1: for i=1 to m do
2:    decodability=True
3:    for k=0 to i-1 do
4:       if $\sum_{j=i-k}^{i} s_j \leq k+1$ then
5:          decodability=False
6:       if decodability=True then
7:          number of decodable layers=i
```

placements becomes $e^{m+2}m$. □

Although the complexity of triangular network coding is much less than random linear coding, it might not achieve efficiency compared to linear coding. Consider Figures 3.2 (a) and (b), where the optimal placement using triangular and random linear coding are shown in Figures 3.2 (a) and (b), respectively. In these figures, coefficients are not shown for simplicity. For example, $l_1 + l_2$ means $\alpha_1 l_1 + \alpha_2 l_2$, where α_1 and α_2 are random coefficients. In the case of triangular network coding, user d_4 will not be able to decode any packet. However, when we use random linear coding, this user can decode video layers l_1 and l_3. Note that, based on our setting, the layer l_3 is not useful for user d_4 since the user does not have access to the layer l_2.

3.2.1 Network Coding-based Content Placement Heuristic

Although the complexity of finding the optimal placement using a triangular network coding is much less than random linear network coding, it is still high. For this reason, we propose a network coding-based content placement heuristic (NCCP) in this section. Based on the proposed setting, we know that the video layer l_i is not useful to a user unless the layers l_j $\forall j, 0 < j < i-1$ are available to the user. Based on this observation, we give more priority to the first layer, and we try to provide it to all of the users. Then, if it is possible, we provide the second layer to the users. In more detail, we try to provide the second video layer to as many users as possible. This process is repeated until we cannot provide the content l_i to any user for a given

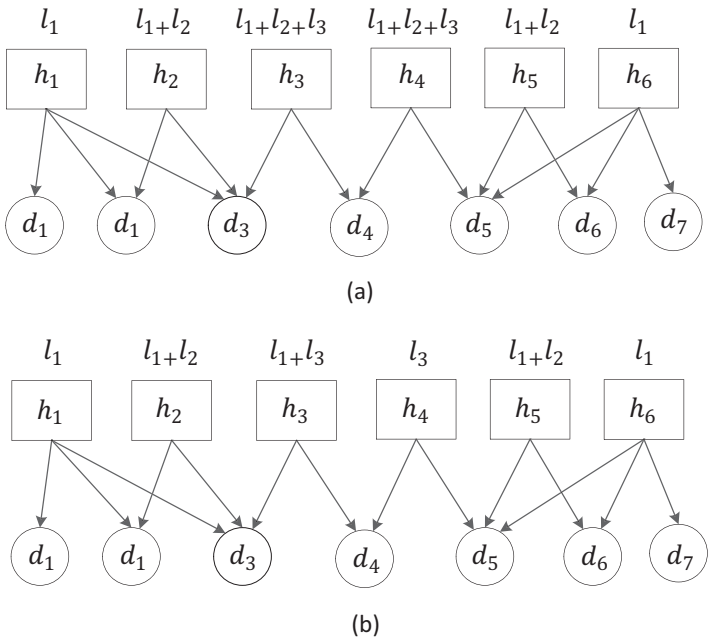


Figure 3.2: A comparison of triangular network coding and random linear coding. Coefficients are not shown for simplicity.

$1 < i \leq h$. This method can also provide a fair solution to the content placement problem. This is because the algorithm tries to provide each layer that has a greater priority (smaller index) to as many users as possible, and then the algorithm will start to provide the next layer, which is less popular to the users. We can observe that this solution has a greedy nature, and thus we can use a greedy algorithm to implement this solution.

In each step, our greedy algorithm selects one of the users and fills up its adjacent cache with random linearly coded video layers. Because of the greedy nature of our algorithm, the generated coded layer will always be a triangular code. The user selection rules are as follows:

- *Rule 1*: the user with the minimum degree is selected. The degree of user d_i, represented as b_i, is defined as the number of its adjacent caches.

- *Rule 2*: in the case where two users have the same degree, the user with a larger number of filled-up cache nodes is selected.

- *Rule 3*: if two users have the same degree and the same number of filled caches, the user whose adjacent caches have less cumulative ranks is selected.

We give more priority to Rule 1 compared to Rule 2 and Rule 3. Also, the priority of Rule 2 is more that that of the Rule 3. The reason for selecting user d_i with the minimum degree is that there are less content placement choices for adjacent cache nodes to that user. If we do not start with such a user d_i, it is possible for us to fill the adjacent cache nodes to user d_i with an undecodable packet. For example, in Figure 3.3, user d_1 has just one adjacent cache. Consequently, we can just provide video layer l_1 to this user. By selecting user d_1 and filing its adjacent cache h_1, we can reduce the number of choices for cache h_2, which is adjacent to user d_2. Then, user d_3 is selected, and its adjacent cache will be filled with l_1. In this example, if we start from user d_2 and fill its adjacent cache nodes with random linear combinations of l_1 and l_2, users d_1 and d_3 will not be able to decode the coded video layers on cache nodes h_1 and h_3, respectively. Note that network coding does not have any benefit in this example, and we use this simple example just to describe our user selection policy.

After selecting the appropriate user based on Rules 1, 2, and 3, the user's neighboring cache nodes should be filled up. Let b_i be the degree of user d_i. We represent the number of user d_i's filled-up adjacent cache nodes and their cumulative rank as v_i and r_i, respectively. Assume that, in the current iteration, user d_i has been selected based on the rules. The degree of this user is b_i, so this user cannot receive more than the first b_i video layers. On the other hand, v_i of the caches are already filled, and their communicative rank is r_i. As a result, the maximum number of decodable layers to user d_i is $b_i - v_i + r_i$. Therefore, the algorithm fills up all of the empty adjacent cache nodes to user d_i with a random linear combination of the $b_i - v_i + r_i$ first video layers. Then, the algorithm selects another unprocessed user by utilizing the user selection rules, and the algorithm uses the same policy to fill the cache nodes

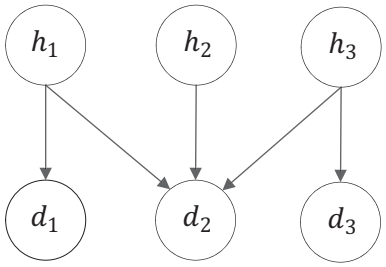


Figure 3.3: User selection rules.

Algorithm 2 Content placement

1: **while** There is an empty cache **do**
2: Select an unprocessed node d_i based on Rules 1, 2, and 3
3: $g = b_i - v_i + r_i$
4: **for** each empty adjacent cache h_k to user d_i **do**
5: Fill up cache h_k with $\sum_{j=1}^{g} \alpha_j l_j$
6: Tag node d_i as a processed node

that are adjacent to the new user. The proposed greedy algorithm is described in Algorithm 2.

Assume that user d_i is selected, and all of the b_i adjacent caches are empty. The algorithm fills up all of them with $\sum_{j=1}^{b_i} \alpha_j l_j$. In this case, the user will have access to b_i linearly independent coded video layers. Thus, the user will be able to decode all of the encoded layers. Now, assume that the degree of user d_j is 3, and two of the adjacent caches are filed with $\alpha_1 l_1$ and $\alpha_2 l_1$. In this case, the rank of the filled caches is 1. Therefore, it is not possible to provide 3 layers to the user, and if we fill up the remaining unfilled cache with a linear combination of l_1, l_2, and l_3, the user will not be able to decode the coded packets. As a result, based on our algorithm, we code the first $g = 3 - 2 + 1$ layers together, and assign $\alpha_3 l_1 + \alpha_4 l_2$ to the remaining cache.

Consider Figure 3.4, in which user d_1 has a minimum degree of 2. As a result, in the first step, the content placement algorithm selects user d_1 and places a random linear combination of the first two layers on the cache nodes adjacent to the user (caches h_1 and h_2). After the first step, the user nodes d_2, d_3, and d_4 have a degree equal to

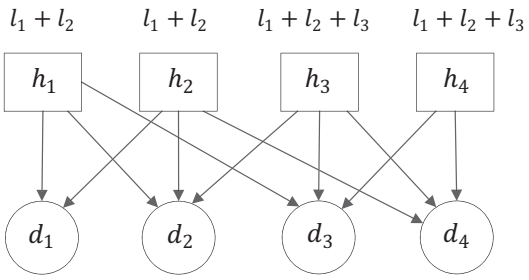


Figure 3.4: Content placement algorithm description. Coefficients are not shown for simplicity.

three. However, user d_2 has two adjacent cache nodes that are filled. Consequently, in the second round, the algorithm selects user d_2 and fills up the remaining cache that is adjacent to the user, which is cache h_3. In this case, $b_2 = 3$, $\phi_2 = 2$, and $r_2 = 2$. Therefore, we fill cache h_3 with a random linear combination of the $3 - 2 + 2 = 3$ first packets. In the last round, the degree, the number of filled-up caches that are adjacent to users d_3 and d_4, and their ranks are equal; thus, we select one of the users randomly. Assume that we choose user d_3. The algorithm fills cache h_4 with a random coded layer of the first three layers. Note that, in Figure 3.4, we do not show the coefficients of the coded packets for simplicity.

3.3 Simulation

In this section, we evaluate and compare the proposed network coding-based cache placement method (NCCP) with the best uncoded and triangular network coding placement strategies. For this purpose, we implemented a simulator in the MATLAB environment. In order to find the best uncoded placement, we try all of the possible original video placements on the caches, and we find the placement that provides the largest amount of available video layers to the users. We perform the same approach to find the best placement using the triangular network coding.

3.3.1 Setting

We perform our simulations on random topologies with different numbers of users and caches. For each setting, we run the simulation for 1000 randomly generated topologies (random adjacency of the users with the cache nodes). The plots in this chapter are based on the average outputs of the simulation runs. The following metrics are compared in the simulations:

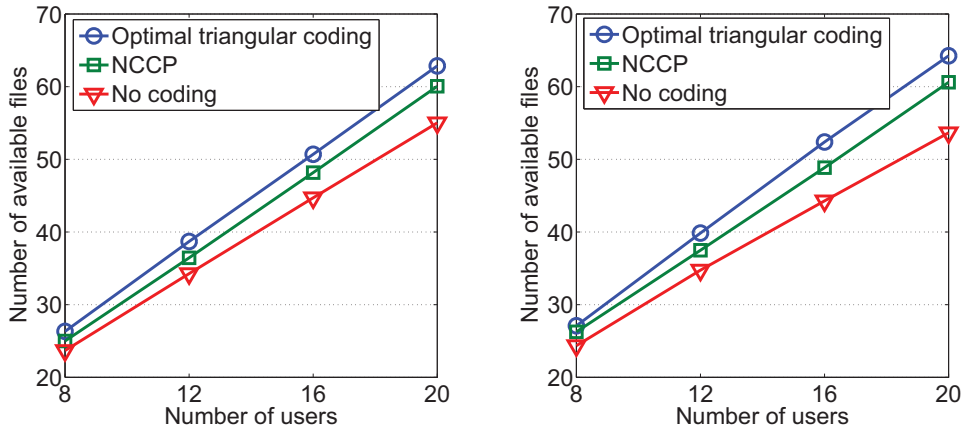
1. Amount of available video layers to the users: the total number of available layers to the users through the cache nodes. Note that, based on the proposed setting, a layer will not be useful to a user unless all of the layers with a smaller index are provided to the user first.

2. Average utility: the utility of each user is defined as the number of available layers to that user through its neighboring caches (the cumulative rank of the neighboring cache nodes) divided by the degree of the user.

3. Fairness: we define unfairness as the average difference between the number of available layers to each user and the average number of available layers to the users. We represent the number of layers available to the i-th user, unfairness, and fairness as c_i, ϕ', and ϕ, respectively. Based on the definition, we have $\phi' = \frac{\sum_{i=1}^{n} |c_i - \Gamma|}{n}$, where $\Gamma = \frac{\sum_{i=1}^{n} c_i}{n}$ is the average number of layers that are available to the users. Fairness is defined as $\phi = \frac{1}{\phi'}$.

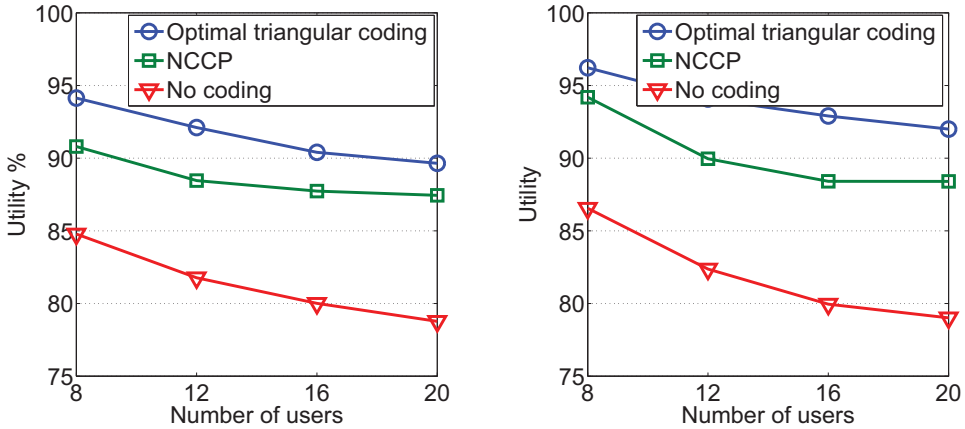
3.3.2 Results

In the first experiment, we evaluate the total number of video layers that are available to the users. In Figure 3.5 (a), the number of cache nodes and layers are 5 and 4, respectively. This figure shows that the optimal triangular network coding can provide the largest number of layers to the users, which is about 14% more than that of the best no-coding placement strategy. Also, the NCCP approach can provide the



(a) (b)

Figure 3.5: Total number of video layers that are available to the users. (a) $m = 5$, $e = 4$. (b) $m = 7$, $e = 4$.



(a) (b)

Figure 3.6: Utility of the content placement methods (the number of video layers that are available to a user divided by its degree). (a) $m = 5$, $e = 4$. (b) $m = 7$, $e = 4$.

users with about 9% more video layers than the best no-coding placement strategy. This difference might not be huge, but it should be noted that finding the best no-coding placement is NP-complete, and the complexity of finding such an optimal non-coding placement is $O(k^m \sum_{i=1}^{m} b_i)$. As we increase the number of users, the effect of network coding increases, which leads to a larger difference between the coding approaches and the no-coding method.

In Figure 3.5 (b), we change the number of cache nodes to 7, and we repeat the previous experiment. In this figure, the total number of available packets in the optimal triangular network coding method is about 17% more than that of the best no-coding placement. Also, the gain of the NCCP approach is up to 12% more than the no-coding method. The figure shows that the efficiency of the network coding increases as we increase the number of users.

Figures 3.6 (a) and (b) show the comparison between the average utility of the three approaches. We define the utility of a user as the fraction of the number of video layers that are available to that user and the degree of the user. The idea behind dividing by the user's degree is that, in the best case, the number of layers available to a user can be equal to its degree. It can be inferred from Figure 3.6 (a) that the utility of both the optimal triangular network coding and the NCCP methods are about 13% and 10% more than the best no-coding method, respectively. We change the number of caches to 7 in Figure 3.6 (b). By comparing Figures 3.6 (a) and (b), we can find that the utility in all of the methods increases as we increase the number of caches. However, increasing the number of caches has more of an effect on the NCCP method compared to that of the other approaches.

In the last experiment, we evaluate and compare the fairness of the methods. We define the unfairness (one divided by fairness) as the average difference between the number of layers that are available to each user and the average number layers that are available to the users. Figure 3.7 (a) shows that the fairness of the optimal triangular network coding and the NCCP approaches are up to 90% and 75% more than that of the best no-coding content placement strategy, respectively. We increase the number of cache nodes in Figure 3.7 (b) from 5 to 7 and evaluate the fairness of the approaches. In this setting, the fairness of the best no-coding approach is more than 56% and 50% less than that of the the best triangular network coding and the NCCP approaches, respectively.

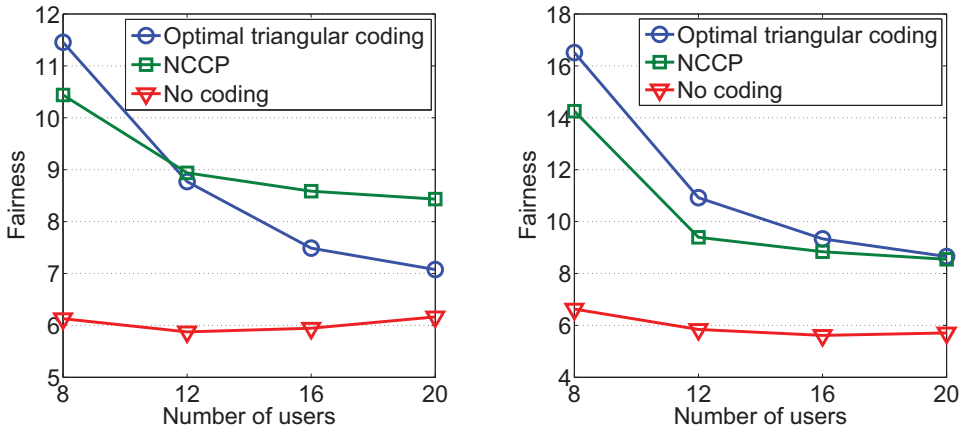


Figure 3.7: Fairness is equal to one divided by unfairness. Unfairness is defined as the average difference between the number of video layers that are available to each user and the average number of available video layers to the users). (a) $m = 5$, $e = 4$. (b) $m = 7$, $e = 4$.

3.4 Conclusion

In this chapter, we study the effects of network coding on increasing the amount of available data to the users through the cache nodes in cellular networks. Providing a larger portion of the data through the caches decreases the pressure on the base station (such as 4G) and enables the base station to serve more users. We propose a network coding-based content placement method, and we compare it with the best uncoded content placement strategy and the best triangular network coding strategy. Our method not only increases the amount of video layers that are available to the users, but it also results in a fair distribution of the data.

CHAPTER 4

MULTI-LAYER VIDEO STREAMING WITH HELPER NODES USING NETWORK CODING

In Chapter 3, we studied the application of network coding in content placement on caches with limited storage. In this chapter, we study a closely related problem, which is the application of network coding in video-on-demand streaming using helper nodes.

4.1 Introduction and Motivation

Recent studies have shown that multimedia streaming produces a significant portion of the traffic on the Internet. For example, 20-30% of the web traffic on the Internet is from YouTube and Netflix [60]. Thousands of hours of videos are uploaded on YouTube every day, and millions of hours of movies are available on Netflix, Hulu, and iTunes sites.

In order to provide a scalable and robust infrastructure that will support large and diverse on-demand streaming, the concept of helpers has been introduced, and the design of *helper-assisted video-on-demand* (VoD) systems has been explored [47, 49, 50]. Helpers are micro-servers with limited storage and bandwidth resources, which can download and store requested videos to be able to serve user requests. The helpers work in conjunction with a central server, which provides users with video files that cannot be obtained from their neighboring helpers (Figure 4.1). It is clear that the central server will be able to serve more users, as long as we can provide

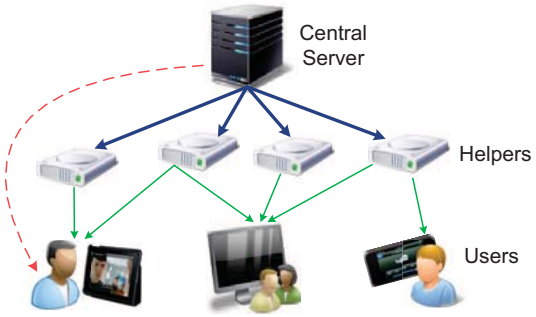


Figure 4.1: The system architecture.

(a) Original (b) Layer 1 (c) Layer 2

(d) Layer 3 (e) Layers 1 & 2 (f) Layers 2 & 3

Figure 4.2: Multi-layer video with 3 layers.

more portions of the requested videos through the helpers.

In addition to the use of helpers, we can benefit from *multi-layer videos* [3, 61] to provide a higher degree of scalable VoD systems. In multi-layer video, also called *multi-resolution codes* (MRC) or *scalable video coding* (SVC) [2, 62, 63], videos are divided into a base and enhancement layers. The base layer (layer 1) is required to watch the video, but the enhancement layers augment the quality of the video. Accessing more layers provides higher video quality, but the i-th enhancement layer is useless without all of the layers with a smaller index. Figures 4.2(b)-(d) show the constructed layers from an original image in Figure 4.2(a). Layer 1 is required by all

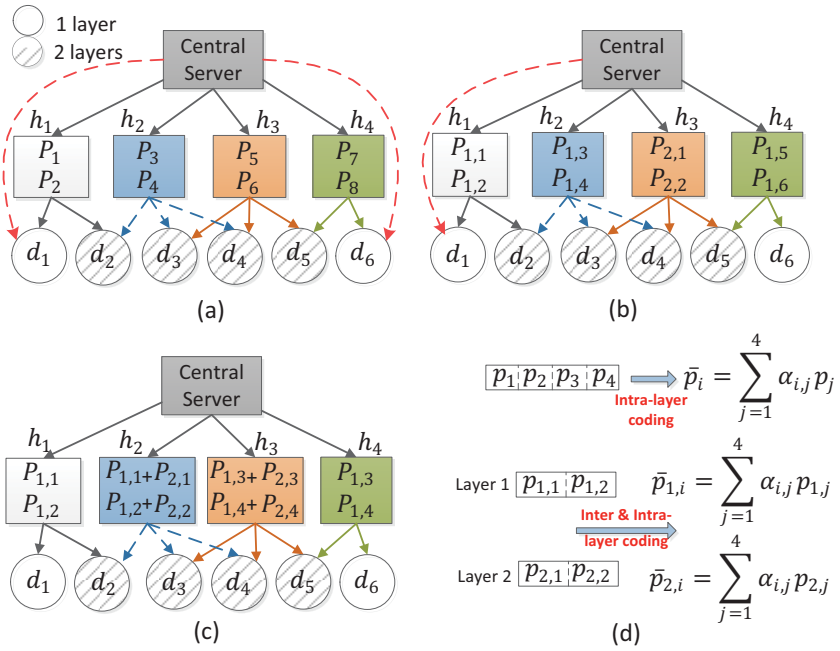


Figure 4.3: The advantage of using NC, (a) No multi-layer NC, (b) Intra-layer NC, (c) Inter- & intra-layer NC, (d) Coding schemes.

the users. Layers 2 and 3 are useless without layer 1, as depicted in Figures 4.2(c) and (d). Figure 4.2(f) shows that adding layers 2 and 3 together without layer 1 is useless, as well. Adding layer 2 to layer 1 increases the quality of the image, as shown in Figure 4.2(e).

In order to optimally use the resources, we need a mechanism to distribute the packets of the videos on the helpers, since, due to storage limitations, the helpers might not be able to store a full copy of the video. Network coding (NC) [27,28] helps to simplify the content distribution problem, and solves it in an efficient way [64]. Consider packets $p_1, ..., p_n$. In *linear NC*, each coded packet is in the form of $\sum_{i=1}^{n} a_i \times p_i$, where a_i is a coefficient. In this scheme, if a user has access to any n linearly independent coded packets, it can use Gaussian elimination to decode the coded packets and retrieve the original packets. In [1], it is shown that when the coefficients are selected randomly, there is a very high probability that the packets will be linearly independent. As a result of this scheme, which is called *random linear NC*, the coded packets

contribute the same amount of data to the users, which simplifies the distribution of the packets. Linear NC can be classified into *intra- or inter-layer NC*, depending on whether the coding is performed between the packets from the same layer or different layers, respectively.

Consider Figure 4.3, in which the users request a two-layer video, each of which consists of 2 packets; thus, the video contains 4 packets. The capacity of the helpers is equal to 2 packets. Users d_1 and d_6 request layer 1, and the other users need both layers. If we were to use the method in [50], the whole video would need to be downloaded for playing, as the method does not support multi-layer coding; thus, the video is considered to be 4 packets, p_1-p_4. Figure 4.3(a) shows an optimal video placement option based on the method in [50], in which random linear coded packets of p_1-p_4 are stored on the helpers (the no multi-layer NC is depicted in Figure 4.3(d)). In this case, users d_2-d_5 have access to 4 coded packets over p_1-p_4; thus, they can decode the coded packets using just the helpers. However, users d_1 and d_6 need to download 2 more packets from the server to decode the coded packets.

Figure 4.3(b) shows an optimal placement using intra-layer NC. The coding structure is shown in Figure 4.3(d). In this case, only user d_2 needs to download 2 packets from the server, so the load on the server is less than that of in Figure 4.3(a). Inter-layer NC can be used in conjunction with intra-layer NC to increase the efficiency of the content placement on the helpers. In Figure 4.3(c), we benefit from inter-layer NC. Users d_2-d_5 have access to 4 linearly coded packets over layers l_1 and l_2, so the server does not need to upload any layer. Moreover, users d_1 and d_6 have access to 2 linearly coded packets over layer l_1, which is sufficient for decoding the first layer.

Motivated by the intuition drawn from the example, in this chapter, we answer the following three questions: how should the packets of the videos be distributed over helper nodes? How should the helper nodes allocate their bandwidth to the users to minimize the load on the central server? And, lastly, how should we design

a coding scheme for content placement?

4.1.1 Setting

We consider a VoD system, where a video provider delivers a set of videos to a set of users. This video provider might consists of a set of servers. In the rest of this chapter, we refer to this video provider as central server. A group of helpers, which are micro-servers with limited storage and bandwidth help the central server in providing the users with the videos. These helpers might be set up by the video providers or third-party companies. In general, the users can also participate in the video distribution by allocating a portion of their local storage to work as a helper. Without loss of generality, we consider the helpers and users to be separate. We represent the set of helpers, users, and videos as H, D, and M, respectively. The users are stationary, and each helper covers a subset of the users (these sets do not need to be mutually exclusive). The coverage can be based on the geographic location or physical connection between the nodes. The k-th video m_k has a constant streaming rate r_k and size v_k. User d_i has a stationary request, denoted as q_i, which means d_i watches a single video at a time from the beginning to the end.

The helper h_j has storage and upload bandwidth capacities equal to S_j and B_j, respectively. If the helpers adjacent to user d_i can cumulatively provide the streaming rate of the requested video by the user, the whole video will be downloaded only from the helpers. Otherwise, the user will request the remaining portion of the video directly from the central server (Figure 4.1), which incurs some costs. In our problem, the cost is in terms of the load on the central server. Our objective in this work is to minimize the server's total upload rate to the users. In other words, we want to maximize the amount of data that helpers provide to their adjacent users.

The users have diverse network conditions and use different types of devices with different processing and bandwidth resources to watch the videos; thus, they might

desire different levels of video quality. In order to provide the users with different levels of video qualities, each video m_k is divided into e_k layers. The l-th layer of video m_k has a streaming rate and size equal to r_{kl} and v_{kl}, respectively. Each user d_i can subscribe to his or her desired number of layers c_i. The user can make a decision regarding c_i based on the quality of its network connection, or any other network limitations it may have. The l-th layer of a video is not useful unless all of the layers with a smaller index are available. Let the j-th helper node's upload rate to its adjacent user d_i over the l-th layer of video m_k be x_{ji}^{kl}. We represent the set of adjacent helpers to the user d_i and adjacent users to the helper h_j as $N(d_j)$ and $N(h_j)$, respectively. We consider the links to be reliable. Later, in Section 4.3, we extend our proposed methods to the case of unreliable connections. Table 4.1 summarizes the set of symbols used in this chapter.

We develop a distributed algorithm to find the optimal solution for the stationary case, where the number of users and helpers are fixed. We show via simulation results that, even for dynamic networks, our algorithm appears to converge to the optimal solution.

4.2 VoD with Multi-Layer Videos

In general, a helper might not be able to store a full copy of a video because of storage limitations. Moreover, a helper might provide more help to the central server by storing more partial videos, rather than by storing a small number of full videos [49]. The reason is that, by storing more partial videos, the helper can provide partial help to a greater number of users. Under this setting, in order to minimize the pressure on the central server, the following questions have to be addressed:

- *Content placement*: Which packets of which layers of each video should a helper store?

Table 4.1: The set of symbols used in this chapter.

Notation	Definition
d_i, D	The i-th user, the set of users
h_j, H	The j-th helper, the set of helpers
m_k, M	The k-th video, the set of videos
B_j/S_j	The bandwidth/capacity limit of helper h_j
r_{kl}/v_{kl}	The rate/size of layer l of video m_k
$N(d_i)/N(h_j)$	The set of adjacent helpers/users to d_i/h_j
x_{ji}^{kl}	Upload rate from h_j to d_i over layer l of video m_k
f_j^{kl}	The fraction of layer l of video m_k stored on helper h_j
e_k	The number of layers of video m_k
q_i	The video requested by user d_i
c_i	The number of layers requested by user d_i
x_j^k	Upload rate of h_j over video m_k (in live streaming)
z_{ji}^k	The download rate of user d_i from helper h_j over video m_k (in live streaming)
$\bar{\epsilon}_{ji}$	Reliability of the link between helper h_j and user d_i
$\bar{\epsilon}_i$	The reliability of the link between server and user d_i

- *Bandwidth allocation*: Which packets, and to which adjacent users, should each helper serve its stored content?

- *Coding scheme*: How should we design the coding scheme for the helpers?

Intra-layer NC helps to simplify the content placement problem on the helpers. As stated in the introduction, intra-layer NC also increases the efficiency of the content placement on the helpers. We divide each layer of a video into segments of n packets, according to the playing time of the video frame that they belong to. Figure 4.4(a) shows a video with 3 layers. Each segment of a layer consists of 2 packets, and the playback of segment 1 is before segment 2. In our intra-layer NC scheme, each coded packet of a segment is a random linear combination of the whole packets in that segment. In Figure 4.4(b), the coefficients are not shown for simplicity. For instance, $p_1 + p_2$ means $a_1p_1 + a_2p_2$, where a_i is a random coefficient. Coding too many packets together increases the time and memory complexity of coding and decoding. That is why we partition the packets into segments and code the packets of each segment

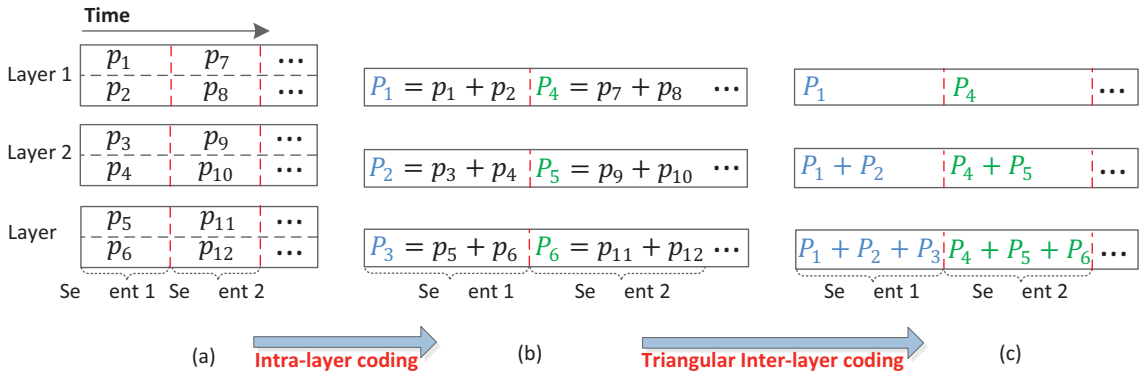


Figure 4.4: (a) Segmentation of a multi-layer video with 3 layers. (b) Intra-layer NC. (c) Joint inter- and intra-layer coding.

together. When using intra-layer NC, all of the coded packets from the helpers will contribute the same amount of information, and a user will be able to view the segment if it downloads any n linearly independent coded packets from the helpers that have the segment stored.

In order to enable a helper to serve any users watching video m, regardless of their playback time, we uniformly store the packets from each segment. Using this scheme, in order to store a fraction f of a layer of video m on helper h, we store $f \times n$ random linearly coded packets of each segment. Consider the video layer in Figure 4.5(a), in which each segment contains 4 packets. Assume that we want to store half of the video layer. We store 2 random linearly coded packets for each segment, as shown in Figure 4.5(b). Note that the 2 coded packets of each segment are different, since they have different random coefficients. For simplicity, we do not show the coefficients in the figure. Using this scheme, helper h can supply at the rate of $f \times r$ to the users that need video m, where r is the rate of the video. The use of intra-layer NC enables a flow-based model of the content, which changes our content placement and bandwidth allocation questions to finding:

- The rate at which coded packets of a video layer should be stored on a helper.

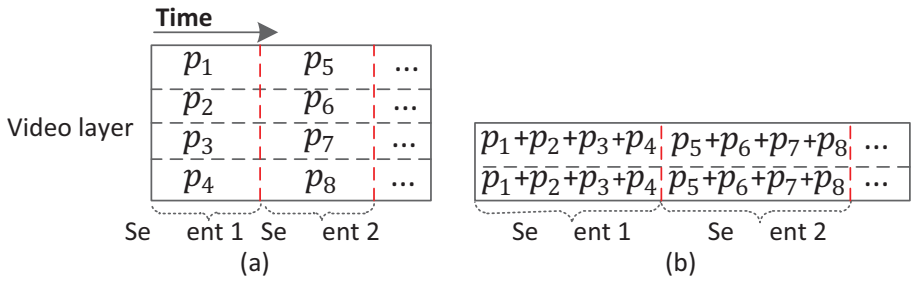


Figure 4.5: (a) Segments of a video layer. (b) Storing a half of the video layer.

- The rates at which coded packets of a video layer should be uploaded to a helper's adjacent users.
- The optimal coding scheme.

An alternative coding approach to random linear NC are rateless (fountain) codes [8, 9], which have less coding and decoding complexity. However, assuming that n packets are coded, $n(1+\Omega)$ coded packets are required for decoding, where Ω is a small overhead. It is shown that as n becomes larger, Ω becomes smaller. In this work we user random linear NC. However, it can be replaced with a rateless code.

A user might receive the packets of the current segment from the helpers and the server with different delays. As the packets of each segment are linearly coded with each other, the user is not able to decode the segment until it receives enough coded packets for the current segment. It means that, the user will be able to decode and watch the current segments when it received the last coded packet of the current segment from the helper or server that has the largest delay. In order to address this problem, which might result in the video lag problem (the video stopping as a result of delay in receiving the required packets), each user buffers the received coded packets and delays the playback so that the differences of the transmission delays and changes in the delays do not result in playback lags. The delay of the paths changes over time; thus, this buffering time should be large enough such that it does not result in lag problem. A large buffering time increases the waiting time to start the

payback; thus, the buffering time should not be too large. Computing the buffering time of the users is beyond the scope of this work.

4.2.1 Intra-Layer Network Coding

In this section, we assume that the links are reliable. As a result, minimizing the server load is equivalent to maximizing the help provided by the helpers. This optimization can be modeled as the following LP optimization problem:

$$\max \sum_{i,k:d_i\in D, m_k=q_i} \sum_{j,l:h_j\in N(d_i), l\leq c_i} x_{ji}^{kl} \tag{4.1}$$

$$s.t \quad x_{ji}^{kl} \leq f_j^{kl} r_{kl}, \ \forall j,i,l,k : d_i \in N(h_j), m_k{=}q_i, l{\leq}e_k \tag{4.2}$$

$$\sum_{i,k:d_i\in N(h_j), m_k=q_i} \sum_{l\leq c_i} x_{ji}^{kl} \leq B_j, \quad \forall j : h_j \in H \tag{4.3}$$

$$\sum_{k:m_k\in M} \sum_{l:l\leq e_k} f_j^{kl} v_{kl} \leq S_j, \quad \forall j : h_j \in H \tag{4.4}$$

$$\sum_{j:h_j\in N(d_i)} x_{ji}^{kl} \leq r_{kl}, \ \forall i,l,k : d_i \in D, l{\leq}c_i, m_k{=}q_i \tag{4.5}$$

$$0 \leq f_j^{kl} \leq 1, \quad \forall j,k,l : h_j \in H, m_k \in M, l \leq e_k \tag{4.6}$$

Figure 4.6: Problem A: LP optimization with intra-layer NC.

We refer to this problem as problem A. Objective function (4.1) is the summation of the helpers' upload rates to their adjacent users, over the subscribed layers of the requested videos. Function (4.1) is a linear function, so it is a concave function (Note that (4.1) is not strictly concave). We use the set of Constraints (4.2) to limit each helper's upload rate at the available service rate of the videos (the rate of stored videos). This upload rate differs for different layers of a video; thus, for each layer of a video, we have a separate constraint. The set of constraints (4.3) and (4.4) are feasibility constraints on bandwidth and storage. In more detail, the total upload rate of a helper and the total stored data on it cannot exceed its bandwidth and capacity limit. Note that in VoD applications, even in the case that the adjacent users to

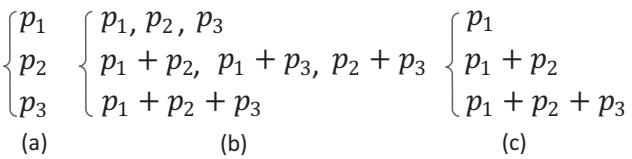


Figure 4.7: p_1+p_2 means $a_1p_1+a_2p_2$ (a_i is a random coefficient). (a) Original packets. (b) General form of random linear NC. (c) Triangular NC.

a helper watch the same video, their playback times are different; so, the helper needs to allocate separate bandwidths for each adjacent user. If two users adjacent to the same helper are watching the same video within a small time difference (e.g, 30 seconds), buffering at the users could be used to transmit the same information to both. However, for long videos it does not happen frequently.

It is sufficient for user d to download layer l of its requested video m_k at a rate equal to the streaming rate of the layer, since more than that value will not be useful. The set of Constraints (4.5) limits the aggregated download rate of the requested layers of video m to the user d at the rate of the layers. The set of Constraints (4.6) are the feasibility constraints on the fraction of stored video layers on the helpers, which limits them to be in the range of 0 and 1.

Assuming that each user is connected to all of the helpers, the number of variables x and f are equal to $|D||H|e$ and $|H||M|e$, where e is the maximum number of video layers. Also, the number of Constraints (4.2)-(4.6) are equal to $|D||H|e$, $|H|$, $|H|$, $|D|e$, and $|H||M|e$, respectively. Thus, the solution of the optimization can be calculated in polynomial time [65].

4.2.2 Joint Inter- and Intra-Layer Network Coding

In this section, we extend the formulation of problem A (Figure 4.6) to the case of joint inter- and intra-layer NC.

In the general form of random linear NC, each packet can be coded with any other packets. Thus, in the case of n packets, there are $2^n - 1$ random linear NC

possibilities. Figure 4.7(b) shows the seven possible ways to code the three packets in Figure 4.7(a) using the general form of NC. For simplicity, the random coefficients are not shown in the figures. In contrast with the general form of coding, in *triangular NC* [57, 66], each coded packet is a random linear combination of the first i packets, $\forall i : 1 \leq i \leq n$. In other words, the coded packets have a prefix form. Therefore, there are just n possibilities for coding n original packets. Figure 4.7(c) shows the three possible coded packets using the triangular coding scheme.

As stated in the introduction, inter-layer NC increases the provided help of the helpers. In order to benefit from joint inter- and intra-layer NC, we first perform intra-layer NC (Figure 4.4(b)). Then, we use the triangular NC scheme to code the intra-layer coded packets together. In our scheme, the coded packets of each segment of a video's l-th layer are a random linear combination of that segment in layers 1 to l. Figure 4.4(c) depicts the joint inter- and intra-layer coded packets, using the triangular scheme. In this figure, the packets of layer 1 are similar to those in the intra-layer approach, but the packets of layer 2 are a linear combination of layers 1 and 2. The packets of layer 3 are a random linear combination of all the 3 layers. The random coefficients are not shown for simplicity. For example, $P_1 + P_4$ means $a_1 P_1 + a_2 P_4$.

We prefer using triangular NC for two reasons. First, it limits the coding space of the coding problem, such that the convex optimization can be solved in polynomial time. Second, in our setting, we limit the number of received layers of each user to his request. Because of this limitation, the gain of the triangular NC is not less than the general form of NC. Before discussing the reason, we propose the following lemma:

Lemma 4.1. *A set of general linear coded packets (non-triangular) can be mapped to a set of triangular coded packets such that the rank of the set is preserved.*

Proof. Assume that A is the coefficient matrix of a set of general linear coded packets.

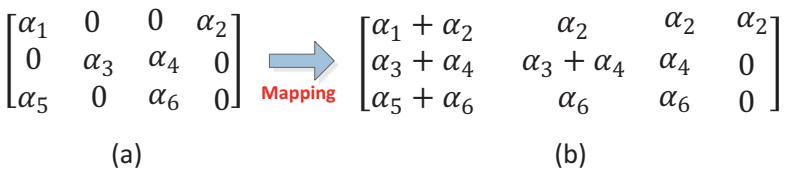


Figure 4.8: (a) Coefficient matrix of a set of general linear coded packets. (b) Coefficient matrix of a set of triangular coded packets.

Each row of A represents the coefficients of a coded packet. Consider z_i as the rightmost non-zero column in row i. In order to map row i to a triangular form, we should change it in way that the columns 1 to z_i becomes non-zero. The rank of a matrix is preserved under elementary column operations [67]. Therefore, we can multiply a column by a number and add the result to another column without changing the rank of the matrix.

Our mapping works as follows. We start from the rightmost column j and add it to its left column $j-1$. Then, we repeat this process by adding the new values of column $j-1$ to column $j-2$. This mapping is done for all of the columns. At the end, matrix A is converted to a coefficients matrix A', which contains the coefficients of a set of triangular coded packets. It should be noted that even in the case where adding column k to column $k-1$ results to a zero value for some of the cells in column $k-1$, we can multiply column k by a multiplier before adding it to column $k-1$ to prevent the zero cells to form. Figures 4.8(a) and (b) show the coefficient matrices of a set of general linear coded packets and their mapping to a set of triangular coded packets. □

Under the proposed setting, we do not provide a user with more layers than he has requested, which makes the gain of the triangular NC not less than that of the general form of inter-layer NC. Assume that the largest non-zero index in a general coded packet is d. Based on our setting, we should deliver c_i layers to user d_i. Thus, we do not transmit this coded packet to a user that has requested fewer than d

layers, and any changes in this coded packet does not have any impact on this user. Also, following Lemma 4.1, mapping a set of general linear coded packets to a set of triangular coded packets does not change the rank of the set. Therefore, the mapping does not have a negative impact on the users that requested at least d layers.

Assume that user d_i has subscribed to c_i layers, each of which contains n packets. We represent the received coded packets of the l-th coded layer as G_l. In [57], it is shown that under the triangular coding scheme, a user can decode all of the c_i layers if $\sum_{j=c_i-l+1}^{c_i} |G_j| \geq ln, \ \forall l \in [1, c_i]$. This means that the total number of received coded packets should be at least equal to $c_i n$. Also, the total number of received coded packets from layers 2 to c_i needs to be equal to or more than $(c_i - 1)n$. In general, the number of received coded packets from layers l to c_i should not be less than $(c_i - l + 1)n$, which gives us an insight into the following lemma:

Lemma 4.2. *Assume that the l-th layer contains n_l packets. Providing more than $\sum_{l=1}^{l'} n_l$ coded packets from the first l' coded layers is not useful to user d.*

Proof. Clearly, receiving more than n_1 coded packets from layer 1 is not useful to user d, since n_1 coded packets are enough to decode layer 1. In other words, receiving n_1 linearly independent coded packets results in a rank equal to n_1, and receiving more coded packets does not increase the rank. Coded layer 2 contains coded packets over the first two original layers. As a result, the user does not need more than $n_1 + n_2$ coded packets from the first two layers. With the same reasoning, receiving more than $\sum_{l=1}^{l'} n_l$ coded packets from the first l' coded layers is not useful. □

From Lemma 4.2, for the case of joint inter- and intra-layer NC, we can modify the formulation of problem A in Figure 4.6, as follows. The objective function is the same as that in problem A. Also, much like problem A, we have the set of Constraints (4.2),

(4.3), and (4.4). The set of Constraints (4.5) should be modified as:

$$\sum_{l=1}^{l'} \sum_{j:h_j \in N(d_i)} x_{ji}^{kl} \leq \sum_{l=1}^{l'} r_{kl}, \tag{4.7}$$
$$\forall i, l', k : 1 \leq l' \leq c_i, d_i \in D, m_k = q_i$$

which implies that the total upload rates of the first l' layers of the user-requested video should not be more than the total streaming rate of those layers. This set of constraints ensures that the helpers will not provide users with coded packets that are not useful for decoding. Moreover, we do not have the set of Constraints (4.6) anymore. The reason is that the coded video layers are joint inter- and intra layer. As a result, the variable f can be greater than 1. For example, consider a 2-layer video with n packets per layer. In the case of inter- and intra-layer coding, a user that does not receive any coded packet from layer 1 and receives $2n$ coded packets over layer 2 is able to decode both of the layers. Consequently, a helper can store the inter-layer coded packets of layer 2 at a rate equal to $f = 2$.

4.3 VoD with Unreliable Connections

We extend our methods for unreliable connections in two cases. In the first case, we assume that the losses happen just on the links between the helpers and the users. In the second case, the links from the server and helpers to the users might be lossy as well.

4.3.1 Reliable Server Links

As we assume that the links from the server to the users are reliable, maximizing the provided data to the users by the helpers minimizes the load on the server. The main difference between the case of unreliable helper links and the proposed methods

in Section 4.2 is that, here, the provided data (delivered successfully) from helper h_j to its neighboring user d_i over layer k is equal to $\bar{\epsilon}_{ji}x_{ji}^{kl}$, where $\bar{\epsilon}_{ji}$ is the reliability of the link between these two nodes. Based on this discussion, in the following sections, we modify the proposed linear programming equations in Section 4.2.

4.3.1.1 VoD with intra-layer Coding

In the case of existence of lossy helper nodes' links, the optimal bandwidth and storage can be found using the following LP:

$$\max \sum_{i,k:d_i\in D,m_k=q_i} \sum_{j,l:h_j\in N(d_i),l\leq c_i} \bar{\epsilon}_{ji}x_{ji}^{kl} \tag{4.8}$$

$$s.t \quad x_{ji}^{kl} \leq \frac{f_j^{kl}}{\bar{\epsilon}_{ji}} r_{kl}, \forall j,i,k,l : d_i \in N(h_j), m_k{=}q_i, l{\leq}e_k \tag{4.9}$$

$$\sum_{i,k:d_i\in N(h_j),m_k=q_i} \sum_{l\leq c_i} x_{ji}^{kl} \leq B_j, \quad \forall j : h_j \in H \tag{4.10}$$

$$\sum_{k:m_k\in M} \sum_{l:l\leq e_k} f_j^{kl} v_{kl} \leq S_j, \quad \forall j : h_j \in H \tag{4.11}$$

$$\sum_{j:h_j\in N(d_i)} \bar{\epsilon}_{ji}x_{ji}^{kl} \leq r_{kl}, \forall i,l,k : d_i \in D, l \leq c_i, m_k{=}q_i \tag{4.12}$$

$$0 \leq f_j^{kl} \leq 1, \quad \forall j,k,l : h_j \in H, m_k \in M, l \leq e_k \tag{4.13}$$

The objective function (4.8) is a summation of the delivered data from the helpers to their adjacent users. In contrast with the case of reliable links, here x_{ji}^{kl} is not equal to the received rate of the user d_i. As a result, the helpers can perform redundant transmission, since some of the transmissions will be lost. That is why, in the set of Constraints (4.9), we divide f_j^{kl} by $\bar{\epsilon}_{ji}$. The set of Constraints (4.10) and (4.11) ensure that the data uploaded by a helper and the stored data on the helper do not exceed its bandwidth and storage constraints. We use the set of Constraints (4.12) to limit the receiving rate of each layer of a video by a user, which is equal to $\bar{\epsilon}_{ji}x_{ji}^{kl}$, to the rate of that layer. We refer to this LP as reliable intra-layer coding.

4.3.1.2 VoD with Joint Inter and Intra-layer Coding

We can modify the proposed joint inter- and intra-layer LP to the case of unreliable helper links. Solving the following LP results in the optimal bandwidth and storage allocation when the links between helpers and the users are unreliable:

$$\begin{aligned}
&\max \sum_{i,k:d_i\in D,m_k=q_i} \sum_{j,l:h_j\in N(d_i),l\le c_i} \bar{\epsilon}_{ji} x_{ji}^{kl} \\
s.t \quad & x_{ji}^{kl} \le \frac{f_j^{kl}}{\bar{\epsilon}_{ji}} r_{kl},\ \forall j,i,k,l : d_i \in N(h_j), m_k = q_i, l \le e_k \\
& \sum_{i,k:d_i\in N(h_j),m_k=q_i} \sum_{l\le c_i} x_{ji}^{kl} \le B_j, \quad \forall j : h_j \in H \\
& \sum_{k:m_k\in M} \sum_{l:l\le e_k} f_j^{kl} v_{kl} \le S_j, \quad \forall j : h_j \in H \\
& \sum_{l=1}^{l'} \sum_{j:h_j\in N(d_i)} \bar{\epsilon}_{ji} x_{ji}^{kl} \le \sum_{l=1}^{l'} r_{kl}, \forall i,l',k : 1{\le}l'{\le}c_i, m_k{=}q_i
\end{aligned}$$

4.3.2 Lossy Server Links

In the previous sections we maximized the provided data from the helpers to the users, since that is equivalent to minimizing the load on the server. It is obvious that when the links between the server and users are lossy, maximizing the provided data from the helpers to the users might not minimize the load on the server. The reason is that, the links between the server and the users that could not download a 100% portion of their requested video from the helpers might be worth much more than the users that receive their requests in full from the helpers. As a result, we need to change the objective functions of the proposed linear programming from maximizing the provided help from the helpers to minimizing the server load as follows:

$$\min \sum_{i,k,l:d_i\in D,m_k=q_i,l\le c_i} \Big[r_{kl} - \sum_{j:h_j\in N(d_i)} \bar{\epsilon}_{ji} x_{ji}^{kl} \Big] / \bar{\epsilon}_i \tag{4.14}$$

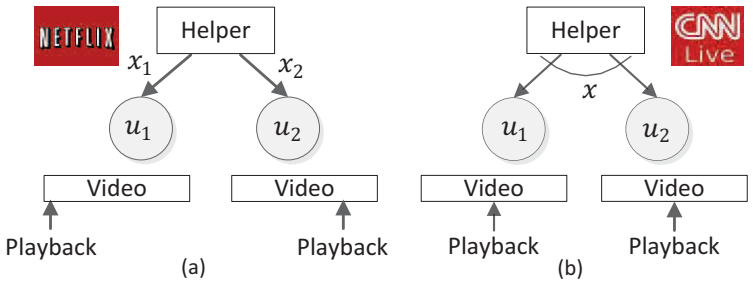


Figure 4.9: VoD vs. Live streaming.

We represent the delivery rate of the link between the user d_i and the central server as $\bar{\epsilon}_i$. The rate of the layer k is equal to r_{kl}, and the inner summation in Equation (4.14) is the total portion of the layer k provided to the user d_i through its neighboring helpers. Therefore, the summation over the users in Equation (4.14) results in the total load on the central server.

4.4 Wireless Live Streaming Applications

In this section, we show how the proposed solution for VoD can be extended for wireless live streaming (LS) applications. By 'LS' we are referring to applications where some videos are broadcast to the users, such as TV station channels or surveillance systems. In VoD, the users can play the videos asynchronously as depicted in Figure 4.9(a). However, in LS, the playback times of the users that watch the same video are synchronous (Figure 4.9(b)). Therefore, the main difference between LS and VoD is that, in LS, the helpers do not need to allocate separate bandwidths to their adjacent users that watch the same video, as shown in Figure 4.9(b).

In the case of VoD, the summation of the allocated bandwidth from each helper to its adjacent users should be less than or equal to its bandwidth. However, in LS, the summation of the allocated bandwidth from each helper for all of the videos should be less than or equal to its bandwidth. The reason for this is that multiple users might request the same video, and all of the users use the same broadcast packets.

Consider Figure 4.10, where users d_1 and d_4 requested the same video, m_1. Also, the users d_2 and d_3 requested video m_2. In this case, helper h_1 shares its bandwidth between videos v_1 and v_2, without assigning a separate bandwidth for each users. In order to formulate the case of LS, we represent the allocated bandwidth for the video m_k over helper h_j as x_j^k. The summation of these variables for each helper should be less than or equal to the helper's bandwidth. Also, the download rate of user d_i over video m_k from the helper h_j, which is represented as z_{ji}^k, should be less than or equal to x_j^k. The problem of LS in the case of single-layer videos can be formulated as follows:

$$\max \sum_{i,k:d_i \in D, m_k = q_i} \sum_{j:h_j \in N(d_i)} z_{ji}^k \tag{4.15}$$

$$s.t \quad x_j^k \leq f_j^k r_k, \quad \forall j,k : m_k \in M \tag{4.16}$$

$$\sum_{k:m_k \in M} x_j^k \leq B_j, \quad \forall j : h_j \in H \tag{4.17}$$

$$\sum_{k:m_k \in M} f_j^k v_k \leq S_j, \quad \forall j : h_j \in H \tag{4.18}$$

$$z_{ji}^k \leq x_j^k, \quad \forall i,j,k : h_j \in N(d_i), m_k = q_i \tag{4.19}$$

$$\sum_{k:m_k = q_i} \sum_{j:h_j \in N(d_i)} z_{ji}^k \leq r_k, \quad \forall i : d_i \in D$$

$$0 \leq f_j^k \leq 1, \quad \forall j,k,l : h_j \in H, m_k \in M, l \leq e_k$$

Objective function (4.15) is the summation of the download rates of users. The set of Constraints (4.16) ensures that the upload rate of a video by a helper cannot exceed the available service rate of the video. Constraints (4.17), (4.18), and (4.6) are feasibility constraints on bandwidth and storage. We limit the download rate of a user from a helper to the upload rate of its requested movie, using the set of Constraints (4.19). We refer to our method as wireless live streaming (WLS). For simplicity, we formulated the problem for the case of single-layer videos. The

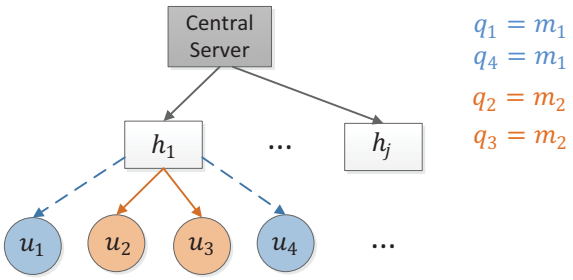


Figure 4.10: Live streaming.

formulation can be easily extended for multi-layer videos.

4.5 Distributed Solution

In this section, we solve the proposed optimization problem A (Figure 4.6) for the case of multi-layer VoD streaming, using intra-layer NC in a distributed way. The same approach can be used to find a distributed solution for the other settings. The idea is to solve the Lagrange dual of the problem using the gradient method [68]. In this way, the helpers start from empty storage, and gradually update their storage and bandwidth allocation, based on the exchanged Lagrange variables between them and their adjacent users.

The objective function (4.1) is not strictly concave, due to the presence of a linear summation. Consequently, a direct application of standard gradient iterative method might lead to multiple solutions. In this case, the output of an iterative method may oscillate between multiple feasible solutions. In order to overcome the problem due to the lack of strict concavity, we can apply the Proximal method described in [69], page 233. The idea behind the Proximal method is to add quadratic terms to the objective function and make it strictly concave. A detailed description of the Proximal method is in [69,70]. To apply the Proximal method, we introduce auxiliary variables y_{ji}^{kl}. By

using the Proximal method, the optimization becomes:

$$\max \sum_{i,k:d_i \in D, m_k = q_i} \sum_{j,l:h_j \in N(d_i), l \leq c_i} \left(x_{ji}^{kl} - (x_{ji}^{kl} - y_{ji}^{kl})^2 \right) \tag{4.20}$$

subject to the set of Constraints (4.2), (4.3), (4.4), (4.5), and (4.6).

The optimal solution of (4.20) is also the solution of (4.1). Let $\vec{x^*}$ and $\vec{f^*}$ be the optimal solution of (4.1) then, $\vec{x} = \vec{x^*}$, $\vec{f} = \vec{f^*}$, and $\vec{y} = \vec{x}$ is the maximizer of (4.20). The standard proximal method iteratively works as follows:

1. Fix $\vec{y}(t)$ and maximize (4.20) with respect to variables $\vec{x}(t)$ and $\vec{f}(t)$.

2. Set $\vec{y}(t+1) = \vec{x}(t)$, increment t, and go back to step 1.

Since the Slater condition holds (see reference [71]), there is no duality gap between the primal and the dual problems. Therefore, we can use the dual approach to solve the problem. Let λ_1^{jil}, λ_2^j, λ_3^j, and λ_4^{il} be the Lagrange variables for Constraints (4.2), (4.3), (4.4), and (4.5), respectively. Here, i, j, and l are corespondent to the indices in the set of Constraints (4.2) to (4.5). The Lagrange function of (4.20) is:

$$\begin{aligned}
&L(\vec{x}, \vec{f}, \vec{y}, \vec{\lambda}) = \\
&\sum_{i,k:d_i \in D, m_k = q_i} \sum_{j,l:h_j \in N(d_i), l \leq c_i} \left(x_{ji}^{kl} - (x_{ji}^{kl} - y_{ji}^{hl})^2 \right) \\
&- \sum_{j,i:h_j \in H, d_i \in N(h_j)} \sum_{k,l:m_k = q_i, l \leq c_i} \lambda_1^{jil} (x_{ji}^{kl} - f_j^{kl} r_{kl}) \\
&- \sum_{j:h_j \in H} \lambda_2^j \Big(\sum_{i,k,l:d_i \in N(h_j), m_k = q_i, l \leq c_i} x_{ji}^{kl} - B_j \Big) \\
&- \sum_{j:h_j \in H} \lambda_3^j \Big(\sum_{k,l:m_k \in M, l \leq e_k} f_j^{kl} v_{kl} - S_j \Big) \\
&- \sum_{i,k,l:d_i \in D, m_k = q_i, l \leq c_i} \lambda_4^{il} \Big(\sum_{j:h_j \in N(d_i)} x_{ji}^{kl} - r_{kl} \Big)
\end{aligned}$$

Algorithm 3 Calculation of $\vec{f}$ (for helper h_j)

$rem = S_j$, calculate $\gamma_j^{kl} \quad \forall k, l : m_k \in M, l \leq e_k$
for each f_j^{kl} in descending order of γ_j^{kl} **do**
  if $\gamma_j^{kl} > 0$ and $rem > 0$ **then**
    if $rem > v_{kl}$ **then**
      set $f_j^{kl} = 1$, $rem = rem - v_{kl}$
    else
      set $f_j^{kl} = \frac{rem}{v_{kl}}$, $rem = 0$
  else
    $f_j^{kl} = 0$, $rem = 0$

By rearranging the terms, we have:

$$
\begin{aligned}
L(\vec{x}, \vec{f}, \vec{y}, \vec{\lambda}) = & \sum_{i,k:d_i \in D, m_k = q_i} \quad \sum_{j,l:h_j \in N(d_i), l \leq c_i} \big[(1 - \lambda_1^{jil} - \lambda_2^j \\
& - \lambda_4^{il}) x_{ji}^{kl} - (x_{ji}^{kl} - y_{ji}^{kl})^2\big] \\
& + \sum_{j,i:h_j \in H, d_i \in N(h_j)} \quad \sum_{k,l:m_k = q_i, l \leq c_i} \lambda_1^{jil} f_j^{kl} r_{kl} \\
& - \sum_{j:h_j \in H} \quad \sum_{k,l:m_k \in M, l < e_k} \lambda_3^j f_j^{kl} v_{kl}
\end{aligned}
$$

By a simple change of variables, the Lagrange function is separable in $\vec{x}$ and $\vec{f}$, and we can rewrite it as:

$$
\begin{aligned}
& L(\vec{x}, \vec{f}, \vec{y}, \vec{\lambda}) = \\
& \sum_{\substack{i,k:d_i \in D \\ m_k = q_i}} \sum_{\substack{j,l:h_j \in N(d_i) \\ l \leq c_i}} \big[(1 - \lambda_1^{jil} - \lambda_2^j - \lambda_4^{il}) x_{ji}^{kl} - (x_{ji}^{kl} - y_{ji}^{kl})^2\big] \\
& + \sum_{\substack{j,k:h_j \in H \\ m_k \in M}} \Big(\sum_{\substack{i,l:d_i \in N(h_j) \\ l \leq c_i}} \lambda_1^{jil} r_{kl} - \sum_{l:l < e_k} \lambda_3^j v_{kl} \Big) f_j^{kl}
\end{aligned} \tag{4.21}
$$

The objective function of the dual problem is $D(\vec{y}, \vec{\lambda}) = \max_{\vec{x} \geq 0, \vec{y} \geq 0} L(\vec{x}, \vec{f}, \vec{y}, \vec{\lambda})$. The dual problem itself is $\min_{\lambda \geq 0} D(\vec{y}, \vec{\lambda})$. The dual optimization can be solved using the

gradient method [68]. The updates of the Lagrange variables are as follows:

$$\lambda_1^{jil}(t+1) = \Big[\lambda_1^{jil}(t) + \beta(x_{ji}^{kl}(t) - f_j^{kl}(t) r_{kl})\Big]^+,$$

$$\forall j,i,k,l : h_j \in H, d_i \in N(h_j), m_k = q_i, l \leq e_k$$

$$\lambda_2^{j}(t+1) = \Bigg[\lambda_2^{j}(t) + \beta \sum_{i,k:d_i \in N(h_j), m_k = q_i} \sum_{l \leq c_i} x_{ji}^{kl}(t) - B_j\Bigg]^+,$$

$$\forall j : h_j \in H$$

$$\lambda_3^{j}(t+1) = \Bigg[\lambda_3^{j}(t) + \beta \sum_{k:m_k \in M} \sum_{l:l \leq e_k} f_j^{kl}(t) v_{kl} - S_j\Bigg]^+,$$

$$\forall j : h_j \in H$$

$$\lambda_4^{il}(t+1) = \Bigg[\lambda_4^{il}(t) + \beta \sum_{j:h_j \in N(d_i)} x_{ji}^{kl}(t) - r_{kl}\Bigg]^+,$$

$$\forall i,k,l : d_i \in D, m_k = q_i, l \leq c_i$$

where $[.]^+$ denotes the projection on $[0, \infty)$. Also, by setting the first derivative of (4.21) with respect to $\vec{x}$ being equal to zero, the optimal $\vec{x}$ can be calculated as follows:

$$x_{ji}^{kl}(t+1) = \frac{1 - \lambda_1^{jil}(t) - \lambda_2^{j}(t) - \lambda_4^{il}(t)}{2} + y_{ji}^{kl}(t)$$

$$\forall j,i,k,l : h_j \in H, d_i \in N(h_j), m_k = q_i, l \leq e_k$$

Algorithm 3 illustrates the computation of $\vec{f}$. Here, γ_j^{kl} is the multiplier of f_j^{kl} in Equation (4.21), which equals $\gamma_j^{kl} = \sum_{\substack{j,k:h_j \in H \\ m_k \in M}} (\sum_{\substack{i,l:d_i \in N(h_j) \\ l \leq c_i}} \lambda_1^{jil} r_{kl} - \sum_{l:l<e_k} \lambda_3^{j} v_{kl})$. Also, *rem* is the free space of helper h_j. The idea here is that, in order to maximize the second line in Equation (4.21), we should give a greater value to the fraction of the videos with a greater γ value. On the other hand, the fraction of videos with a negative γ value should be equal to zero. Therefore, for each helper, we sort the γ_j^{kl} in descending order of their values, and we start to fill the helpers with videos that have a greater γ. Let us represent the set of videos requested by the neighboring

Algorithm 4 Users' Protocol (for user d_i)

1: **Initialization**: Send the request and the number of desired layers to the adjacent helpers. Set $\lambda_4^{il}(1,0)=0$
2: **Iteration Phase** at the τ-th iteration
3: **for** $t=0,...,T-1$ perform the following step sequentially
4: send $\lambda_4^{il}(\tau,t+1)=[\lambda_4^{il}(\tau,t)+\beta(\sum_{j:h_j\in N(d_i)} x_{ji}^{kl}(\tau,t)-r_{kl})]^+ \quad \forall l: l\leq c_i$ to all adjacent helpers.
5: $\vec{\lambda}_4(\tau+1,0)=\vec{\lambda}_4(\tau,T)$

users of helper h_j as M_j. For each layer of a video we have a f_j^{kl} variable; thus, the number of executions of the *for* loop in Algorithm 3 is equal to the total number of layers of the videos in M_j. Therefore, the time complexity of Algorithm 3 is in order of $O(\sum_{k\in M_j} e_k)$. If we represent the maximum number of layers as e, the complexity will be in the order of $O(|M_j|e)$.

We can define two iterative levels for the distributed algorithm [70]. In the inner loop, we fix the auxiliary variables $\vec{y}$ and update $\vec{x}$, $\vec{f}$, and $\vec{\lambda}$, for T times. We run the outer loop τ times, in which we set $\vec{y}(\tau+1,0)=\vec{x}(\tau,T)$. The users' and helpers' policies are shown in Algorithms 4 and 5, respectively. The users can receive x_{ij}^{km} from the helpers explicitly. However, they can compute it based on the actual receiving data rate from their neighboring helpers. The convergence of our algorithm can be proven using a technique similar to [72]. We omit the proof for brevity, and in our simulations, we empirically verify the convergence.

The *for* loop in Algorithms 4 runs for T iterations. Also, user d_i needs to calculate c_i different λ_4^{il} variables. For each of these Lambda variables, the the summation in line 5 has a complexity of $O(|N(d_i)|)$. Therefore, the complexity of Algorithms 4 is $O(|N(d_i)|c_iT)$. The *for* loop in Algorithms 5 runs for T iterations. Line 5 needs to be calculated for $|N(h_j)|e$, where e is the maximum number of video layers. The complexity of lines 6, 7, and 8 is $O(e|M_j|)$. Also, in line 9, the helper node runs Algorithm 3. As a result, the complexity of Algorithms 5 is in order of $O(Te(|M_j|+|N(h_j)|))$.

Algorithm 5 Helpers' Protocol (for helper h_j)

1: **Initialization**: Set $x_{ji}^{kl}(1,0) = 0$, $f_i^{kl}(1,0) = 0$, $\lambda_1^{jil}(1,0) = 0$, $\lambda_2^j(1,0) = 0$, $\lambda_3^j(1,0) = 0$, $y_{ji}^{kl}(1,0) = 0$

2: **Iteration Phase** at the τ-th iteration

3: **for** $t = 0, ..., T-1$ perform the following steps sequentially

4: $\lambda_1^{jil}(\tau, t+1) = [\lambda_1^{jil}(\tau,t) + \beta(x_{ji}^{kl}(\tau,t) - f_j^{kl}(\tau,t) r_{kl})]^+$

5: $\lambda_2^j(\tau, t+1) = [\lambda_2^j(\tau,t) + \beta \sum_{i:d_i \in N(h_j)} \sum_{l \leq c_i} x_{ji}^{kl}(\tau,t) - B_j]^+$

6: $\lambda_3^j(\tau, t+1) = [\lambda_3^{jl}(\tau,t) + \beta(\sum_{k:m_k \in M} \sum_{l:l \leq e_k} f_j^{kl}(\tau,t) v_{kl} - S_j)]^+$

7: $x_{ji}^{kl}(\tau, t+1) = \frac{1 - \lambda_1^{jil}(\tau,t) - \lambda_2^j(\tau,t) - \lambda_4^{il}(\tau,t)}{2} + y_{ji}^{kl}(\tau,t)$

8: run algorithm 1 to calculate $\vec{f}(\tau, t+1)$

9: $\vec{y}(\tau+1, 0) = \vec{x}(\tau, T)$, $\vec{x}(\tau+1, 0) = \vec{x}(\tau, T)$, $\vec{\lambda_1}(\tau+1, 0) = \vec{\lambda_1}(\tau, T)$, $\vec{\lambda_2}(\tau+1, 0) = \vec{\lambda_2}(\tau, T)$, $\vec{\lambda_3}(\tau+1, 0) = \vec{\lambda_3}(\tau, T)$

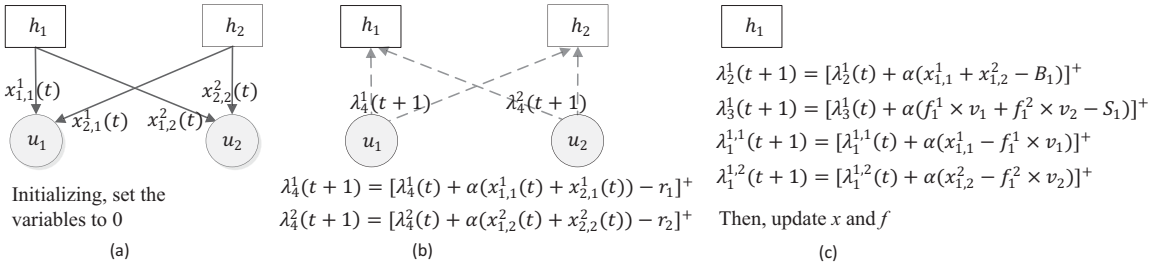


Figure 4.11: Example of the distributed solution.

Consider the topology in Figure 4.11, in which users d_1 and d_2 requested movies m_1 and m_2, respectively. In order to simplify the example, we assume that the movies contain one layer. As a result, we remove the index l from all of the notations and equations. Also, without loss of generality, we discuss the process for one round and we do not show variable τ in the equations. At the beginning, the users d_1 and d_2 set up their Lagrange variables λ_4^1 and λ_4^2 to zero, respectively. Moreover, helpers h_1 and h_2 initialize their correspondent variables x_{ji}^k, f_j^k, $\lambda_1^{j,i}$, λ_2^j, and λ_3^j to zero, or any other default value (Figure 4.11(a)). The variables y_{ji}^k are initialized to the values of their correspondent variables x_{ji}^k by the helpers.

Following the initializing phase, the users update their Lagrange variables λ_4^1

and λ_4^2 based on the received bandwidths x_{ji}^k from the helpers. User d_1 calculates $\lambda_4^1(t+1) = [\lambda_4^1(t) + \beta(x_{1,1}^1(t) + x_{2,1}^1(t) - r_1)]^+$, and transmits it to the helpers h_1 and h_2. Similarly, as shown in Figure 4.11(b), user d_2 calculates $\lambda_4^2(t+1) = [\lambda_4^2(t) + \beta(x_{1,2}^1(t) + x_{2,2}^1(t) - r_2)]^+$ and sends it to the helpers. In the next round, the helpers use the received Lagrange variables to update their Lagrange variables, and modify the bandwidth and storage assignment.

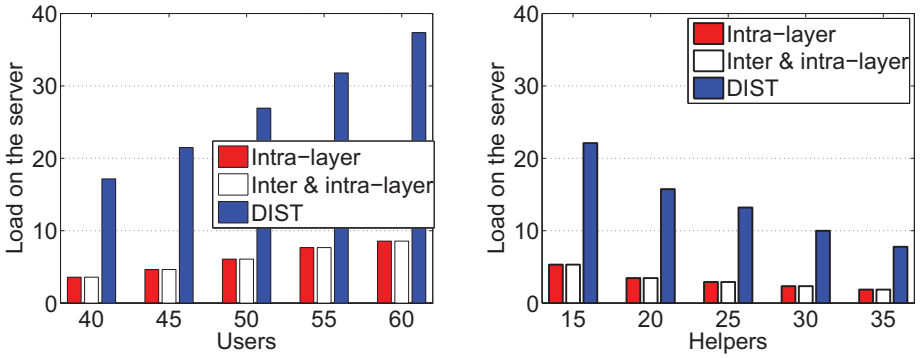
In Figure 4.11(c) we show the updates for helper h_1. The updates for helper h_2 can be done in a similar way. The helper h_1 first updates $\lambda_1^{1,1}$ and $\lambda_1^{1,2}$ using equations $\lambda_1^{1,1}(t+1) = [\lambda_1^{1,1}(t) + \beta(x_{1,1}^1(t) - f_1^1(t)r_1)]^+$ and $\lambda_1^{1,2}(t+1) = [\lambda_1^{1,2}(t) + \beta(x_{1,2}^2(t) - f_1^2(t)r_2)]^+$, respectively. These Lagrange variables correspond to the set of Constraints (4.2). Moreover, helper h_1 calculates λ_2^1 and λ_3^1 as shown in Figure 4.11(c). Then, the helper runs Algorithm 3 to calculate f_1^1 and f_1^2. Finally, helper h_1 assigns its bandwidth to the users d_1 and d_2, using equations $x_{1,1}^1(t+1) = [1 - \lambda_1^{1,1}(t) - \lambda_2^1(t) - \lambda_4^1(t)]/2 + y_{1,1}^1(t)$ and $x_{1,2}^1(t+1) = [1 - \lambda_1^{1,2}(t) - \lambda_2^1(t) - \lambda_4^1(t)]/2 + y_{1,2}^1(t)$, respectively. The processes in Figure 4.11(b) and (c) are repeated periodically. After T times of running these steps, the variables $y_{1,1}^1$ and $y_{1,2}^2$ are set to $x_{1,1}^1$ and $x_{1,2}^2$, respectively, and the updates by the users and helpers are repeated.

4.6 Simulation Results

We compare our proposed methods with an optimal single layer VoD streaming using the helpers. We refer to this scheme as DIST methods. We also study the convergence of the proposed distributed method under the static and dynamic cases. For this purpose, we implemented a simulator in the Matlab environment. For solving the linear programming optimizations, we use the Linprog optimization toolbox which is embedded in Matlab. We assume that the popularity of the videos and the number of subscribed layers by each user are uniformly distributed. We evaluate the methods on 100 random topologies, and use the average output of the simulation for plots

Table 4.2: The ranges of the parameters in the simulations.

Video's rate	Video's size	Bandwidth capacity	Storage capacity	Number of adjacent helpers to a user
[1,2] kbps	[0.5,2] MB	[2,4] kbps	[0.5,2] MB	[1,3]



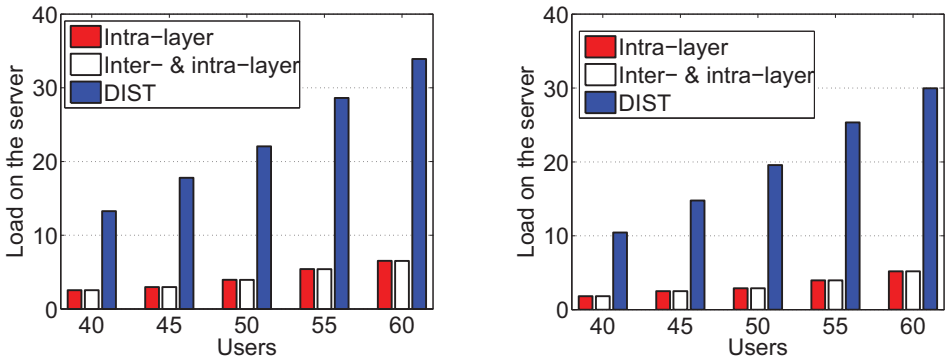
(a) Number of helpers: 20 (b) Number of users: 40

Figure 4.12: Server's load in kbps, VoD, 5 videos, 5 layers.

of this section. By random topologies we mean randomly connecting the users to the helpers and randomly setting the bandwidth limit of the helpers. Number of adjacent helpers to user d_i is randomly selected in the range of [1,3]. Assuming that this number is g_i, we connect user d_i to g_i helpers. The range of a video's rate, size, storage capacity, bandwidth capacity, and number of adjacent helpers to each user are randomly chosen in the ranges shown in Table 4.2.

4.6.1 Performance

In Figure 4.12(a), we compare the loads on the central server. Each video contains 5 layers, and the number of requested layers by each user is randomly chosen in the range of $[1, 5]$. The other parameters are shown in Table 4.2. The figure shows that the result of the joint inter- and intra-layer coding is very close to that of the intra-layer coding. Moreover, the server's load in our methods is up to 75% less than that of the DIST approach. The figure shows that the slope of the load in the DIST method is more than that of our proposed approaches, which is due to to users' greater amount



(a) $a = 2$ (b) $a = 3$

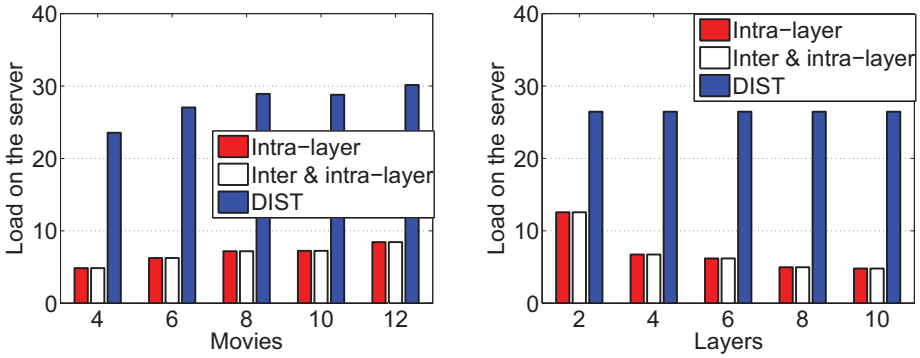
Figure 4.13: Server's load in kbps, VoD power-law distribution of the videos popularity, 10 helpers, 5 layers.

of required resources when using the DIST method.

In our next experiment, we study the effect that the number of helpers has on the server's load. It is clear that more helpers can provide more portions of the videos, due to more available capacity and bandwidth resources. As a result, the server's load in all of the methods decreases as we increase the number of helpers, as illustrated in Figure 4.12(b).

We repeat the experiment in Figure 4.12(a) for the case of videos' popularity with a power-law distribution [73], i.e., the fraction of movies with d request (denoted as P_d) is proportional to d^{-a}: $P_d = (a-1)d^{-a}$. Here, a is the power-law distribution parameter, which usually satisfies $a \in [2, 3]$. In Figure 4.13(a), we set $a = 2$. The other parameters are the same as those in Figure 4.12(a). Compared to Figure 4.12(a), the load on the server reduces in Figure 4.13(a). The reason is that, in this case, the probability of common requests increases; thus, the storage of the helpers can be used more efficiently. We increase a to 3 and repeat the experiment. Figure 4.13(b) shows that as the popularity of few videos increase, the advantage of using the helpers increases as well; as a result, the load on the server reduces.

Figures 4.14(a) and (b) depict the effect of the number of videos and layers of the server's load. The simulation's parameters are chosen randomly in the ranges shown



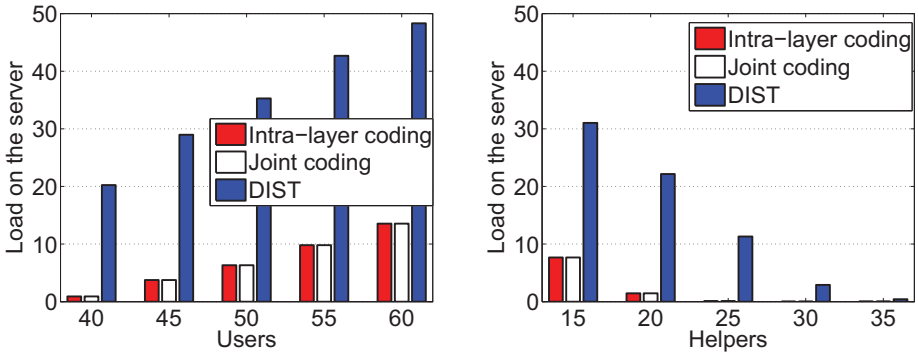
(a) Number of layers: 5 (b) Number of videos: 5

Figure 4.14: Server's load in kbps, VoD, 50 users, 20 helpers.

in Table 4.2. The server's load increases as we increase the number of videos. This is because, as we increase the number of choices, the number of common requests decreases; thus, the helpers need to store more videos, which is not feasible due to the storage limitations. More layers give the users the choice to select videos with a lower quality, which decreases the load on the server, as shown in Figure 4.14(b). The server's load is almost fixed in the DIST method, since DIST is a no-layer approach.

In order to validate the result of Figure 4.12(a), we repeat the same experiment in the case of geographic distribution of the nodes. For this purpose, we distribute the nodes randomly in a 4×4 M area, and set the transmission range of the nodes to 1 M. Figure 4.15 (a) shows the load on the server of the different methods. In this figure, the video rates, video sizes, storage capacity, and bandwidth capacity of the helpers are in the range of $[1, 2]$, $[0.5, 2]$, $[0.5, 2]$, and $[1, 3]$, respectively. Most similar to Figure 4.12 (a), the load on the server increases as we increase the number of users, which is due to more video requests. In Figure 4.15(a), the average number of adjacent helpers to the users is equal to 4.2. Figure 4.15(b) shows the effect of changing the number of helpers on the server's load. As it is expected, most similar to Figure 4.12(b), the load on the server decreases as we increase the number of helpers.

As we stated in the introduction, there are cases where the inter-layer NC reduces the server's load. However, Figures 4.12 and 4.14 show that the server's load using

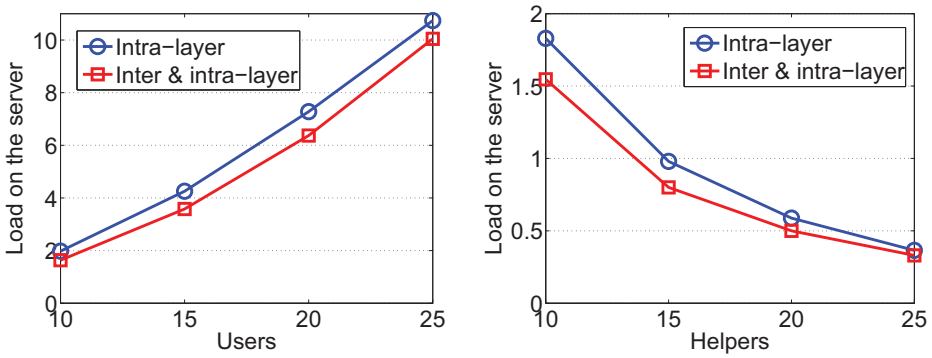


(a) Number of helpers: 20 (b) Number of users: 40

Figure 4.15: A server's load (in kbps) in the case of geographic distribution of the user and helpers, VoD, 5 videos, 5 layers.

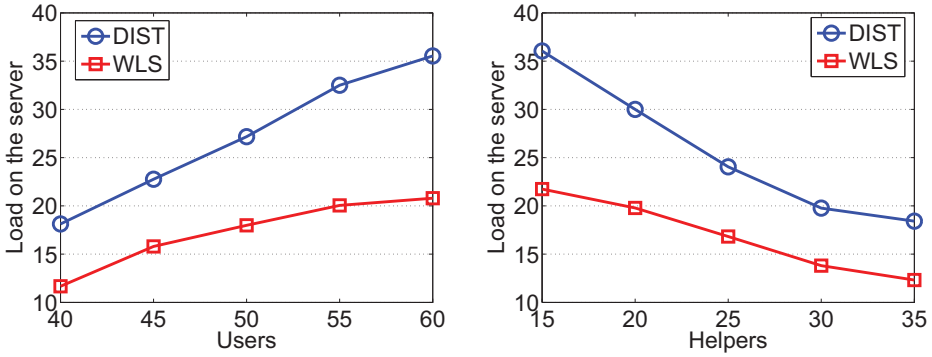
joint NC and just intra-layer coding are very close. In order to study the benefit of inter-layer coding, we repeat the first experiment with a single video, as to eliminate competition between the users with different video requests. The helpers' bandwidths are in the range of $[5, 10]$, and the video size, video rate, and the storage capacities are set to 4, 4, and 1. Also, the degree of each user is in the range of $[1, 4]$, and we set the number of requested layers of each user to its degree. Figure 4.16(a) shows that the server's load using joint coding is up to 17% less than that of the intra-layer coding method. Figure 4.16(b) shows that, as we increase the number of helpers, the difference between the methods decreases, which is due to the availability of a high percentage of the video through the helpers in both methods. Based on our observation, we can find that when the users compete to receive different videos, and the bandwidth is the bottleneck, inter-layer NC cannot increase the content available to the users. As a result, we can conclude that inter-layer NC is not very useful in practice.

Figures 4.17(a) and (b) show the comparison between the server's load in the DIST and WLS (wireless live streaming) methods. The experimental parameters are chosen randomly in the ranges shown in Table 4.2. In the case of LS, the playback time of the users that watch the same video are synchronous. Thus, in the WLS method, the



(a) Number of helpers: 10 (b) Number of users: 10

Figure 4.16: The advantage of joint inter- & intra-layer coding over intra-layer coding, VoD, 4 layers.



(a) Number of helpers: 20 (b) Number of users: 50

Figure 4.17: Live streaming, Load on the server in kbps, 5 videos, single layers.

helpers do not assign a separate bandwidth to the users that watch the same video, which results in providing more portions of the videos through the helpers. As a result, the server load in the WLS method is less than that of the DIST method. In Figure 4.17(a), the slope of DIST is more than that of the WLS, which means that the helpers do not have enough free bandwidth to support more users. On the other hand, in Figure 4.17(b), WLS has less slope than the DIST method since, even in the case of 15 helpers, the users receive a large portion of their requests.

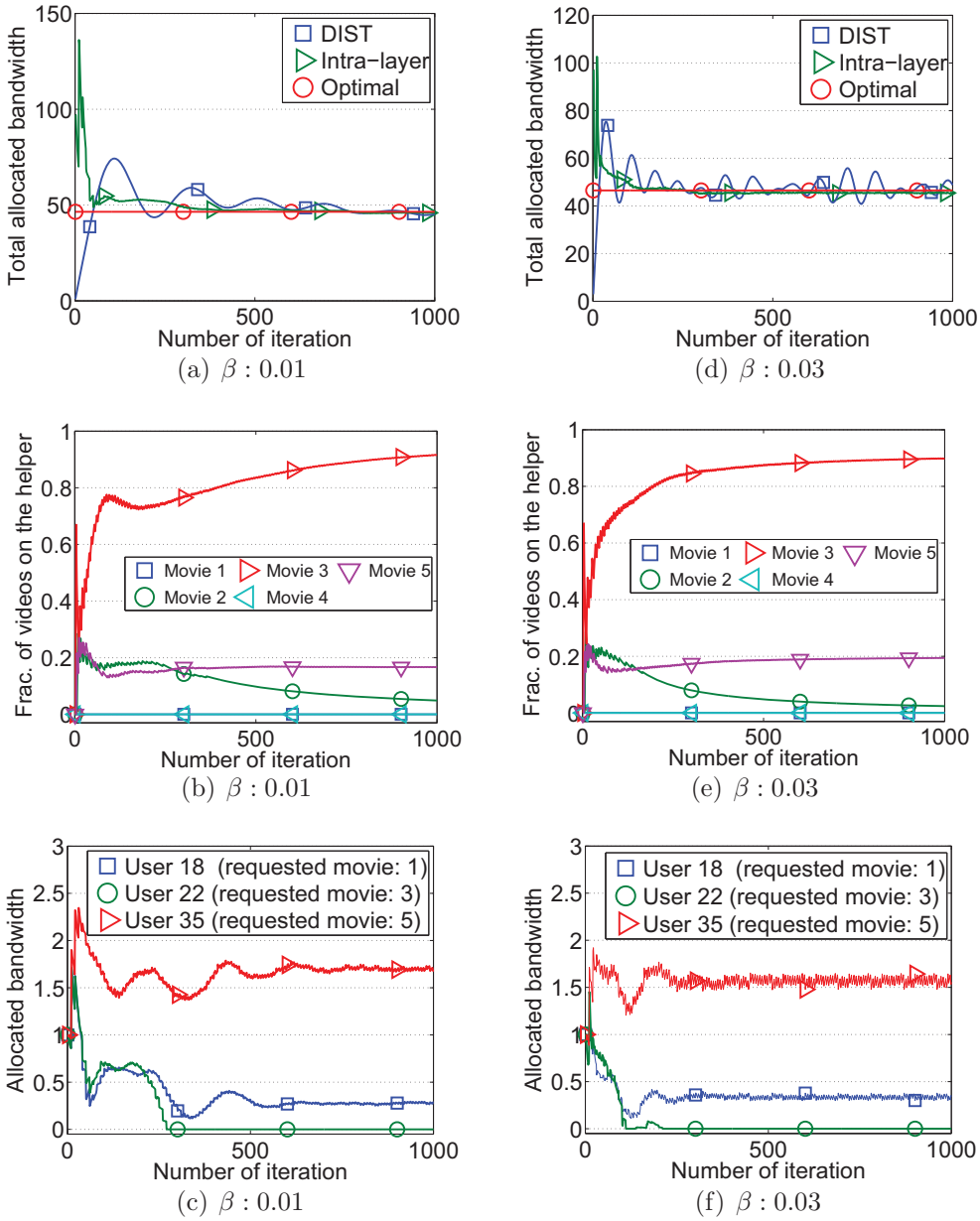


Figure 4.18: VoD with intra-layer coding. Convergence of the proposed distributed method to the optimal solution in a static network case. The number of users, helpers, and videos are equal to 50, 20, and 5, respectively. (a) and (d): Total allocated bandwidth to the users. (b) and (e): The fraction of each video on helper h_5. (c) and (f): The allocated bandwidth from helper h_5 to its adjacent users.

4.6.2 Convergence

In this section, we study the convergence of our distributed solution under both the static and dynamic cases.

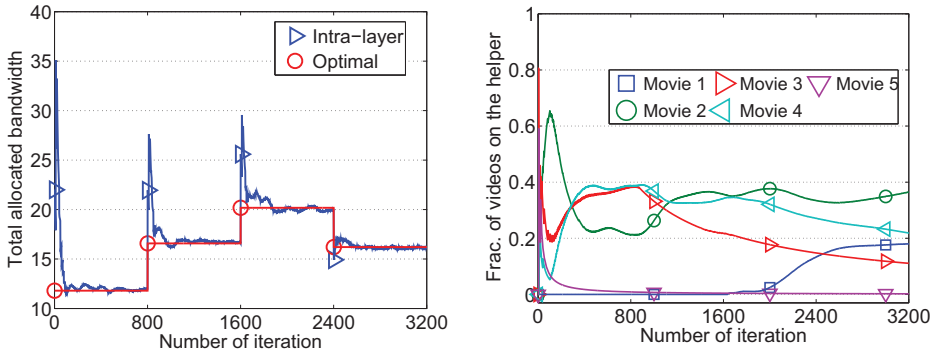
4.6.2.1 Static System

We evaluate the convergence of the proposed distributed algorithm in Figure 4.18. We solve the DIST method using a similar scheme to our solution in the case of inter-layer coding. However, in order to study the effect of Proximal method, we do not use the proximal method for DIST. In this figure, the number of users, helpers, and videos are equal to 50, 20, and 5, respectively. In order to have a fair comparison, we set the number of video layers to 1. The optimal solution is computed off-line for comparison. It is clear in Figure 4.18(a) that the proposed distributed solution converges to the optimal solution very fast; however, the convergence speed of the DIST approach is less than that of our approach. Moreover, the DIST method oscillates around the optimal solution. Figure 4.18(b) depicts the convergence of a particular helper's (helper h_5) storage allocation. The allocated storage for videos 2 and 4 goes to zero, since these videos are not requested by the adjacent users of this helper. The convergence of the allocated bandwidth from helper h_5 to its adjacent users is shown in Figure 4.18(c).

We repeat the previous experiment by increasing the step size β from 0.01 to 0.03. The results are shown in Figures 4.18(d), (e), and (f). By comparing Figures 4.18(a) and (d), it can be inferred that our distributed method converges faster to the optimal solution as we increase the step size. Moreover, even with a greater β, our method does not oscillate. On the other hand, the DIST method's oscillation increases rapidly as we increase the step size. Figures 4.18(e) and (f) illustrate the bandwidth and storage allocation of helper h_5, respectively.

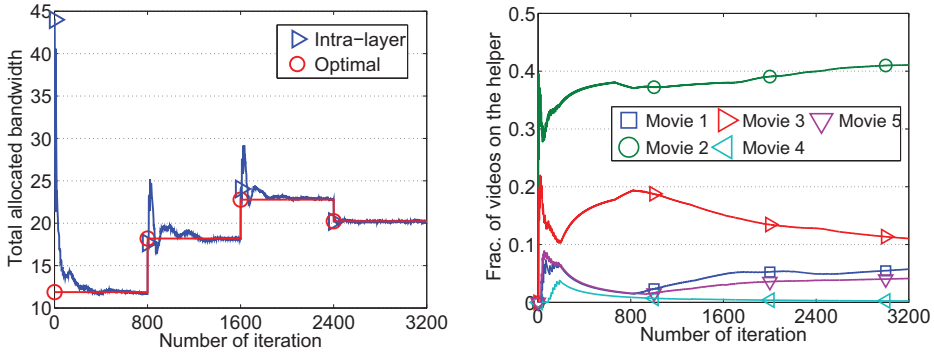
4.6.2.2 Dynamic System

Here, we show that our distributed approach automatically adapts to the system dynamics. As a result, the users and the helpers only need to run the distributed algorithm, regardless of the changes in the system.



(a) (b)

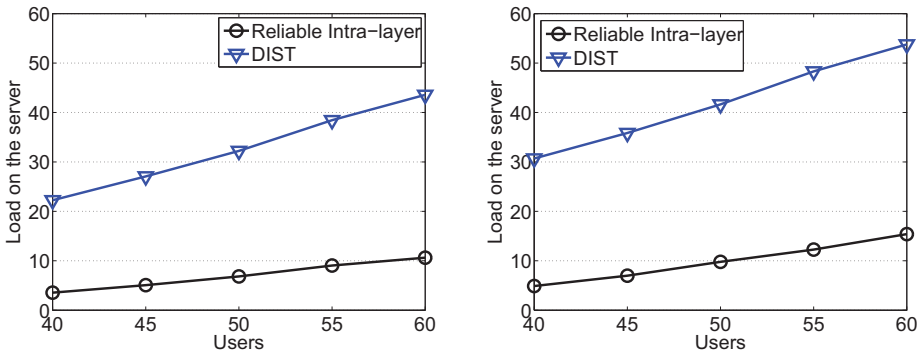
Figure 4.19: Convergence of the proposed distributed method to the optimal solution in the case of dynamic users, 10 helpers, 5 videos. (a) Total allocated bandwidth. (b) Frac. of videos on helper h_8.



(a) (b)

Figure 4.20: Convergence of the proposed distributed method to the optimal solution in the case of dynamic helpers, 20 users, 5 videos. (a) Total allocated bandwidth. (b) Frac. of videos on helper h_3.

We study the effect of changing the number of users to the system in Figure 4.19. For this purpose, we add 5 users at both iterations 800 and 1600, and we randomly connect them to [1,3] helpers. We also remove 5 users at iteration 2400. The initial number of users is 10, and there are 10 helpers in the system. We set the number of videos to 5. The optimal solution is computed off-line for comparison. Figure 4.19(a) shows that the total allocated bandwidth of the optimal solution changes as we add or remove users, and the distributed solution converges to the optimal result. We depict the fraction of stored videos on a helper h_8 in Figure 4.19(b).



(a) (b)

Figure 4.21: Server's load in kbps, in the case of unreliable links from helpers to the users and reliable server links, VoD, 20 helpers, 5 videos, 5 layers. (a) $\bar{\epsilon} \in [0.8, 1]$ (b): $\bar{\epsilon} \in [0.6, 0.8]$

We repeat the previous simulation for the case of dynamic helpers. We set the number of users, helpers, and videos to 20, 6, and 5, respectively. We add 3 new helpers at iterations 800 and 1600, and remove 3 helpers at iteration 2400. Figures 4.20(a) and (b) show that the proposed distributed method adapts to the changes in the dynamic case. After removing 3 helpers, the the allocated bandwidth does not return to the level of iteration 1600. The reason is that, the removed helpers are not those that are added at iteration 1600. We can conclude that the storage size or bandwidth of the removed helpers are less than those of the added helpers. Another possibility is that the added helpers covered the users with a common video request, but the removed helpers did not.

4.6.3 Unreliable links

Here, we repeat the experiment in Section 4.6.1 for the topologies with unreliable links between the helpers and the users. We set the reliability of these links in the range of $\bar{\epsilon} \in [0.8, 1]$, and measure the effect of number of users on the server load. The number of helpers, movies, and the video layers are equal to 20, 5, and 5 respectively. The server load of the methods increases in Figure 4.21(a) as we increase the number

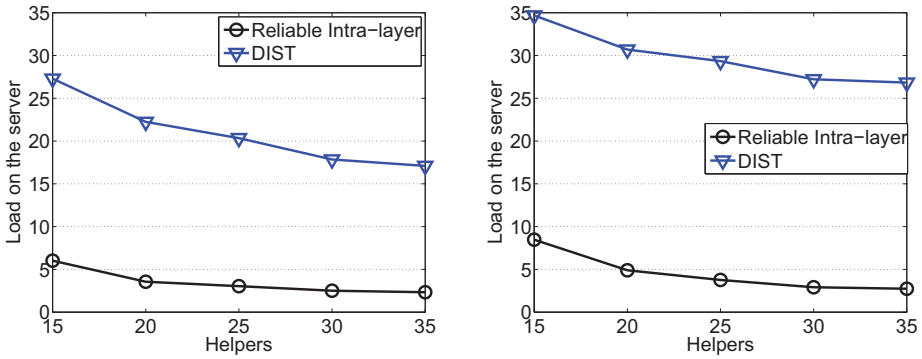
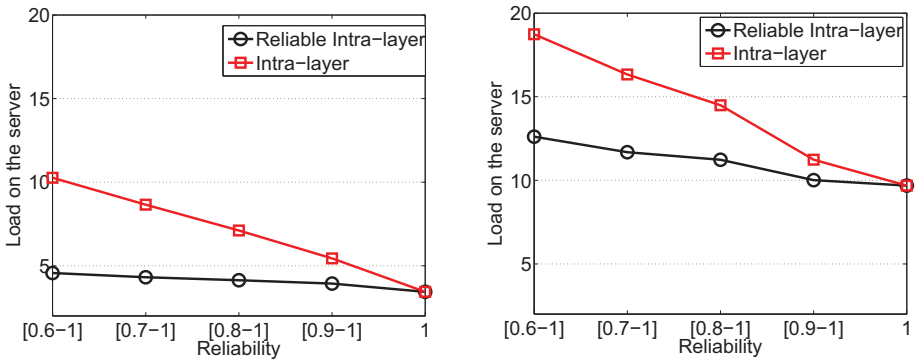


Figure 4.22: Server's load in kbps, in the case of unreliable links from helpers to the users and reliable server links, VoD, 40 users, 5 videos, 5 layers. (a) $\bar{\epsilon} \in [0.8, 1]$ (b): $\bar{\epsilon} \in [0.6, 0.8]$

of users, which is due to limited bandwidth and storage resources. By comparing Figures 4.21(a) and 4.12(a), we find that the gap between our proposed method and the DIST method is more in Figure 4.21(a). This is because the DIST method does not consider the unreliability of the links.

In Figure 4.21(b), we reduce the reliability of the links to the range of $\bar{\epsilon} \in [0.6, 0.8]$ and repeat the previous experiment. The two reasons that make the server load in the DIST method more than the reliable intra-layer coding are a.) the layered approach in our method, and b.) considering the reliability of the links. Having more users increases the requests for the resources, which results in a greater load on the server. The server load in Figure 4.21(b) is more that that of in Figure 4.21(a), which is due to less reliable helpers' links.

The number of users, videos, and layers in Figures 4.22(a) and (b) are equal to 40, 5, and 5, respectively. We set the range of links' reliability in Figure 4.22(a) to $\bar{\epsilon} \in [0.8, 1]$. Most similar to Figure 4.14(a), the load on the sever decreases as we increase the number of helpers. The reason is that, more helpers can provide a larger portion of the videos to the users. We change the reliability of the links from $\bar{\epsilon} \in [0.8, 1]$ to $\bar{\epsilon} \in [0.6, 0.8]$, and repeat the previous experiment. By comparing



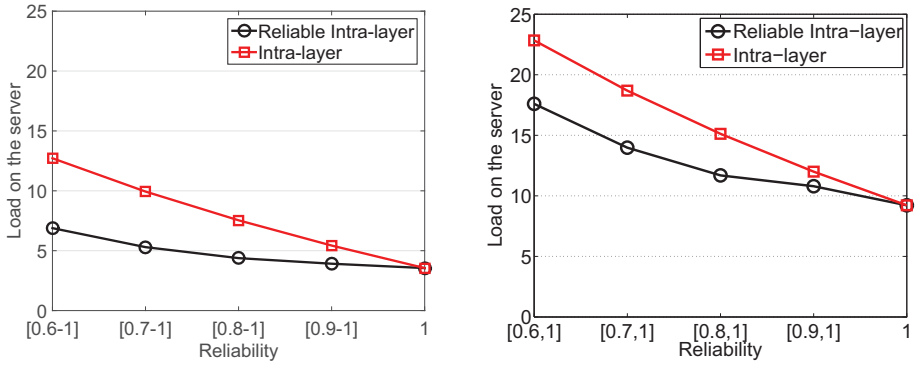
(a) Number of users: 40 (b) Number of users: 60

Figure 4.23: Server's load in the case of unreliable links from helpers to the users and reliable server links, VoD, 20 helpers, 5 videos, 5 layers.

Figures 4.22(a) and (b), we find that the unreliability of the links has more of a negative effect on the DIST approach, compared to the reliable intra-layer coding method.

Next, we compare the proposed intra-layer coding method with the reliable intra-layer coding. Figure 4.23(a) shows the load on the server for different ranges of link reliability. The number of helpers, videos, and layers are equal to 20, 5, and 5, respectively. Moreover, we set the number of users to 40. As expected, the proposed reliable intra-layer coding method reduces the load on the server, since it considers the unreliable links between the helpers and the users. This method gives more priority to the more reliable helper links, since giving the same priority to the links results in wasting the bandwidth resources. As the reliability of the links increases, the gap between the proposed method vanishes. In the case of reliable links, the server load of the methods becomes the same. We repeat the same experiment for 60 users in Figure 4.23(b), which results in more server load than that of in Figure 4.23(a).

We repeat the last two experiments in the case of unreliable links from the server and helpers to the users. Obviously, the server lossy links result in more of a load on the server, which can also be inferred from Figures 4.24(a) and (b). Remember that, in the case of lossy server links to the users, we change the objective function



(a) Number of users: 40 (b) Number of users: 60

Figure 4.24: Server's load in kbps, in the case of unreliable links from helpers and the server to the users, VoD. 20 helpers, 5 videos, 5 layers.

from maximizing the provided help through the helpers to minimizing the load on the server.

4.7 Conclusion

In this chapter, we study the problem of utilizing helpers to minimize the load on the central video servers. For this purpose, we formulate the problem as an LP optimization problem. This is done by using joint inter- and intra-layer NC. We discuss the advantages of joint inter- and intra-layer NC over just intra-layer NC, and through an empirical study, we found the cases in which joint coding reduces the server's load. We use a lightweight triangular inter-layer NC instead of the general form of inter-layer NC, to reduce the time complexity of the optimization. We solve the proposed optimization in a distributed way, and evaluate the convergence and the gain of our distributed approach via comprehensive simulations. Our future work is to consider the cost of helpers in the optimization and study the overhead that results from introducing the helpers.

CHAPTER 5

ROBUST WIRELESS TRANSMISSION OF SCALABLE CODED VIDEOS USING INTER-LAYER NETWORK CODING

In the first part of this dissertation, we studied the application of network coding in resource allocation. In chapters 3 and 4, we show that triangular network coding can result in using the limited network resources more efficiently. In the remaining chapters, we take a look at the applications of network coding in robust wireless transmissions. In this chapter, we study the application of network coding in making multi-layer videos robust against transmissions errors.

5.1 Setting

We consider a single-hop wireless network shown in Figure 5.1, in which a source node broadcasts a scalable video to a set of destinations over erasure wireless channels. We represent the i-th destination as d_i. The delivery rate from the source to d_i is represented as $\bar{\epsilon}_i$.

The transmitted video is coded using H.264/SVC (scalable video coding) video compression standard, which can encode a video into temporal, spatial and quality layers [53, 54]. In this chapter, we consider temporal and spatio (resolution) video layers, and assume that the encoded video contains m and n temporal and spatio layers, respectively. The temporal layers are correspondent to the time sequence frames of the videos. In contrast, the spatio video layers are correspondent to the

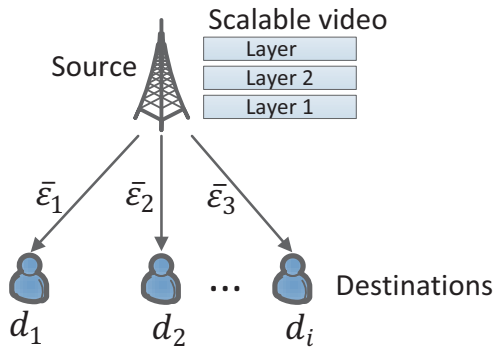


Figure 5.1: The system model.

resolution of the video frames. Receiving more spatio and temporal layers increases the quality of the decoded video (note that the video quality is different than the quality layers). However, a spatio layer is almost useless without having access to all of the spatio layers with a smaller index. The same characteristic exists between the temporal video layers.

Figure 5.2(a) shows the temporal layers of a video and the dependencies between them. Figure 5.2(b) depicts the same dependencies in a different way. In this figure, layers l_1 to l_4 represent the temporal layers. As Figure 5.2(c) shows, each temporal layer itself contains some spatio layers. The first spatio layer of each temporal layer is the base layer. The next spatio layers are enhancement layers, which can increase the quality of the video. Similar dependencies as those in Figure 5.2(a) and (b) exist between the enhancement spatio layers of different temporal layers, which are not shown for simplicity.

We represent the total number of transmissions for a GoP that the source node can perform as X. The distribution of these transmissions to the layers is shown as $(x_{1,1}, ..., x_{m,n})$. Moreover, the number of received packets for different layers is represented as $(y_{1,1}, ..., y_{m,n})$. We assume that layer $l_{i,j}$ contains $r_{i,j}$ packets.

As the different users (destinations) have different wireless channel conditions (delivery rates), they receive different numbers of layers. Consequently, the quality

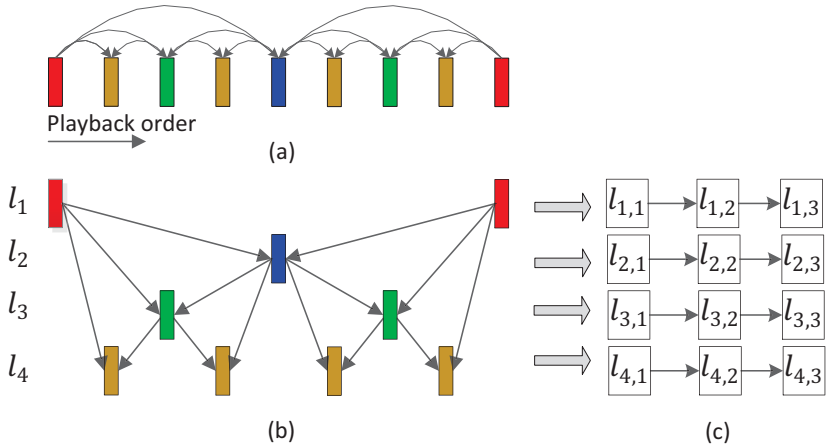


Figure 5.2: Scalable video coding with hierarchical B pictures. (a) and (b): Temporal layers. (c) Spatio layers.

Table 5.1: The set of symbols used in this chapter.

Notation	Definition
d_i	The i-th destination node.
$\bar{\epsilon}_i$	The delivery rate of the link between the source and the i-th destination node.
l_i	The i-th temporal video layer.
$l_{i,j}$	The j-th spatio layer of the i-th temporal layer.
$L_{i,j}$	The triangular coded layers over $l_{1,1}$ to $l_{i,j}$.
$r_{i,j}$	Number of packets in layer $l_{i,j}$.
X	Total number of transmissions for a GoP.
$x_{i,j}$	The number of transmitted triangular coded packets over layer $L_{i,j}$.
$y_{i,j}$	The number of received triangular coded packets over layer $L_{i,j}$.
m	The number of temporal layers.
n	The number of spatio layers.
α	Random coefficient for the linear coded packets.

of the decoded videos at the destination nodes are different. These qualities not only depend on the number of received layers, but also on which layers have been received. Our objective in this work is to maximize the total quality of the received video by the destination nodes. In our simulations, we use both the number of decoded layers and the PSNR of the decoded videos to evaluate our methods. The set of symbols used in this chapter is summarized in Table 5.1.

5.2 Robust Network Coding Schemes

Our video transmission method has two phases. In the first phase, we decide about the NC scheme that should be used to code the video layers. In the second phase, the total number of transmissions are distributed among the possible network coded packets of the selected NC scheme. Before discussing about the NC schemes on video layers, we provide some preliminary on NC.

5.2.1 Background on Network Coding

In this chapter, random linear network coding (RLNC) is used to code the video packets. In RLNC, coded packets are random linear combinations of the original packets over a finite field of size q. Each coded packet is in the form of $\sum_{i=1}^{k} \alpha_i \times p_i$. Here, α_i and p_i are random coefficients and the packets, respectively. The coefficients are selected with a uniform distribution over a finite field. Also, k is the number of packets that are coded together. Instead of transmitting the original packets, the source node generates and transmits random coded packets over the k packets. In order to decode the coded packets and retrieve the original packets, the destination nodes need to receive k linearly independent coded packets. They can use Gaussian elimination to decode the coded packets.

Using the general form of RLNC for layered videos has a problem. Consider 3 single-packet video layers l_1, l_2, and l_3, which contain packets p_1, p_2, and p_3, respectively. If we perform RLNC on these layers, we should combine these 3 packets together, and the coded packets will be in form of $\alpha_1 p_1 + \alpha_2 p_2 + \alpha_3 p_3$. Using RLNC, two cases can happen. If a destination node receives 3 linearly independent coded packets, it will be able to decode all of the 3 layers. In contrast, in the case of receiving less than 3 linearly independent coded packets, the destination cannot decode and retrieve any packet. As a result, none of the layers will be decodable. It means that using RLNC, there is no way to receive a subset of the layers. The probability of

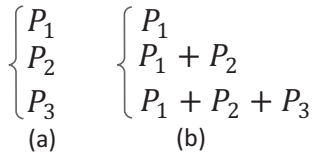


Figure 5.3: (a) The 3 original packets. (b) The 3 triangular coded packets. The coefficients are not shown for simplicity.

receiving a certain number of linearly independent coded packet is impacted by the value of q.

In order to solve this problem, triangular network coding is proposed. In triangular coding, the i-th coded layer is a linear combination of the packets in the first i layers. The triangular coded packets of the previous example are shown in Figure 5.3. The coefficients are not shown for simplicity. Using triangular coding, if a destination node receives 2 linearly independent coded packets over packets p_1 and p_2, it can decoded the packets and retrieve layers l_1 and l_2. Moreover, it is possible to receive layer l_1 alone.

The triangular NC can be easily generalized to the case of multiple packets per layer. Assume that layer l_i contains r_i packets. In order to code layers l_1 to l_i, we code all of the packets in these layers. For example, the coded packets over layer l_1 are random linear combination of the packets in l_1. As a result, in order to retrieve layer l_1, r_1 coded packets over this layer are required. Moreover, in order to code layers l_1 and l_2, we code the packets of these 2 layers together.

5.2.2 Network Coding Schemes for Video Transmission

Without loss of generality, we describe the NC schemes in the case of single packet per layer. In Figure 5.2, we have 4 temporal and 3 spatio layers. We can reshape the video layers structure in Figure 5.2 and simplify the layer dependencies to Figure 5.4(a). Each row and column in Figure 5.4(a) represents a spatio and a temporal layer, respectively. Also, the figure shows the dependencies between the layers. For

example, layer $l_{2,2}$ is dependent on layer $l_{1,2}$ and $l_{2,1}$ directly. Moreover, layer $l_{2,2}$ is indirectly dependent on layer $l_{1,1}$. It can be inferred from Figure 5.4(a) that layer $l_{a,b}$ is not useful unless we have access to layers $l_{i,j}$: $\forall 1 \leq i \leq a, 1 \leq j \leq b$. In the case of single-layer videos, there is only one triangular coding scheme, as described in the previous section. However, for two-dimensional layered videos, there are several ways to choose which layers to code together. In the following sections we discuss the possible coding approaches.

5.2.2.1 Canonical Triangular Coding

The work in [57], performs triangular NC on the quality layers. In canonical triangular coding, the k-th coded layer is a linear coded layer over the first k quality layers of all of the temporal layers. In other words, the coded layer k is in the form of $\sum_{i=1}^{m} \sum_{j=1}^{k} \alpha \times l_{i,j}$. In this chapter, we consider spatio layers instead of the quality layers. As a result, if we apply this method in our setting, we should replace quality layers with spatio layers.

5.2.2.2 Vertical Triangular Coding

In the case that we want to give more priority to the spatio layers, we can first perform the triangular coding scheme on the spatio layers of the first temporal layer, as shown in Figures 5.4(a)-(c). In this method, the first coded layer contains only layer $l_{1,1}$. The second coded layer is coded over layers $l_{1,1}$ and $l_{1,2}$. In general, the first n coded layers are in the form of $\sum_{j=1}^{k} \alpha \times l_{1,j},\ \forall k : 1 \leq k \leq n$.

Figures 5.4(a)-(c) show the first 3 coded layers using the vertical coding scheme. As depicted in Figure 5.4(d), after coding the spatio layers of the first temporal layer, we perform triangular coding on the temporal layers. The next coded layers contains layers $l_{1,1}$ to $l_{3,3}$ (Figure 5.4(e)), and the last coded layer contains all of the layers as shown in Figure 5.4(f).

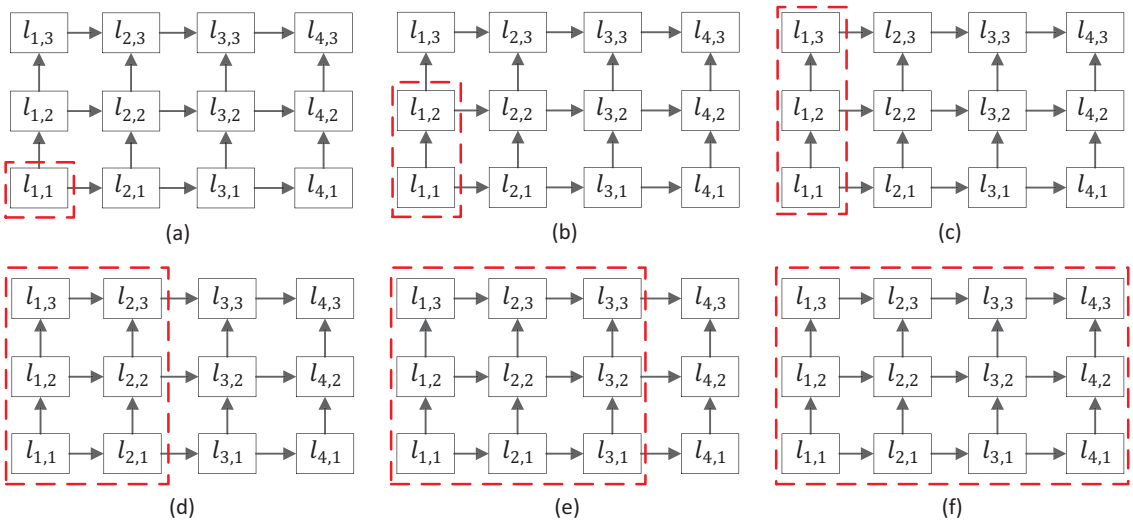


Figure 5.4: Vertical Triangular Network coding scheme.

5.2.2.3 Horizontal Triangular Coding

This scheme is the reverse of the vertical triangular coding scheme. We first apply triangular coding on the temporal layers of the first spatio layer. Then, for the second spatio layer, we code all of the temporal layers of the first and second spatio layer together. This process is repeated for the other spatio layers. In contrast with the vertical coding scheme, horizontal coding gives more priority to the temporal layers than the spatio layers. The set of possible codings in the case of 4 temporal and 3 spatio layers is shown in Figure 5.5.

5.2.2.4 Diagonal Triangular Coding

This scheme gives the same priority to the temporal and spatio layers. Figures 5.6(a)-(d) show our diagonal triangular coding scheme. The first layer is $l_{1,1}$, which is not coded with the other layers. The second layer contains layers $l_{1,1}$, $l_{1,2}$, $l_{2,1}$, and $l_{2,2}$. In the same way, the third coded layer is coded over layers $l_{i,j} : 1 \leq i, j \leq 3$. If n and m are not equal, the shape of the layers becomes non-square as depicted in Figure 5.6(d). In this case, after reaching the last spatio or temporal layer, depending

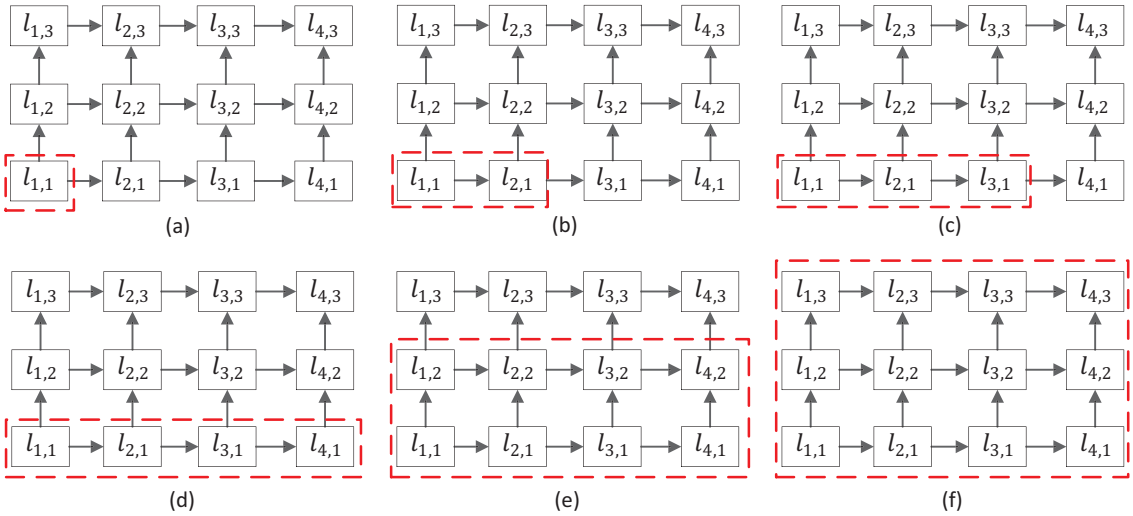


Figure 5.5: Horizontal Triangular Network coding scheme.

on whether n or m is smaller, we just increase the index of the other dimension in the next coded packets. Assuming that n is smaller than m, the general form of the coded packets is:

$$\sum_{i,j\in[1,k]} \alpha_{i,j} l_{i,j},\ \forall\ 1 \leq k \leq n$$

$$\sum_{i\in[1,k],j\in[1,n]} \alpha_{i,j} l_{i,j},\ \forall\ n < k \leq m$$

5.2.2.5 General Two-Dimensional Triangular Coding

The coded layers in the general triangular coding are the union of the above schemes. In this scheme, any layer $l_{i,j}$ is coded with the layers $l_{1,1}$ to $l_{i,j}$. For example, as Figure 5.4(b) shows, layer $l_{1,2}$ is coded with layer $l_{1,1}$. Moreover, in Figure 5.6(b), layers $l_{1,1}$ to $l_{2,2}$ are coded together. If we want to code layer $l_{3,3}$, we code all of the layers with i and j in the range of 1 and 3. The idea behind this coding scheme is that because of the dependencies between the layers, layer $l_{i,j}$ is not useful unless the layers with smaller i and j indices are available to the user. In this coding scheme,

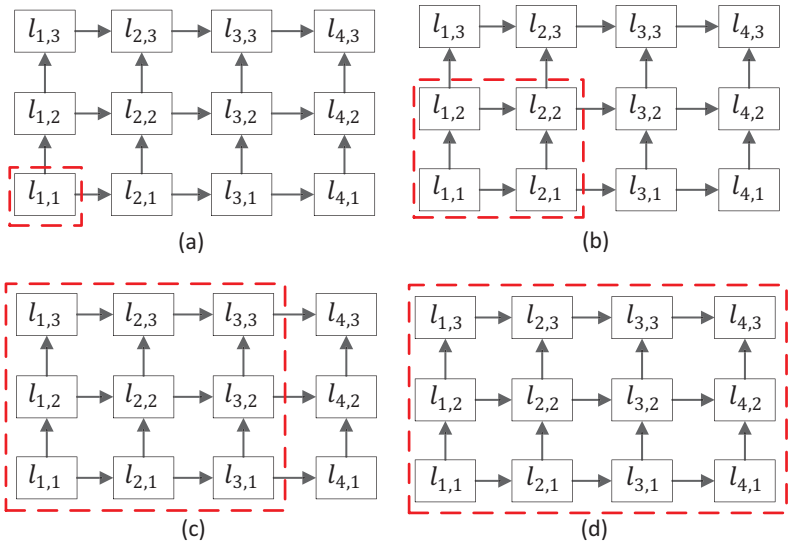


Figure 5.6: Diagonal Triangular Network coding scheme.

we have mn possible coded layers.

It is shown in [57] that finding the optimal distribution of the transmissions to the different layers is a difficult problem, even in the case of a one-dimensional scalable video. Therefore, it is not a straightforward task to find the optimal distribution in the general triangular coding. Due to too many coding possibilities in the case of general two-dimensional coding (specifically mn), the time complexity of finding the optimal distribution of the transmissions is high. As a result, we avoid this type of coding in our work.

5.2.2.6 Coding Speed

In order to make the diagonal coding scheme more general, and at the same time keep the complexity of the method low, we introduce the concept of *coding speed* (CS). Coding speed is a triple $CS = (z_1, z_2, z_3)$. Here, z_1 shows the number of spatio layers that should be added to the coded layer before a temporal layer be added. In the same way, z_2 represents the number of temporal layers that should be added to the coded layer before a spatio layer be added. Also, z_3 shows whether the coding

direction is first horizontal or vertical. In more details, $s_3 = 0$ and $s_3 = 1$ mean that the coding direction is first vertical and horizontal, respectively. The possible codings with coding speed $CS = (1, 2, 1)$ are shown in Figure 5.7. Here, $z_3 = 1$ means that the coding starts in the horizontal direction. Also, $z_1 = 1$ and $z_2 = 2$ mean that, after adding 2 temporal layers (horizontal), 1 spatio layer (vertical) will be added to the coding set. This process will be repeated until we reach the last temporal and spatio layers.

As shown in Figure 5.7(a), the first coded layer only includes layer $l_{1,1}$. We need to first add 2 temporal layers to the coding set, and then add one spatio layer to the coding set. As shown in Figure 5.7(b) and (c), layers $l_{2,1}$ and $l_{3,1}$ are added to the coding set. In Figure 5.7(d), we add layer $l_{3,2}$ to the coding set. In order to add layer $l_{3,2}$, we also need to include layers $l_{1,2}$ and $l_{2,2}$. We perform the coding as an atomic procedure, which means, in each step, we just increase the height or length of the coding set.

We refer to this scheme as speed triangular coding (STC). The horizontal and vertical triangular coding schemes are subsets of the STC method with a coding speed equal to $CS = (m - 1, n - 1, 1)$ and $CS = (m - 1, n - 1, 0)$, respectively. The optimal CS depends on the video coding scheme and the rates of the layers. For example, in the cases that the temporal layers have more of an impact on the video quality than the spatio layers, e.g. videos that the similarity between their frames is low, more speed can enhance the quality of the video played by the users. In contrast, if the spatio layers are more important than the temporal layers, a speed less than 1 might be better.

5.2.3 Comparing the Coding Schemes

It is clear that receiving more layers increases the video watching experience. In other words, receiving more layers reduces video distortion. However, if we want to

have a comparison between the temporal and spatio layers, we can consider receiving more temporal layers as a smoother playback of the videos, and more spatio layers as frames with a higher resolution. Therefore, the horizontal triangular coding provides a smooth playback, and the vertical triangular coding gives more priority to the resolution of the frames. Moreover, the diagonal triangular coding gives the same importance to the resolution and the smoothness of the videos. Because of time complexity necessary to find the optimal solution using the general form of two-dimensional network coding is high, we do not use it in our work.

Referring to the above discussion, we can set up coding speed by performing a trade-off between playback smoothness and resolution of the video, or by considering the overall distortion. Coding speeds with $z_3 = 1$ and a large z_2 gives more priority to the smoothness. Also, coding speed with $z_3 = 0$ and large z_1 gives more importance to the resolution. In the extreme cases, the coding becomes similar to the vertical and horizontal triangular coding. Moreover, in the case of a coding speed equal to $CS = (1, 1, 1)$ or $CS = (1, 1, 0)$, the importance of the resolution and smoothness are the same. From an overall distortion perspective, we should try to minimize the distortion of the video.

5.3 Optimal Transmissions Distribution

After deciding on the coding scheme, we need to distribute the transmissions among the coded layers of that coding scheme. Consider a video with 4 temporal and 3 spatio layers that is coded using a horizontal coding scheme. Using the horizontal coding scheme, we will have 6 coded layers $L_{1,1}$ to $L_{4,1}$, $L_{4,2}$, and $L_{4,3}$. Considering the limited time and bandwidth, the question is that how many times the source node needs to transmit the packets of each coded layer. Similar to the work in [57], in order to find the optimal solution, we check all of the possible distributions and select the distribution that maximizes the total gain. With the purpose of reducing

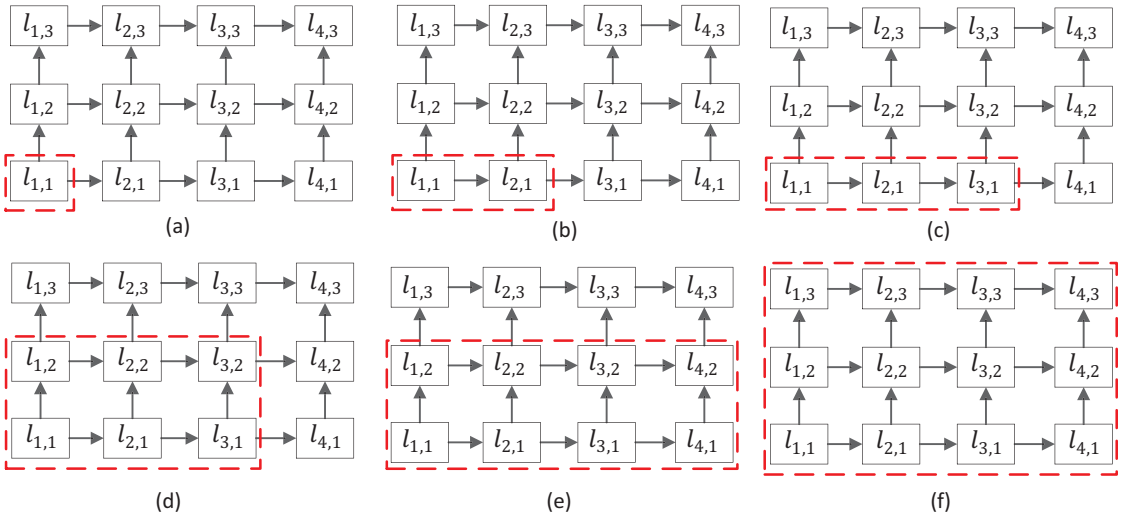


Figure 5.7: Coding speed illustration, $CS = (1, 2, 1)$.

the time complexity of checking all of the possible distributions, we can assign the transmissions to different layers with granularity g.

In our method, we generate a reference (look-up) table which shows the decoding probability of each layer for each possible distribution of the transmissions and a delivery rate. It should be noted that the reference table needs to be generated just once, and the source node can use it to find the optimal distribution in any delivery rate scenario. Having the reference table, the source node can easily search the reference table to find the best distribution scheme in the case of multiple destinations with different delivery rates. In the following sections, we first propose an algorithm for creating a reference table in the case of single packet per layer. We then extend the algorithm to the case of multiple packets per layer.

5.3.1 Avoiding Gaussian Elimination

We denote a transmission distribution as $(x_{1,1}, ..., x_{m,n})$, and the number of received transmissions by a destination node as $(y_{1,1}, ..., y_{m,n})$. In order to check the probability of decoding each layer for a given transmission distributions, we need to

check which layers can be decoded for each possible outcome (transmission receptions). For example, assuming that the transmissions of 2 coded layers $L_{1,1}$ and $L_{2,1}$ is equal to $x = (1, 1)$, the possible outcomes are $y = (0, 0)$, $y = (1, 0)$, $y = (0, 1)$, and $y = (1, 1)$. One approach to check the layers that can be decoded is using Gaussian elimination. In order to reduce the time complexity of reference table creation, we want to avoid Gaussian elimination. In [57], a remark is proposed to calculate the number of decodable layers for a given reception outcome, without Gaussian elimination for the case of one-dimensional layered-videos. We first prove the remark, as it gives insight for a remark in the case of two-dimensional triangular coding and its proof. Here, we assume a large field size, e.g. $q \geq 2^8$. As a result, as shown in [74], with a high probability, the generated random linear network coded layers are linearly independent. For a small field size q, the coded layers might not be linearly independent.

Remark 5.1. Under a one-dimensional triangular coding and a reception outcome $(y_1, ..., y_m)$, the user can decode layers up to l_a if and only if:

$$\sum_{i=h}^{a} y_i \geq a - h + 1, \ \forall \ h : 1 \leq h \leq a \tag{5.1}$$

Proof. Assume that we have a coded packets over layers l_1 to l_a with a rank equal to a, which means that layers l_1 to l_a can be decoded. Also, assume that for a given h, we have $\sum_{i=h}^{a} y_i < a - h + 1$. As a result, the number of coded packets over layers l_1 to l_{h-1} is equal to $\sum_{i=1}^{h-1} y_i > a - (a - h + 1) = h - 1$. Consequently, we have at least h coded packets over the first $h - 1$ layers. It means that these coded packet cannot be linearly independent, which contradicts the assumption. □

For example, the client can decode layers l_4 and its predecessor layers l_1, l_2, and l_3, if and only if $y_4 \geq 1$, $y_4 + y_3 \geq 2$, $y_4 + y_3 + y_2 \geq 3$, and $y_4 + y_3 + y_2 + y_1 \geq 4$.

The remark 5.1 gives us an insight for the decodability conditions in the case of

two-dimensional coding:

Remark 5.2. Under any two-dimensional triangular coding scheme, for a reception outcome $(y_{1,1}, ..., y_{m,n})$, the user can decode layers up to $l_{a,b}$ if and only if:

$$\sum_{i=h}^{a}\sum_{j=k}^{b} y_{i,j} \geq (a-h+1)(b-k+1) \tag{5.2}$$
$$\forall\ h,k : 1 \leq h \leq a, 1 \leq k \leq b$$

Proof. Assume that we have ab coded packets over layers $l_{1,1}$ to $l_{a,b}$ with a rank equal to ab. In other words, layers $l_{1,1}$ to $l_{a,b}$ are decodable. Moreover, assume that h and k are the largest indices such that:

$$\sum_{i=h}^{a}\sum_{j=k}^{b} y_{i,j} < (a-h+1)(b-k+1)$$

As a result, the number of coded packets over the layers with smaller indices that h or k is equal to:

$$\sum_{i=1}^{h-1}\sum_{j=1}^{b} + \sum_{i=1}^{a}\sum_{j=1}^{k-1} > ab - (a-h+1)(b-k+1) \tag{5.3}$$
$$= ak - a + hb - hk + i - b + k - 1$$
$$= a(k-1) + b(h-1) - (h-1)(k-1)$$

On the other hand, the number of packets involved in these coded layers is equal to $a(k-1)+b(h-1)-(h-1)(k-1)$. As a result, the number of coded packets on these original layers is greater than the number of original packets. It means that these coded packets cannot be linearly independent, which contradicts the assumption. □

For example, the client can decode layers $l_{2,2}$ and its predecessor layers $l_{1,1}$, $l_{1,2}$, and $l_{2,1}$, if and only if $y_{2,2} \geq 1$, $y_{2,2}+y_{1,2} \geq 2$, $y_{2,2}+y_{2,1} \geq 2$, and $y_{2,2} + y_{1,2} + y_{2,1} + y_{1,1} \geq 4$.

Algorithm 6 Distribution Algorithm for Diagonal Coding

```
1: // dist(x, i, j, X)
2: if i ≤ m or j ≤ n then
3:    for k = 1 to X do
4:       x_{i,j} = k
5:       if i ≤ m and j ≤ n then
6:          dist(i + 1, j + 1, X − k)
7:       else if i ≤ m then
8:          dist(i + 1, j, X − k)
9:       else
10:         dist(i, j + 1, X − k)
11: else
12:    RTC(1, 1, x)
```

5.3.2 Reference Table Creation

In order to create the reference table, we first produce all the possible distributions of the X transmissions given a specific coding scheme. The distribution algorithm for diagonal coding is shown in Algorithm 6. In order to distribute X transmissions among the layer, we should call the algorithm with parameters X, $i = 1$, and $j = 1$. The algorithm recursively calls itself until i and j reach m and n, respectively. During each run of the algorithm, the remaining transmissions are assigned to each valid triangular coding. After producing a possible distribution, the reference creation algorithm (RTC) is called to calculate the successful decoding probability of each layer and to fill the reference table.

In order to find the successful decoding probability of a layer, we need to check if a given layer is decodable in the different possible delivery outcomes. The RTC algorithm receives a given distribution x and creates the possible delivery outcomes. It first sets the decodability of each layer to 1. It then checks Remark 5.2 for each layer to find which layers are not decodable under each delivery outcome. Then, the RTC algorithm calculates the probability of that specific outcome happening, and add this probability to the cell correspondent to the decodable layer with the largest indices. The algorithm calculates this probability in terms of different packet delivery

$$\begin{bmatrix} q_{1,1,0.95} & q_{1,2,0.95} & \cdots & q_{1,n,0.95} \\ q_{2,1,0.95} & q_{2,2,0.95} & \cdots & q_{2,n,0.95} \\ \vdots & \vdots & \ddots & \vdots \\ q_{m,1,0.95} & q_{m,2,0.95} & \cdots & q_{m,n,0.95} \end{bmatrix}$$

Figure 5.8: Example of reference table.(a) Delivery rate $\bar{\epsilon} = 0.95$. (b) Delivery rate $\bar{\epsilon} = 0.90$. (c) Delivery rate $\bar{\epsilon} = 0.85$.

Algorithm 7 Reference Table Creation (RTC)

```
1: for each outcome y do
2:   for i = 1 to m do
3:     for j = 1 to n do
4:       dec(i, j) = 1
5:   for a = 1 to m do
6:     for b = 1 to n do
7:       for h = 1 to a, k = 1 to b do
8:         if sum_{i=h}^{a} sum_{j=k}^{b} y_{i,j} < (a − h + 1)(b − k + 1) then
9:           dec(a, b) = 0
10:  for i = 1 to m do
11:    for i = 1 to n do
12:      if dec(i, j) = 1 then
13:        for ε̄ = 0.05 to 1, step = 0.05 do
14:          q = prob(receiving y out of x) transmissions
15:          RT(i, j, p) = q
```

rates. In this chapter, we consider 0.05 as the granularity of the packet delivery rates. The details of the RTC algorithm are shown in Algorithm 7.

Figure 5.8 shows a part of a three-dimensional reference table, which is created for a given transmission strategy x. The figure depicts the successful decoding probability of each layer in the case of a delivery rate equal to 0.95. In this figure, we represent the successful decoding probability of layer $L_{i,j}$ for delivery rate $\bar{\epsilon}$ as $q_{i,j,p}$. For each delivery rate, there will be a matrix similar to that in Figure 5.8.

5.3.3 Extension to Varying Number of Packets per Layer

In the previous sections and the proposed algorithm, we assumed that each layer contains a single packet. However, the different layers of the videos might be encoded at different bitrates, and as a result, contain different number of packets. Remark 5.2 and the proposed algorithm can be easily extended to the case of multiple packets per layer. We just need to change the decoding condition in Remark 5.2 and use it in the algorithm. Considering $r_{i,j}$ packets in layer $l_{i,j}$, the decoding condition in Remark 5.2 becomes:

$$\sum_{i=h}^{a}\sum_{j=k}^{b} y_{i,j} \geq \sum_{i=h}^{a}\sum_{j=k}^{b} r_{i,j} \tag{5.4}$$

$$\forall\ h,k : 1 \leq h \leq a, 1 \leq k \leq b$$

5.4 Evaluation

In this section we evaluate our proposed methods by comparing them with the Percy method [57]. In Percy, the triangular coding is performed among the spatio layers (in [57], quality layers are mentioned; however, Percy can be applied on spatio layers instead on the quality layers). Therefore, the first coded layer contains layers $l_{1,1}$ to $l_{m,1}$. The next coded layers are similar to those of the horizontal coding. Different from their work, we consider spatio and temporal layers. In order to evaluate the methods, we implemented a simulator in the MATLAB environment. Moreover, we use JSVM reference software [75] for encoding and decoding videos and measuring the PSNR of the decoded videos.

5.4.1 Simulation Setting

We use the Bus [76] and Akiyo [77] video sequences in our evaluations, which are shown in Figure 5.9. The resolution and the frame rate of the videos that we use are

(a) Bus sequence, base layer

(b) Bus sequence, three layers

(c) Akiyo sequence, base layer

(d) Akiyo sequence, three layers

Figure 5.9: Decoded video frames from Bus and Akiyo video sequences. (a) and (c) Decoded video frames using the base spatio layer. (b) and (d) Decoded video frames using all of the 3 spatio layers.

equal to 352×288 pixels and 30 frames per second, respectively. We partition the videos to 4 and 3 temporal and spatio layers. The resolution of the spatio layers 1, 2, and 3 are equal to 176×144, 320×240, and 352×288, respectively. Figures 5.9(a) and (b) depict the decoded videos frames using the base layer and the 3 spatio layers of the Bus video sequence.

We evaluate the methods in terms of number of decodable layers and PSNR. We calculate the PSNR of the decoded videos using the PSNRStatic function provided in JSVM. We use the decoded video using all of the layers as the reference to calculate the PSNR. Before we calculate the PSNR of a decoded video, we upsample the decoded video to make its frame rate and resolution consistent with that of the reference video. The PSNR of the decoded videos for different layers of the Bus and Akiyo video sequence are shown in Tables 5.2 and 5.3, respectively.

Table 5.2: PSNR of the decoded layers of the Bus video sequence.

n \ m	1	2	3	4
1	31.24	32.85	34.30	35.62
2	31.72	34.12	36.97	40.6
3	39.51	49.4	67.11	99

Table 5.3: PSNR of the decoded layers of the Akiyo video sequence.

n \ m	1	2	3	4
1	35.58	35.96	36.17	38.74
2	37.83	38.78	39.49	45.24
3	39.24	40.85	42.38	99.99

Figure 5.10: Number of decodable layers in the case of single user. $m = 4$, $n = 3$, $X = 12$, $CS = 1$.

In the simulations, we set $CS = (1, 1, 1)$ for the speed coding method. We run the simulations for 1000 random delivery rates, and show the average output in the plots.

5.4.2 Simulation Results

5.4.2.1 Number of Decodable Layers

In the first experiment, we measure the number of decodable layers in the case of single user in Figure 5.10. We assume single packet per layer, and set the total number of transmissions to 12. As expected, the figure shows that the number of decodable layers increases as the delivery rate of the link increases. Moreover, the number of decodable layers of the horizontal coding is always more than that of the Percy method. The reason is that the coding in the Percy method is a subset of the coding possibilities in the horizontal approach. However, for some of delivery rates, the Percy method delivers more layers that that of the vertical approach. Figure 5.10 shows that the number of decodable layers in the horizontal and vertical methods are up to 12% and 33% more than that of the Percy method, respectively. In this plot, the total number of transmissions is equal to the number of packets in the video. The

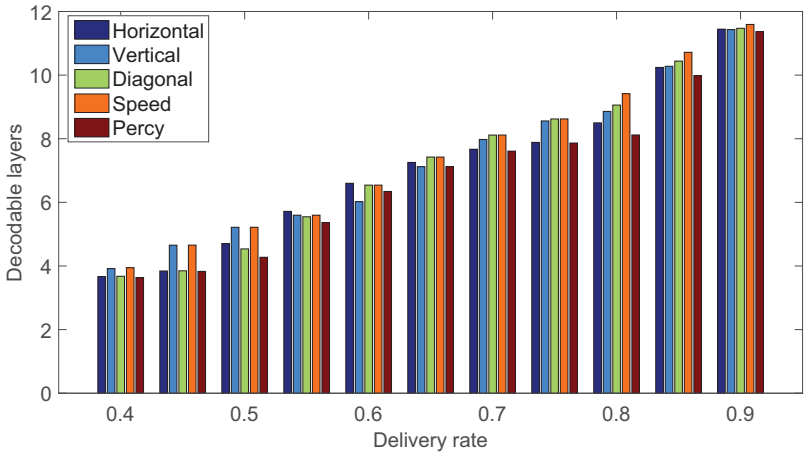


Figure 5.11: Number of decodable layers in the case of single user. $m = 4$, $n = 3$, $X = 15$, $CS = 1$.

idea behind selecting this low number of transmissions is to measure the quality of the received videos in the case of limited bandwidth.

We repeat the previous experiment in the case of 15 packet transmissions. The result is depicted in Figure 5.11. The figure shows that in the case of a delivery rate equal to 0.9, the number of received videos in different methods becomes very close to each other. The reason is the high delivery rate and high redundancy level. However, for a moderate delivery rate, the number of received layers using our methods is still more than that of the Percy method.

Figure 5.12(a) shows the average number of decodable layers for different numbers of users. The total number of transmissions in this experiment is set to 12, and the reliability of the links are randomly chosen in the range of $[0.4, 0.9]$. In other words, the reliability is distributed uniformly in the specified range. As the figure illustrates, the average number of decodable layers by the users decreases as we increase the number of users. The reason is that, as we increase the number of users, the diversity of the channels' delivery rates increases. In this case, it is probable that a good distribution choice for a user is not appropriate for another user. Therefore, we cannot satisfy all of the users at the same time. It can be inferred from the figure

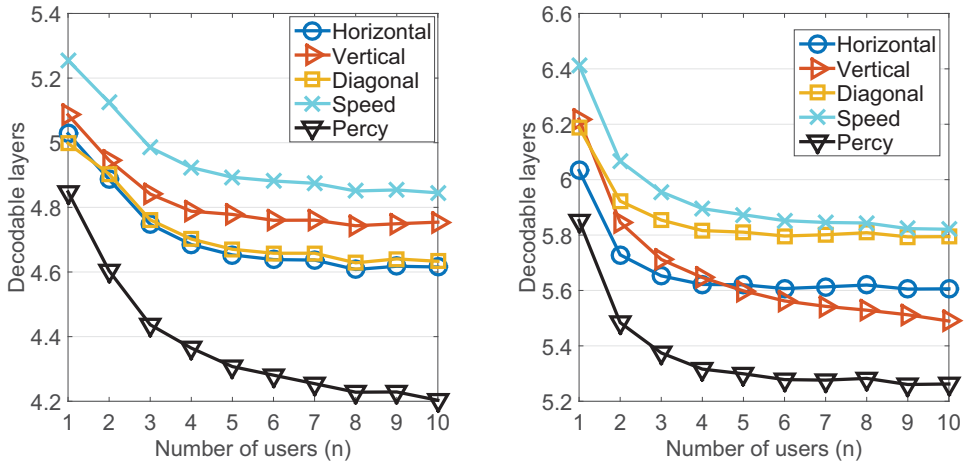


Figure 5.12: Number of decodable layers in the case of multiple users. $m = 4$, $n = 3$. (a) $X = 12$. (b) $X = 14$.

that the number of decodable layers in the speed coding method is up to 14% more than that of the Percy method.

We increase the total number of transmissions to 14 and repeat the previous experiment. The result is shown in Figure 5.12(b). It is clear that a higher number of transmissions results in more decodable layers. The figure illustrates that the speed coding scheme and the Percy method has the highest and lowest number of decodable layers, respectively.

5.4.2.2 PSNR

In the previous experiments, we just checked the number of decodable layers. However, the relation of number of decodable layers and the quality of the video is not always linear. Therefore, we also measure the PSNR of the decoded videos. In Figure 5.13, the PSNR of the decoded videos in the case of single user are shown for different delivery rates. The Bus video sequence contains 18 packets, and the number of transmitted packets is equal to 20. The figure shows that the PSNR of the decoded video increases as the delivery rate of the link increases. Figure 5.13 depicts that the

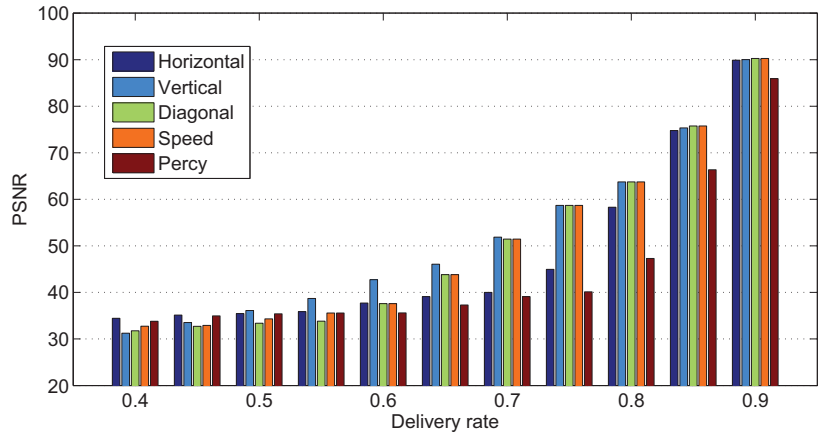


Figure 5.13: Bus video sequence. SNR of the decoded video in the case of single user. $m = 4$, $n = 3$, $X = 20$, $CS = 1$.

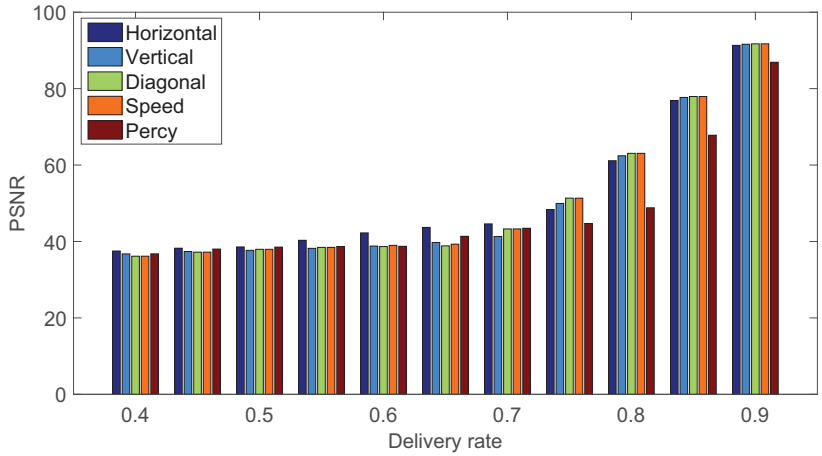


Figure 5.14: Akiyo video sequence. SNR of the decoded video in the case of single user. $m = 4$, $n = 3$, $X = 20$, $CS = 1$.

PSNR of the vertical, diagonal, and the speed coding methods are up to 45% more than that of the Percy method.

In Figure 5.14, we repeat our last experiment on Akiyo video sequence. The figure shows that our proposed methods can deliver a higher video quality than the Percy method in terms of PSNR. By comparing Figures 5.13 and 5.14, we can say that our methods are more efficient in the case of Bus video sequence than the Akiyo sequence. That is because of the different video characteristic. The Bus video sequence is more dynamic than the Akiyo method. As a result, losing some temporal or quality layers

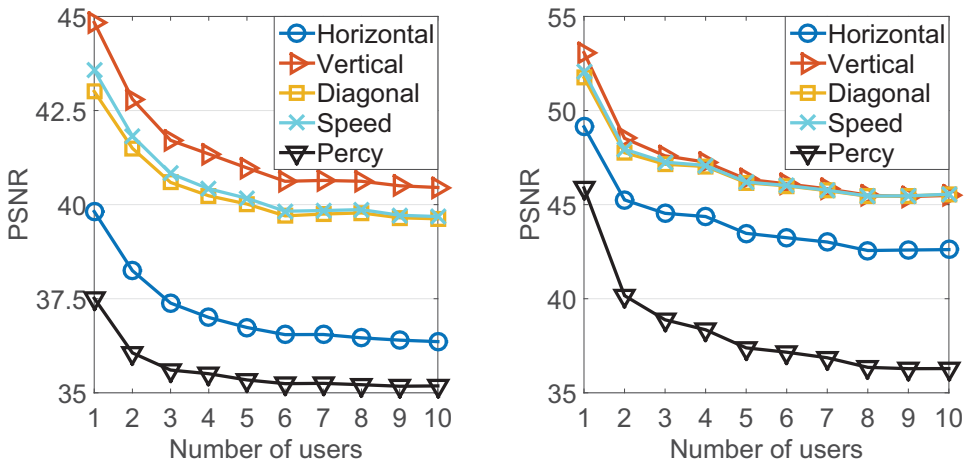


Figure 5.15: Bus video sequence. PSNR of the decoded video in the multiple users. Bus video sequence. $m = 4$, $n = 3$. (a) $X = 18$. (b) $X = 20$.

has a less impact on the total PSNR of the decoded video. In other words, the Akiyo video is naturally more robust against losses than the Bus video trace. As a result, even with an inappropriate protection method, the quality of the decoded video is higher than that of the Bus video.

Figure 5.15(a) shows the average PSNR of the decoded videos in the case of multiple users. The number of users varies from 1 to 5. Moreover, the reliability of the links are randomly chosen in the range of $[0.4, 0.9]$. The Figure shows that the PSNR of the vertical coding method is up to 20% more than that of the Percy method. Moreover, the average PSNR decreases as the number of users increases. As mentioned before, the reason is that as we increase the number of users, the diversity of the channels' delivery rate increases, which results in a decrease in the average number of layers delivered to the users.

In the next experiment, we increase the number of transmissions to 20, and repeat the previous experiment. The result is shown in Figure 5.15(b). As the number of transmissions increases, the gap between our coding methods decreases, which is due to the larger number of received layers in all of the methods. However, the PSNR of

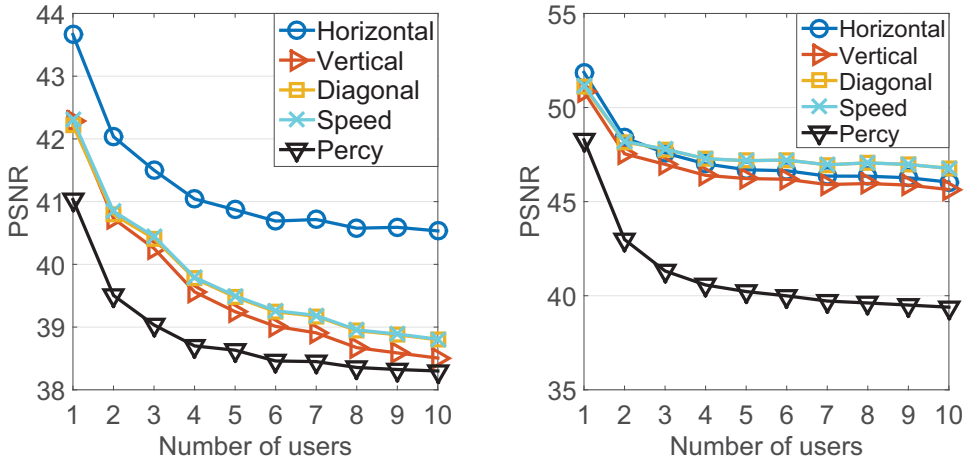


Figure 5.16: Akiyo video sequence. PSNR of the decoded video in the multiple users. $m = 4$, $n = 3$. (a) $X = 18$. (b) $X = 20$.

the Percy method is still much less than that in our methods. As Table 5.2 shows, in the Bus video sequence, the spatio layers have more of an impact on the PSNR. However, the Percy method cannot deliver partial temporal layers, and cannot assign more transmissions to the spatio layers than the temporal layers, which results in less PSNR compared to our methods.

We repeat the previous experiment on Akiyo video sequence. As Figure 5.16(a) shows, our proposed horizontal triangular coding results in the highest PSNR compared to the other methods. This is in contrast with the case of Bus video trace, in which the performance of horizontal triangular coding was less that that of the vertical, diagonal, and speed coding methods (Figure 5.15(a)). This can be justified by the difference in the characteristics of the Bus and Akiyo video sequences. As the number of users increase in Figure 5.16(a), the average PSNR decreases. However, the slop of this average PSNR loss decrease as we the number of users becomes more.

In Figure 5.16(b), we change the number of transmission from 18 to 20. in this figure, the performance of the horizontal coding method is less than that of the Diagonal as speed coding methods. From Figures 5.16(a) and (b) we find that a

coding method which is appropriate for a specific number of transmissions might not be efficient for the other number of transmissions.

5.4.3 Discussion and Summary

The presented simulation results suggest that the number of decodable layers in our proposed coding schemes are almost always more than that of the Percy method, which is due to more coding possibilities in our schemes. In the case of single destination, depending on the delivery rate of the link, different coding methods might be optimal. However, in general, the number of decodable layers of the speed coding method is more than that of the other methods. Also, the Percy method has the lowest number of decodable layers and PSNR among the compared methods.

Comparing the number of decodable layers and the PSNR values of the different methods, we can find that the relation between the number of decodable layers and PSNR is not linear. For instance, in the case of multiple users, the speed method has the largest number of decodable layers among the evaluated methods. However, for Bust video sequence, its PSNR is less than that of the vertical coding method.

5.5 Conclusion

As a result of the rapid increase in the popularity of wireless devices, such as smartphones and tablets, and video streaming over Internet, wireless video multicasting is becoming an important application. However, the diversity of users' channel conditions is a challenge to video multicasting to multiple receivers. Multi-resolution video coding, in which a video is divided into a set of base layer and enhancement layers, is an efficient method to address this challenge. In the previous works, it has been shown that a triangular network coding scheme can increase the quality of the received videos by the users. In this work, we propose a two-dimensional network coding method to further increase the users' experience. In our coding scheme, we

combine both temporal and spatio layers together, and we introduce a search algorithm to find the optimal distribution of the transmissions to different coded layers. Through simulations results, we show the effectiveness of our two-dimensional coding scheme, as compared to the one-dimensional network coding scheme. Our future work is to analyze the effect of spatio and temporal layers on the video quality, and to find a mechanism to select the appropriate coding strategy based on the characteristics of the videos.

CHAPTER 6

SYMBOL-LEVEL TRANSMISSION IN ERROR-PRONE WIRELESS NETWORKS

In Chapter 5 we proposed a network coding scheme to provide robust wireless video transmission. In this chapter, we study symbol-level data transmission in error-prone wireless networks using network coding. We first provide an introduction and motivation to the problem. After presenting the setting and formulating the problem, we propose our symbol-level network coding methods and discuss about the challenges. Finally, we present the evaluation results and conclusion of the chapter.

6.1 Introduction and Motivation

Broadcasting schemes are widely used for disseminating data and control messages in wireless networks. However, the error-prone wireless links creates challenges in these networks. To handle these challenges, different mechanisms [4, 5, 8–10] have been proposed to provide reliability. In the case of numeric data, e.g., the captured information by sensor nodes, the importance of the data (numbers) decreases from the left (most significant bit) to the right (least significant bit). Therefore, any mechanism that addresses numeric data transmissions in a lossy environment should consider the weights of the bits. The problem of reliable transmission has received a lot of attention; however, to the best of our knowledge, nobody has studied the problem of transmitting symbols (a group of bits) with different weights.

In contrast to the previous works, in this chapter, we propose a novel broadcasting

approach in wireless networks which considers the importance of the symbols. Instead of providing reliable transmissions and guaranteeing a full delivery of the data, we are interested in maximizing the expected total gain of the destination nodes, with a fixed given number of symbol transmissions. In applications such as transmitting numeric data from a source node to a set of destination nodes, encountering an error in more important bits has a more negative impact, and with a given number of transmissions, it is more efficient to allocate more transmissions to the most important part of the data.

Figure 6.1 (a) shows an example, in which a packet with 2 symbols is transmitted to a destination node. The weights of the symbols s_1 and s_2 are equal to 2 and 1, respectively. Assume that the error-rate of the link is equal to 0.6. The window size for transmitting the packet is equal to 2 symbols, and after that, another packet will be ready for transmission. In this case, the traditional methods transmit each symbol once. Now, let us compute the expected gain. We represent the number of transmissions of symbols s_1 and s_2 as x_1 and x_2, respectively. Thus, the probability of successful delivery of symbols s_1 and s_2 is equal to $1-\epsilon^{x_1}$ and $1-\epsilon^{x_2}$, respectively. Consequently, the expected gain is equal to $w_1 \times (1-\epsilon^{x_1}) + w_2 \times (1-\epsilon^{x_2})$, where w_1 and w_2 are the weights of symbols s_1 and s_2. The possible distribution of 2 transmissions and their respective utilities are shown in Figure 6.1 (b). The figure shows that it is more efficient to allocate both of the transmissions to symbol s_1. Now assume that the window size is equal to 3 transmissions. Figure 6.1 (b) shows that the optimal solution is allocating 2 transmissions to symbol s_1, and 1 transmission to symbol s_2. It should be noted that if there is no deadline, then the optimal solution is a simple extension from the channel coding theory [78].

Motivated by this example, we propose a novel broadcasting approach in wireless networks, which considers the importance of the symbols. Instead of providing reliable transmissions and guaranteeing a full delivery of the data, we are interested

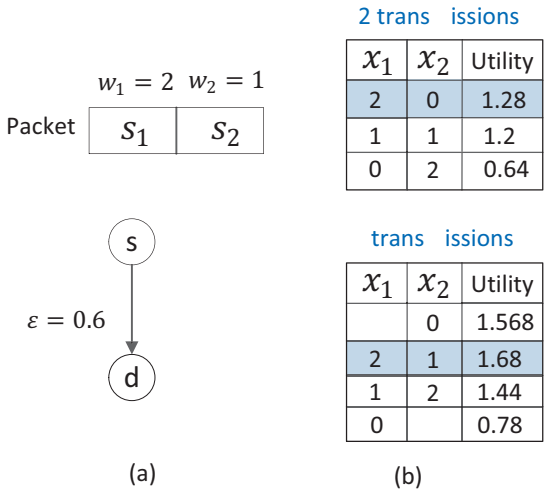


Figure 6.1: Motivation example; (a) setting, (b) the choices with 2 and 3 transmissions.

in maximizing the expected total gain of the destination nodes, with a fixed given number of symbol transmissions. In applications such as transmitting numeric data from a source node to a set of destination nodes, encountering an error in more important bits has a more negative impact, and with a given number of transmissions, it is more efficient to allocate more transmissions to the most important part of the data. In our first work, we study the optimal distribution of the transmissions to different symbols with unequal importance, in order to maximize the total expected gain.

6.2 Setting

We consider a single-hop wireless network that consists of one source and n destination nodes d, as depicted in Figure 6.2. The source node has a batch of k packets to send to the destination nodes, and each packet consists of m symbols. Each symbol itself might contain several bits. Each symbol has a weight w_i, and in general, $w_i > w_{i+1}, \forall i : 1 \leq i \leq m-1$. For simplicity, we assume that the weight of the i-th symbols of all of the packets are the same. However, the proposed solutions in this

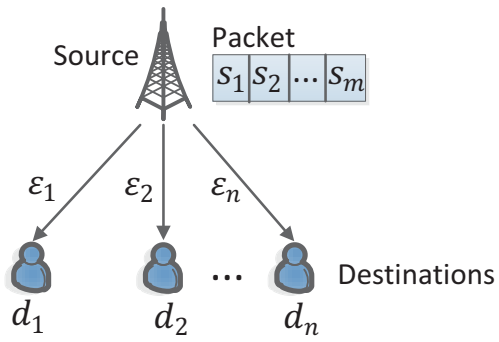


Figure 6.2: System setting.

chapter can be easily extended to the case of packets with different symbols' weights. We assume that the error rate of each transmitted symbol (or packet) from the source node to the i-th destination node is equal to ϵ_i. We represent the number of times that the i-th symbol is transmitted as x_i.

In our model, the packets of a batch have a deadline to be received by the destination nodes, which is equal to the window size, and after this time another batch of packets will be ready for transmission. As a result, channel coding, hierarchical coding, and unequal error protection methods cannot be applied in our setting. We assume that this window size for a batch of packets is enough for transmitting $t \times k$ symbols, where t is the assigned window for a single packet. If the packets are not delay sensitive, or the source has infinite packets to transmit, the optimal solution is a simple extension of the well-known channel coding theory [78]. Our goal is to maximize the total weight of the received symbols of a batch of k packets by the destination nodes. As a result, our utility function becomes:

$$u = k \times \sum_{i=1}^{m} \sum_{l=1}^{n} w_i \times (1 - {\epsilon_l}^{x_i}) \tag{6.1}$$

$$s.t. \quad \sum_{i=1}^{m} x_i = t$$

It is obvious that we should assign a larger portion of the transmissions to the

Table 6.1: The set of symbols used in this work.

Notation	Definition
d_i	The i-th destination node
n	The number of destination nodes
m	The number of symbols inside each packet
k	The number of packets
w_i	The weight of the i-th symbol of each packet
ϵ_i	The error rate of the link between the source and the i-th destination node.
t	The size of the transmission time window for each packet (in the term of number of symbols)
s_i	The i-th symbol (in the case of single packet)
$s_{j,i}$	The i-th symbol of the j-th packet
S_i	The i-th coded symbol
p_i	The i-th packet
Δ_{x_i}	The change in the utility gain as we increase x_i to $x_i + 1$
u	The utility function
u_i	The gain (utility) from the i-th symbols
u_i^{NC}	The gain (utility) from the i-th symbols when we use linear NC
u^{NC}	The total gain (utility) of using linear NC
u^{UC}	The total gain without using linear NC
c^{NC}	The header cost of a linear coded packet
c^{UC}	The header cost of an uncoded packet
τ	The number of sets of t transmissions performed so far
$\epsilon_{i,\tau}$	The actual error rates of the link between the source and node d_i in the set of τ-th set of transmissions
$\hat{\epsilon}_{i,\tau}$	The estimated error rates of the link between the source and node d_i in the set of τ-th set of transmissions
$r_{i,\tau}$	number of successfully received symbols by the destination node d_i in the set of τ-th set of transmissions

symbols that are more important than the other symbols, as successful delivery of these packets to the destination nodes results in more gain. However, it is not clear how we should assign and distribute the transmissions (duplications) to the different symbols of the packets in order to maximize the total gain. Our goal in this work is to find this optimal assignment. In the rest of the chapter we use gain and utility interchangeably. The set of symbols used in this chapter is summarized in Table 6.1.

For the case of data like binary data, in which the importance of the i-th bit is twice that of the $i + 1$-th symbol, the weight of the i-th symbol can be defined as

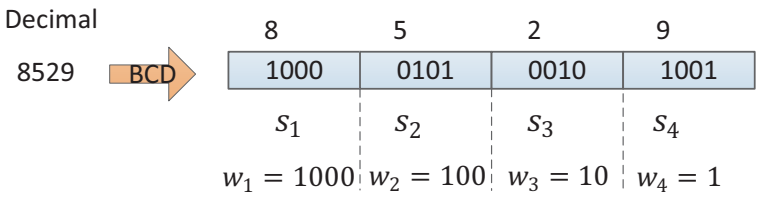


Figure 6.3: Binary coded decimal (BCD) and the weights of the symbols.

2^{m-i}. As a result, the objective function becomes:

$$k \times \sum_{i=1}^{m} \sum_{l=1}^{n} 2^{m-i} \times (1 - {\epsilon_l}^{x_i})$$

In this case each symbol contains one bit. In Binary-Coded Decimal (BCD), each decimal digit is represented with a fixed number of bits, usually 4 bits. Figure 6.3 shows a decimal number and its conversion to BCD. For the case of BCD, we can consider the 4 bits that correspond to the same decimal digit as a symbol, in which the weight of the i-th symbol is 10 times that of the $i+1$-th symbol. Consequently, for the BCD data, the objective function will be:

$$k \times \sum_{i=1}^{m} \sum_{l=1}^{n} 10^{m-i} \times (1 - {\epsilon_l}^{x_i})$$

In this work, we do not restrict the solution to a special weighting system, and solve the problem in the general case. The parameters in our proposed method can be adjusted based on the application and the structure of the data to be transmitted.

6.3 Optimal Solution for the Case of Single Packet

In the following two sections, we first find the optimal distribution of transmissions to different symbols in the case of destination node with homogeneous channels, and then we extend it for heterogeneous destination nodes.

6.3.1 Destinations with Homogeneous Channels

We first investigate and address the problem in the case of a packet size equal to 2 symbols. Then, we generalize the solution to the case of m symbols.

6.3.1.1 Packet Size $m = 2$

for a packet size $m = 2$, the objective function becomes:

$$\begin{aligned} & u = n \times \left[w_1 \times (1 - \epsilon^{x_1}) + w_2 (1 - \epsilon^{x_2}) \right] \\ s.t. \quad & x_1 + x_2 = t \end{aligned}$$

We denote the change in the total gain as we increase the i-th symbol's transmissions from x_i to $x_i + 1$ as Δ_{x_i}, so we have:

$$\begin{aligned} \Delta_{x_i} &= n \times w_i \times (1 - \epsilon^{x_i+1} - (1 - \epsilon^{x_i})) \\ &= n \times w_i \times (1 - \epsilon) \times \epsilon^{x_i} \end{aligned}$$

As mentioned in the setting, $w_1 > w_2$. Thus, it is obvious that, in order to achieve more gain, the number of times the source node transmits the first symbol should be more than or equal to that of the second symbol. If we consider the problem in t rounds of transmission, the first time we should increment x_2 and transmit the second symbol is when the gain of increasing x_1 is less than that of increasing x_2. In other words, the condition to increase x_2 is $\Delta_{x_1} < \Delta_{x_2}$. Consequently, we have:

$$n \times w_1 \times (1 - \epsilon)\epsilon^{x_1} < n \times w_2 \times (1 - \epsilon)\epsilon^{x_2}$$

and as a result,

$$\epsilon^{x_1} < \frac{w_2}{w_1}\epsilon^{x_2} \tag{6.2}$$

In this case, we are incrementing x_2 for the first time, so $x_2 = 0$, and we have:

$$\epsilon^{x_1} < \frac{w_2}{w_1} \tag{6.3}$$

Therefore, the first time we should increment x_2 is when $\epsilon^{x_1} < \frac{w_2}{w_1}$; we refer to this point as the *saturation point*. After this point, whenever $\epsilon^{x_1} < \frac{w_2}{w_1}\epsilon^{x_2}$, we should increment x_2, since it results in more gain. In contrast, if $\epsilon^{x_1} \geq \frac{w_2}{w_1}\epsilon^{x_2}$, we increment x_1.

We show the optimal distribution of the transmissions between x_1 and x_2 for different total numbers of transmissions t in Figure 6.4. The weights of symbols s_1 and s_2 in this example are assumed to be 5 and 1, respectively. To find the optimal distribution, we compute the utility for all of the possible distributions. It can be inferred from this figure that, after incrementing x_2 for the first time, the optimal solution has a round-robin incrementing pattern. The insight behind this phenomenon is as follows. The ratio of Δ_{x_1} and Δ_{x_2} is equal to:

$$\frac{n \times \Delta_{x_1}}{n \times \Delta_{x_2}} = \frac{w_1 \times (1-\epsilon) \times \epsilon^{x_1}}{w_2 \times (1-\epsilon) \times \epsilon^{x_2}} = \frac{w_1 \times \epsilon^{x_1}}{w_2 \times \epsilon^{x_2}} \tag{6.4}$$

Before we reach the saturation point, $\Delta_{x_1} \geq \Delta_{x_2}$, and the ratio in Equation (6.4) is greater than 1. However, after the saturation point, whenever we increment x_2, the ratio in Equation (6.4) is multiplied by $\frac{1}{\epsilon}$, and it becomes greater than 1. As a result, the next transmission should be assigned to x_1, as it results in more gain. In contrast, whenever we increment x_1, the ratio is multiplied by ϵ, which makes the ratio less than 1. In this case, it is more beneficial to assign the next transmission to

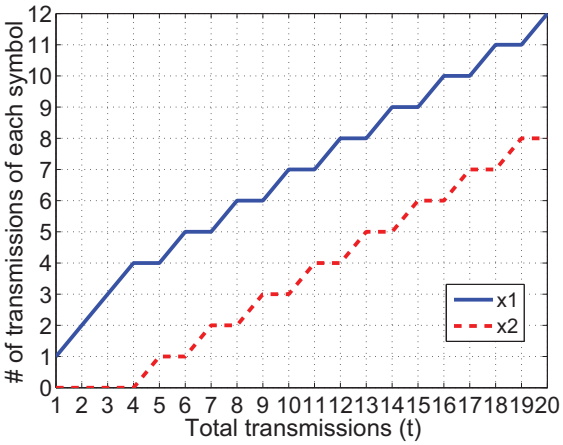


Figure 6.4: Optimal distribution of transmissions between 2 symbols for an error probability $\epsilon = 0.5$, $w_1 = 5$, and $w_2 = 1$.

x_2.

Based on the discussion, our algorithm works as follows. We iteratively increment x_1 and decrement t until $\epsilon^{x_1} < \frac{w_2}{w_1}$. If any more transmissions are left, we start to distribute these remaining transmissions between x_1 and x_2 in a round-robin pattern. In the following, we prove the optimality of this algorithm, and we show that the optimal solution has a round-robin pattern. The utility function in the case of transmitting one packet to homogeneous destinations is as follows:

$$u = \sum_{i=1}^{m} n \times w_i \times (1 - \epsilon^{x_i})$$
$$s.t. \quad \sum_{i=1}^{m} x_i = t$$

For the packet size equal to 2 symbols ($m = 2$) we have:

$$u = n \times \left[w_1 \times (1 - \epsilon^{x_1}) + w_2(1 - \epsilon^{x_2})\right]$$
$$s.t. \quad x_1 + x_2 = t$$

Lemma 6.1. *If* $\epsilon^{x_1} < \frac{w_1}{w_2}\epsilon^{x_2}$, *then* $\epsilon^{x_1} > \frac{w_1}{w_2}\epsilon^{x_2+1}$.

Proof. We use contradiction to proof Lemma 6.1. We refer to the optimal solution at the current iteration as (x_1, x_2). Assume that the current state is (x_1, x_2) and $\epsilon^{x_1} < \frac{w_1}{w_2}\epsilon^{x_2+1}$. As a result, $\epsilon^{x_1-1} < \frac{w_1}{w_2}\epsilon^{x_2}$, and we have:

$$w_1\epsilon^{x_1-1} < w_2\epsilon^{x_2}$$

By multiplying the two sides of this inequality with $n \times (1-\epsilon)$ we will have:

$$n \times w_1 \times (1-\epsilon)\epsilon^{x_1-1} < n \times w_2 \times (1-\epsilon)\epsilon^{x_2}$$
$$\Rightarrow \Delta_{x_1-1} < \Delta_{x_2}$$

As a result, it should be more efficient to increase x_2 in the previous iteration. Therefore, in the current iteration we will have $(x_1 - 1, x_2 + 1)$, which contradicts the assumption that the current state is (x_1, x_2). Consequently, we have $\epsilon^{x_1} > \frac{w_1}{w_2}\epsilon^{x_2+1}$. □

Lemma 6.2. *If* $p^{x_1} > \frac{w_1}{w_2}\epsilon^{x_2}$*, then* $\epsilon^{x_1+1} < \frac{w_1}{w_2}\epsilon^{x_2}$*.*

Proof. We use contradiction to proof Lemma 6.1. Assume that the current state is (x_1, x_2) and $\epsilon^{x_1+1} > \frac{w_1}{w_2}\epsilon^{x_2}$. As a result, $p^{x_1} > \frac{w_1}{w_2}\epsilon^{x_2-1}$, so we have:

$$w_1\epsilon^{x_1} > w_2\epsilon^{x_2-1}$$

By multiplying the two sides of this inequality with $1-\epsilon$ we will have:

$$n \times w_1 \times (1-\epsilon)\epsilon^{x_1} > n \times w_2 \times (1-\epsilon)\epsilon^{x_2-1}$$
$$\Rightarrow \Delta_{x_1} > \Delta_{x_2-1}$$

Therefore, it should be more efficient to increment x_2 in the previous state. Thus, in the current state, we will have $(x_1 + 1, x_2 - 1)$, in which $x_2 \geq 1$ ($x_2 - 1$ cannot be

negative) contradicts the assumption that the current state is (x_1, x_2). Consequently, $\epsilon^{x_1+1} < \frac{w_1}{w_2}\epsilon^{x_2}$. □

Theorem 6.3. *Assigning the transmissions to x_1 for $x_1 \leq \log_\epsilon \frac{w_2}{w_1}$ and then incrementing x_1 and x_2 in a round-robin pattern will result in the optimal solution.*

Proof. Based on Equation 6.3, if $\epsilon^{x_1} < \frac{w_2}{w_1}$ then $\Delta_{x_1} < \Delta_{x_2}$, so x_2 should be zero. In addition, based on Lemma 6.1 after this point, every time we increment x_2, Δ_{x_2+1} becomes less than Δ_{x_1}. Therefore, in this case, assigning the next transmission to x_1 results in a larger gain. Lemma 6.2 is the reverse of Lemma 6.1, which results in a round-robin incrementing pattern. □

6.3.1.2 General Packet Size m

Similar to the case of $m = 2$, the first symbol (the symbol with the smallest index) has more weight, so it is more important than the other symbols. As a result, we should not transmit other symbols until $\Delta_{x_1} > \Delta_{x_2}$. It should be noted that this condition implies that $\Delta_{x_1} > \Delta_{x_i}, \forall i : 2 \leq i \leq m$. The reason is that, $w_2 > w_i, \forall i : 3 \leq i \leq m$, and $x_i = 0, \forall i : 2 \leq i \leq m$. Consequently, similar to the case of packet size $m = 2$, the first time that we should increment x_2 is when $\epsilon^{x_1} < \frac{w_2}{w_1}$, and after this point, the transmissions should be distributed between x_1 and x_2. However, in contrast with the case of $m = 2$, after a specific point, we should start to transmit the third symbol. The condition to increment x_3 is when $\Delta_{x_1} < \Delta_{x_3}$ and $\Delta_{x_2} < \Delta_{x_3}$. For $\Delta_{x_1} < \Delta_{x_3}$ we have:

$$n \times w_1 \times (1-\epsilon)\epsilon^{x_1} < n \times w_3 \times (1-\epsilon)\epsilon^{x_3}$$

At this step, we are increasing x_3 for the first time; therefore, $x_3 = 0$, and the first optimality condition becomes $\epsilon^{x_1} < \frac{w_3}{w_1}$. Moreover, for the second condition

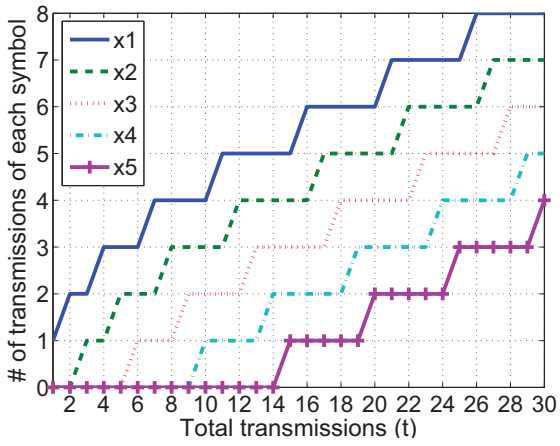


Figure 6.5: Optimal distribution of transmissions between 5 symbols for an error probability $\epsilon = 0.5$, $w_1 = 16$, $w_2 = 8$, $w_3 = 4$, $w_4 = 2$, and $w_5 = 1$.

$\Delta_{x_2} < \Delta_{x_3}$ we have:

$$n \times w_2 \times (1-\epsilon)\epsilon^{x_2} < n \times w_3 \times (1-\epsilon)\epsilon^{x_3}$$

As $x_3 = 0$, the equation becomes $\epsilon^{x_2} < \frac{w_3}{w_2}$. When these two conditions hold, we should start assigning the remaining transmissions to the first 3 symbols in a round-robin pattern. By the same reasoning, the condition for increasing x_m is when $\epsilon^{x_i} < \frac{w_m}{w_i}, \forall i : 1 \leq i \leq m-1$. Figure 6.5 shows the optimal distribution of the transmissions between different symbols when $m = 5$ for different numbers of total symbol transmissions t. The link's error rate and w_i are equal to 0.5 and 2^{5-i}, respectively. This figure shows that, even in the case of a packet size more than 2 symbols, the round-robin distribution of the transmissions results in the optimal solution.

We can summarize the discussion and the procedure of our weighted retransmission with homogenous destinations (WRH) algorithm as follows. We assign the transmissions to x_1 until $\epsilon^{x_1} < \frac{w_2}{w_1}$. Then, we distribute the remaining transmissions between x_1 and x_2 until $\epsilon^{x_1} < \frac{w_3}{w_1}$ and $\epsilon^{x_2} < \frac{w_3}{w_2}$. After this point, we continue the

round-robin distribution of the remaining transmissions among x_1, x_2, and x_3. In general, we start incrementing x_j when $\epsilon^{x_i} < \frac{w_j}{w_i}, \forall i : 1 \leq i \leq j-1$, and we add x_j to the round-robin incrementing process. We continue this process until t becomes 0. In the following, we proof the optimality of the WRH algorithm.

Lemma 6.4. *If* $\epsilon^{x_i} > \frac{w_j}{w_i}\epsilon^{x_j}\ \forall i, j \in [1, m], j \neq i$, *then* $\epsilon^{x_i+1} < \frac{w_j}{w_i}\epsilon^{x_j}$.

Proof. Assume that the current state is $(x_1, x_2, ..., x_m)$, and there is a j such that $\epsilon^{x_i+1} > \frac{w_j}{w_i}\epsilon^{x_j}$. Then, $\epsilon^{x_i} > \frac{w_j}{w_i}\epsilon^{x_j-1}$ in one of the previous states. As a result, $\Delta_{x_i} > \Delta_{x_j-1}$, so we should see a state with $x_i + 1$ and $x_j - 1$. In this case, there is no way to see the current state, which contains x_i and x_j. □

Theorem 6.5. *The WRH algorithm results in an optimal solution.*

Proof. It can be inferred from Lemma 6.4 that the optimal assignment has a round-robin pattern. The reason is that, when we increment x_i, ϵ^{x_i} becomes less than $\frac{w_j}{w_i}\epsilon^{x_j}, \forall j : j \neq i$. The next time ϵ^{x_i} becomes greater than $\frac{w_j}{w_i}\epsilon^{x_j}$ is when we increment all $x_j, j \neq i$. □

The binary and BCD representations of decimal number 83 are shown in Figures 6.6 (a) and (b), respectively. In BCD, the weight of symbol s_1 is 10 times that of symbol s_2. Also, the weight of symbol s_i is twice that of symbol s_{i+1} in binary representation. Assuming that the error rate of the link between the source and destination nodes is equal to 0.2, we show the optimal solutions in the cases of 8, 12, and 16 symbol transmissions for the binary number in Figure 6.6 (a). Note that, in this case, each symbol contains one bit. In BCD, the size of each symbol is 4 times that of the binary representation. Therefore, in Figure 6.6 (b), we show the optimal transmissions with 2, 3, and 4 symbol transmission in the BCD representation.

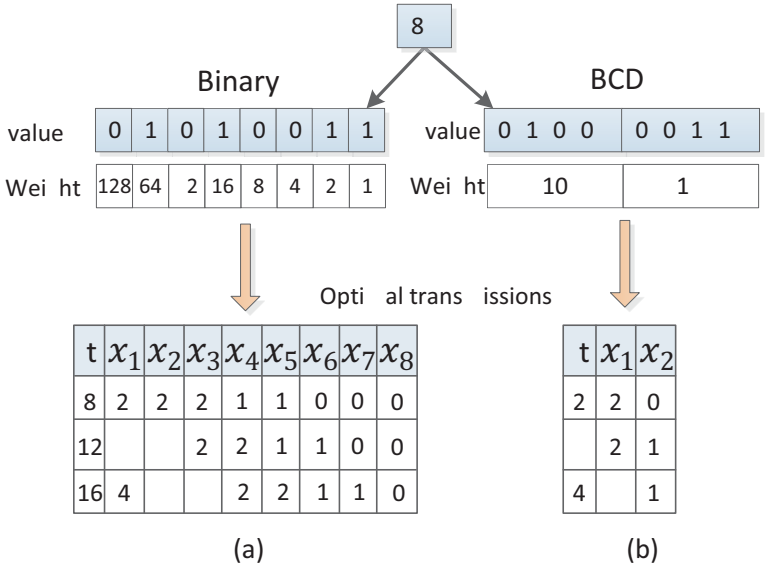


Figure 6.6: The binary and BCD representations of a decimal number, and the optimal symbol transmissions to homogeneous destinations with a different total number of symbol transmissions. $\epsilon = 0.2$; (a) Binary representation, (b) BCD representation

6.3.2 Destinations with Heterogenous Channels

In the case of multiple destination nodes with different transmission error rates, the round-robin distribution pattern does not exist. For this reason, we use an iterative algorithm, which we call weight retransmission (WR). In each iteration of the WR method, we assign one transmission to a symbol such that it maximizes the increase in the total gain. In the case of heterogenous destination nodes, Δ_{x_i} can be calculated as follows:

$$\begin{aligned}\Delta_{x_i} &= w_i \times \sum_{l=1}^{n} \left[1 - \epsilon_l^{x_i+1} - (1 - \epsilon_l^{x_i})\right] \\ &= w_i \times \sum_{l=1}^{n} \left[\epsilon_l^{x_i} - \epsilon_l^{x_i+1}\right]\end{aligned}$$

Algorithm 8 WR Algorithm

```
1: for i=1 to m do
2:     x_i = 0
3: for j=1 to t do
4:     max = 0
5:     argmax = 0
6:     for i=1 to m do
7:         Δ_{x_i} = w_i × Σ_{l=1}^{n} (ε_l^{x_i} − ε_l^{x_i+1})
8:         if Δ_{x_i} > max then
9:             max = Δ_{x_i}
10:            argmax = i
11:    x_{argmax} = x_{argmax} + 1
```

and the total utility is equal to:

$$u = \sum_{i=1}^{m} \Big[w_i \times \sum_{l=1}^{n} (1 - \epsilon_l^{x_i}) \Big]$$

The WR algorithm assigns the total number of transmissions t to the different symbols in t rounds. At each iteration, our algorithm computes $\Delta_{x_i}, \forall i : 1 \leq i \leq m$, and it assigns the current transmission to x_j that increasing its number of transmissions by one results is more increase in the total gain. In other words, $j = \arg\max_{1 \leq i \leq m} \Delta_{x_i}$. Algorithm 8 shows the iterative process.

The second loop (the loop over j) and its internal for loop in Algorithm 8 run t and m times, respectively. Moreover, Δ_{x_i} is a summation over n nodes. As a result, the complexity of the WR method is in the order of $O(t \times m \times n)$.

Lemma 6.6. *The optimal x_i, $\forall 1 \leq i \leq m$ are non-decreasing as we increase the number of transmissions t.*

Proof. The utility of a symbol s_i is equal to:

$$\sum_{l=1}^{n} w_i \times (1 - \epsilon_l^{x_i}) \tag{6.5}$$

which is a non-decreasing function. Therefore, assigning more transmissions to a

symbol results in more utility. Moreover, the utility of each symbol is a summation of concave functions; therefore that is a concave function. It means that the Δ_{x_i}, $\forall 1 \leq i \leq m$ is a decreasing function. Assume that for a given t', the optimal number of transmissions for symbols s_j and s_k are equal to x_j and x_k, respectively. Moreover, for a $t > t'$ transmissions, the optimal number of transmission for s_j and s_k are $x_j - y$ and $x_k + y$, respectively, where y is a given positive number. It contradicts with the optimality of x_j and x_k transmissions in the case of t' total transmissions. The reason is that if $x_j - y$ and $x_k + y$ results in more gain, then x_j and x_k cannot result in optimal solution for the case of t' transmissions. Note that this holds since the utility of each symbol (Equation (6.5)) is a concave and non-decreasing function. Consequently, x_i are non-decreasing. □

The following corollary can be concluded from Lemma 6.6.

Corollary 6.7. *The optimal solution for t transmissions can be calculated from the optimal solution for $t' < t$ transmissions.*

Theorem 6.8. *The WR algorithm results in an optimal solution.*

Proof. We proof the optimality of the WR algorithm by induction. Let $t = 1$. It is obvious that the transmission should be assigned to the symbol s_i with the maximum Δ_{x_i}. Now, assume that for $t - 1$ transmissions the optimal solution is $(x_1, .., x_m)$. By Lemma 6.6, each x_i is non-decreasing. Therefore, from Corollary 6.7, in order to find the optimal solution for t transmissions, we just need to find the symbol s_i with the maximum Δ_{x_i} and increase x_1 by one. That is exactly the same as what the WR algorithm performs. □

6.4 Efficient Solution for the Case of Multiple Packets

In order to broadcast a batch of k packets from a source node to a set of destination nodes, we can use two approaches: without and with network coding. We describe

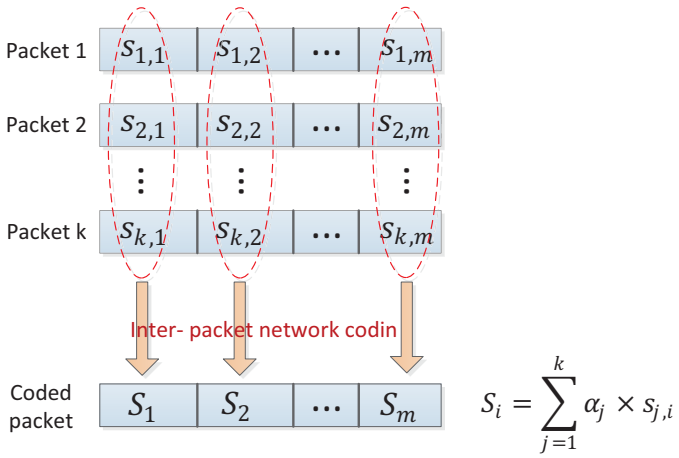


Figure 6.7: Inter-packet network coding.

the details of the mechanisms in the following sections.

6.4.1 Without Network Coding

In our model, we assume that the packet sizes (in term of symbols) are equal. Moreover, the weights of the i-th symbols in different packets are the same. As a result, the problem of sending k independent packets becomes k similar problems with the same solution. Consequently, we can simply use the result of the previous section, and repeat the same process for the different packets. In the weighted multiple packets retransmission (WMPR) mechanism, we first compute the optimal number of transmissions for each symbol. For this purpose, we perform one of the proposed algorithms in the previous section (WRH or WR), depending on the channels condition. Then, we use the output values x_i from the first step, and transmit each of the i-th symbols of the different packets x_i times. As we repeat the same process on k packets, the utility of this scheme is k times the gain of transmitting one packet.

6.4.2 Inter-Packet Network Coding

Random linear network coding can increase the gain of the WMPR mechanism. In our heuristic algorithm with network coding, much similar to the WMPR method, we run the WRH or WR algorithms to compute the optimal value of $x_i, \forall i : 1 \leq i \leq m$. Then, as it is shown in Figure 6.7, we code all of the i-th symbols of the k packets together. We denote the i-th coded symbols, as S_i. The coded symbols are in the form of $S_i = \sum_{j=1}^{k} \alpha_j \times s_{j,i}$, where $\alpha_{j,i}$ is a random coefficient. In this scheme, the source node generates and sends $x_i \times k$ coded symbols from the i-th original symbols. This is in contrast with the WMPR approach, in which the source node transmits the i-th symbol of each packet x_i times ($x_i \times k$ transmissions for k packets). We refer to our proposed weighted multiple packets retransmission method with network coding as WMPR-NC method.

In the discussed inter-packet network coding policy, each destination node is able to decode the i-th coded symbols and retrieve the k original i-th symbols of different packets, if it receives at least k linearly independent coded symbols. The decoding phase can be done using Gaussian elimination for solving a system of linear equations. Consequently, the gain from the i-th symbols of the k packets can be calculated using the following equation:

$$u_i^{NC} = w_i \times k \times \sum_{l=1}^{n} \Big[\sum_{j=k}^{x_i \times k} \binom{k \times x_i}{j} \times (1 - \epsilon_l)^j \times \epsilon_l^{x_i \times k - j} \Big] \tag{6.6}$$

In Equation (6.6), we multiply w_i by k since, when we code the i-th symbols of the k packets together, any destination node can decode all of the symbols, or none of them. The total number of transmissions for the set of i-th symbols is equal to $x_i \times k$; as a result, the probability of receiving j coded symbols correctly, and happening error in the rest of the coded symbols, is equal to $\binom{k \times x_i}{j} \times (1 - \epsilon)^j \times \epsilon^{x_i \times k - j}$, where $\binom{k \times x_i}{j}$ is the number of possible ways to select j coded symbols out of the transmitted

coded symbols. A node needs at least k coded symbols to decode the coded symbols; therefore, the number of received coded symbols should be in the range of k and $x_i \times k$.

Because of using network coding, each coded symbol contributes the same amount of information to the destination nodes. Therefore, receiving any k coded symbols is sufficient for retrieving the symbols. This is in contrast to the case of non-coding transmissions, in which a destination node might not receive some of the symbols, and might receive the other symbols multiple times. In this case, receiving a symbol multiple times does not contribute to the total gain. However, network coding decreases the probability of receiving partial i-th symbols of the packets. The reason is that, if a destination node receives enough coded symbols, it can decode the coded packets and retrieve all of the original symbols; however, it cannot decode the coded symbols in the case of receiving an insufficient number of coded packets.

Consider the example in Figure 6.8, in which the source node wants to send two single symbol packets to the destination node d_1. Assume that the transmission error rate is equal to 0.5, and $x_1 = 2$. The WMPR scheme sends each symbol twice. As a result, the probability of the destination node receiving both of the packets is equal to $(1-\epsilon^2) \times (1-\epsilon^2) = 0.5625$, and the probability of receiving just one of the packets is equal to $2 \times (1-\epsilon^2) \times \epsilon^2 = 0.3750$. On the other hand, the WMPR-NC scheme sends 4 random linear combinations of the symbols. Therefore, the destination node can decode and recover both of the symbols, if it receives at least any 2 coded symbols out of the 4 transmitted coded symbols. In this case, the probability of retrieving both of the packets is equal to $1-\epsilon^4-3\times\epsilon^3\times(1-\epsilon) = 0.75$, which is more than the WMPR mechanism. The reason for this difference is that, in the case of non-coded symbols, the destination node needs to receive each of the transmitted symbols at least once, and receiving one of the symbols twice does not have any advantage. However, in the case of network coding, the probability of retrieving just one of the symbols is equal

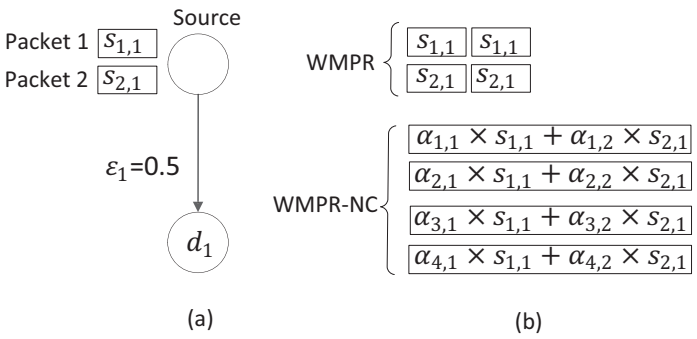


Figure 6.8: Example of inter-packet network coding.

to 0; as in random linear network coding, partial retrieval is not possible.

Figure 6.9 shows the gain of the network coding and no coding approaches for a different number of transmissions t. The number of packets and the link's error rate are equal to 10 and 0.5, respectively. It can be inferred from the figure that, in this case, for a t greater than 2, it is more efficient to use the proposed inter-packet network coding. In contrast, for a t less than or equal to 2, we should avoid using network coding, since it reduces the gain.

Referring to our discussion, for each set of symbols from the different packets, it might be beneficial to use network coding, or it might be more efficient to avoid using network coding. Therefore, for each set of the i-th symbols of the packets, we compute the utility of the non-coding and coding mechanisms. If the performance of the coding policy is more than that of the non-coding, we generate $k \times x_i$ random coded symbols, where x_i is the optimal number of transmissions when we use the non-coding mechanism. This process is shown in Algorithm 9. It should be noted that, if it is more efficient to transmit the i-th symbols of the packets without using network coding, we do not need to continue the algorithm for the remaining symbols. The reason is that, always, $x_j \leq x_i, \forall i, j : j > i$, as $w_j \leq w_i$. Therefore, if is not efficient to encode the i-th symbols together, it is definitely not efficient to encode the j-th symbols.

Algorithm 9 WMPR-NC Algorithm

Compute the optimal $\vec{x}$ by running WRH or WR merhods
for i=1 to m **do**
  $u_i = w_i \times k \times \sum_{l=1}^{n}(1-\epsilon_l^{x_i})$
  $u_i^{NC} = w_i \times k \times \sum_{l=1}^{n}\left[\sum_{j=k}^{x_i \times k}\binom{k \times x_i}{j} \times (1-\epsilon_l)^j \times \epsilon_l^{x_i \times k - j}\right]$
  if $u_i^{NC} > u_i$ **then**
    for i=1 to $k \times x_i$ **do**
      Create a random linear combination of the i-th symbols

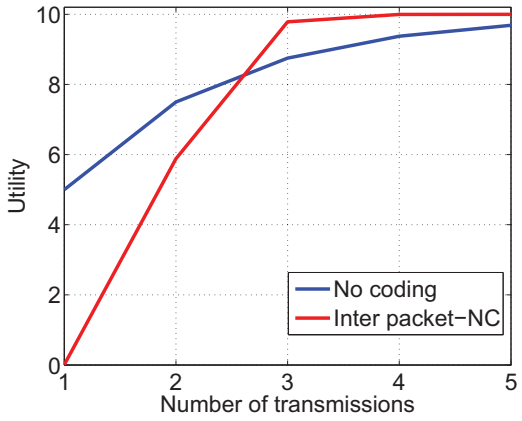


Figure 6.9: Comparison between the gain of the inter-packet network coding and no coding mechanisms, error probability $\epsilon = 0.5$, number of packets $k = 10$.

6.5 Implementation

6.5.1 Packet Header

After assigning the transmissions to the symbols, we should put them together to form the packets. In the WRH, WR, and WMPR mechanisms, we need to specify the index of each symbol in the packet. If we had just one transmission for each symbol, we could simply mention the first and the last index of the symbols that are included in the packet. Then we could put the symbols in the packet in increasing order of their index. However, in our schemes, each symbol might be included in a packet several times. As a result, we need 3 fields in the header to indicate the locations of symbol s_i. The first field represents the index of the symbol. The second and the third fields are used to show the starting and the ending locations of symbol s_i in the

Algorithm 10 Optimal header duplication

1: $Max\ gain = 0$
2: **for** $x_0 = 1$ to $t - 1$ **do**
3: depending on the setting, run the WRH, WR, and WMPR algorithms to compute the optimal $\vec{x}$ in transmitting $t - x_0$ symbols
4: use Equation (6.7) to compute u
5: **if** $u > Max\ gain$ **then**
6: $Max\ gain = u$
7: **else**
8: return x_o and $\vec{x}$
9: exit loop

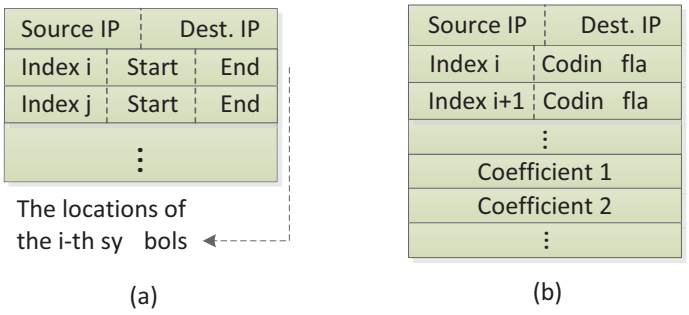


Figure 6.10: Packets' header, (a): The WRH, WR, and WMPR mechanisms, (b): The WMPR-NC mechanism.

packet, respectively. Figure 6.10 (a) shows the structure of the header in the WRH, WR, and WMPR mechanisms.

The header contains important information about the location of the symbols in the packet. As a result, the header must be received correctly by the destination nodes. To increase the reliability, forward error correction (FEC) codes [12–14] can be used. In addition to FEC codes, we can include the header multiple times in the packet, as this part of the packet is much more important than the other parts. If we consider the correct delivery of the header, the Objective Function (6.1) can be

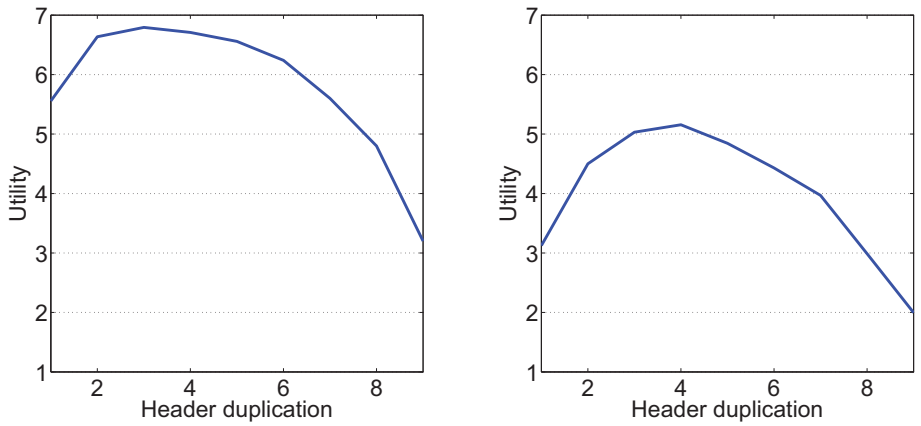
rewritten as follows:

$$u = k \times \sum_{i=1}^{m} \sum_{l=1}^{n} w_i \times (1 - \epsilon_l^{x_o}) \times (1 - {\epsilon_l}^{x_i}) \tag{6.7}$$
$$s.t. \quad \sum_{i=0}^{m} x_i = t$$

where x_0 is the header duplication.

Consider Figures 6.11 (a) and (b). We assign different values to x_0 and run the WRH algorithm to find the optimal distribution of the remaining transmissions to the symbols. Figures 6.11 (a) and (b) show the maximum achievable gain when the total number of transmissions is equal to 10, and the error rates are equal to 0.2 and 0.5, respectively. These figures show that, as we increase the duplication of the header, the total gain increases. The reason is that, a correctly received symbol is not useful unless the header is also received correctly. However, after a specific point, the total gain starts to decrease. To find the optimal header duplication x_0, we start with $x_0 = 1$, and run the WRH, WR, and WMPR algorithms to compute the optimal x_i in transmitting $t - 1$ symbols. We repeat the same process for $x_0 = 2$ and $t - 2$, and stop once we find that the utility decreases as we increment x_0. The details are shown in Algorithm 10.

In the WMPR-NC mechanism, the i-th symbol might be encoded or non-coded. Therefore, we need a flag field to indicate the encoded symbols. The packets' header in the WMPR-NC method is shown in Figure 6.10 (b). In addition to the source and destination IP addresses, we use *index* and *coding flag* to show the encoded symbols. The coefficients of the coded symbols are also included at the end of the header, which increases the overhead. In order to decrease this overhead, we can put some predefined random coefficient vectors on the destination and the source nodes. In this way, instead of including the coefficient in the header, the source can just put the index of the coefficient vectors in the header. In order to make the coefficient



(a) (b)

Figure 6.11: Optimal header duplication, total number of transmissions equal to 10; (a): $\epsilon = 0.2$. (b): $\epsilon = 0.5$.

vectors useful for any packet batch size, the size of the predefined vectors should be chosen long enough. If the size of a given batch is less than the vector size, the extra elements of the vector can be ignored by the destination nodes.

6.5.2 Packet Header Overhead

It is clear that the header overhead of network coded packets is more than that of the uncoded packets. As a result, depending on the header costs, network coding might be efficient or inefficient. In order to consider the packet's header cost, we modify Algorithm 9 to Algorithm 11, and refer it as the WMPR-header algorithm. We denote the total gain in the case that network coding is enabled as u^{NC}. Moreover, the total gain without network coding (uncoded packets) is represented as u^{UC}. We first compute the utilities in the cases that network coding is enabled or disabled (raw utilities). Each iteration of the loop computes the utility of each symbol. In each iteration of the for loop, we add $\max(u_i^{NC}, u_i)$ to u^{NC}, since the i-th symbols in the network coding mode can be coded or uncoded (see Algorithm 9).

After computing the raw utilities, we subtract the header costs from the raw

Algorithm 11 MPT-header

$u^{NC} = 0$
$u^{UC} = 0$
Compute the optimal $\vec{x}$ by running WRH or WR methods
for i=1 to m **do**
    $u_i = w_i \times k \times \sum_{l=1}^{n}(1 - \epsilon_l^{x_i})$
    $u_i^{NC} = w_i \times k \times \sum_{l=1}^{n}\left[\sum_{j=k}^{x_i \times k} \binom{k \times x_i}{j} \times (1-\epsilon_l)^j \times \epsilon_l^{x_i \times k - j}\right]$
    $u^{NC} = u^{NC} + \max(u_i^{NC}, u_i)$
    $u^{UC} = u^{UC} + u_i$
$u^{NC} = u^{NC} - c^{NC} \times (\sum_{i=1}^{m} w_i) \times k \times \frac{t}{m} \times n$
$u^{UC} = u^{UC} - c^{UC} \times (\sum_{i=1}^{m} w_i) \times k \times \frac{t}{m} \times n$
if $u^{NC} \leq u^{UC}$ **then**
    Turn off network coding

utilities. Assume that the header cost of a linear coded packet and an uncoded packet are equal to c^{NC} and c^{UC}, respectively. The value of each packet is equal to $\sum_{i=1}^{m} w_i$. Moreover, we have k packets and each of them will be transmitted $\frac{t}{m}$ times (note that t is the total number of symbol transmissions for each packet). Consequently, the total overhead of network coded packets is equal to $c^{NC} \times (\sum_{i=1}^{m} w_i) \times k \times \frac{t}{m}$. We are computing the total utility of n nodes; thus, we multiply the overhead by n and subtract it from the raw utilities. If $u^{NC} \leq u^{UC}$, we disable network coding, as it reduces the gain. Algorithm 11 shows the details.

6.5.3 Unknown Channel

So far, we have assumed that the channel erasure probabilities are perfectly known by the source node. The total gain is highly dependent on the error rate of the links; therefore, the source node needs to learn them, when it does not have perfect channel knowledge. For this purpose, each destination node d_i sends a feedback message to the destination node at the end of the $t \times k$ transmissions by the destination node (t transmissions for the case of single packet), which contains the number of successfully received symbols. Assume that the number of correctly received symbols in the last transmission window τ, and the estimated error rate of the destination node d_i after

Algorithm 12 Updating channels' error rate

1: After the τ-th set of transmissions update the error rates $\epsilon_{i,\tau+1}$, as follows,
2: **for** i=1 to n **do**
3: Receive $r_{i,\tau}$ from destination d_i
4: $\epsilon_{i,\tau} = \frac{t \times k - r_{i,\tau}}{t \times k}$
5: $\hat{\epsilon}_{i,\tau+1} = \frac{(\tau-1) \times \hat{\epsilon}_{i,\tau} + \epsilon_{i,\tau}}{\tau}$

the τ-th set of transmissions, are equal to $r_{i,\tau}$ and $\hat{\epsilon}_{i,\tau}$, respectively. Accordingly, the estimated channel error rate of the destination node d_i is given by:

$$\hat{\epsilon}_{i,\tau+1} = \frac{(\tau - 1) \times \hat{\epsilon}_{i,\tau} + \epsilon_{i,\tau}}{\tau} \tag{6.8}$$

where $\epsilon_{i,\tau}$ represents the error rate of the link between the source and node d_i in the τ-th set of transmissions, and can be calculated as follows:

$$\epsilon_{i,\tau} = \frac{t \times k - r_{i,\tau}}{t \times k}$$

In Equation (6.8), we multiply $\tau - 1$ by $\hat{\epsilon}_{i,\tau}$ to compute the total error rate in the $\tau - 1$ set of transmissions. Then, we sum it up with the measured error rate in the last set of transmissions, and compute the average error rate. Algorithm 12 shows the updating process of the error rates.

6.6 Simulation

6.6.1 Setting

In this section, we evaluate our proposed mechanisms WRH (weighted retransmission with homogenous destinations), WR (weighted retransmission with heterogenous destinations), WMPR (weighted multiple packets retransmission), and WMPR-NC (weighted multiple packets retransmission with network coding). We compare our proposed mechanisms with a simple retransmission (SR) method. In this method,

we distribute the transmissions evenly to the different symbols of the packets. As mentioned in the setting, the packets of a batch have a deadline to be received by the destination nodes, which is equal to the window size, and after this time another batch of packets will be ready for transmission. Thus, channel coding, hierarchical coding, and unequal error protection methods cannot be applied in our setting. That is the reason we do not include them in our simulations. Moreover, the objective of the mentioned papers in the related work is to provide 100% reliability, and they do not have any constraint on the number of transmissions. In contrast, we want to maximize the gain with a fixed number of transmissions. We run the simulations on 1,000 random topologies, with different links' error rates, and for each of the random topologies, we run the simulations 10 times. The plots in this chapter are based on the average outputs of the simulation runs. We assume that the weight of the i-th symbol of a packet is equal to 2^{m-i}. The tunable metrics in the simulations are as follows:

- *Total number of transmissions*: in order to study the effect of the number of transmissions on the total gain, we evaluate the methods with a number of transmissions in the range of m and $4 \times m$ for each packet.

- *Packet size*: the number of symbols in each packet in different plots are in the range of 5 to 10.

- *Number of packets*: in the case of multiple packets transmission, we change the number of packets that the source node transmits to the destination nodes from 20 to 50.

We choose these ranges since we believe that they are reasonable numbers in a typical scenario.

6.6.2 Simulation Results

6.6.2.1 Single Packet

In the first experiment, we compare the total gain of the WRH and the SR methods in Figure 6.12 (a). The packet size in this experiment is equal to 10 symbols. Also, the number of destination nodes and the link error probability are equal to 5 and 0.3, respectively. It is clear that the total gain should increase as we increase the total number of transmissions, which can be seen in the figure. Moreover, the figure shows that the difference between the WRH and the SR methods decreases as we increase the total number of transmissions from 10 to 40 symbols. The reason is that the successful delivery of all of the symbols approaches 1 in both of the mechanisms as we increase the number of retransmissions. Figure 6.12 (a) shows that the total gain of the WRH mechanism is up to 30% more than that of the SR method.

We increase the link's error rate to 0.5, and repeat the previous experiment in Figure 6.12 (b). Similar to Figure 6.12 (a), the difference between the two mechanisms decreases as we increase the number of retransmissions in Figure 6.12 (b). However, by comparing Figures 6.12 (a) and (b), we find that the efficiency of our proposed mechanism, WRH, increases as the link's error rate increases. The total gain of the WRH approach in this figure is up to 60% more than that of the SR method.

In the next experiment, we evaluate the gain of the WR mechanism in sending a packet to multiple destinations, by comparing it to the SR method in Figure 6.13 (a). We set the packet size to 10 symbols, and transmit a total of 10 symbols. In each of the 1,000 runs, the links' error rates are randomly chosen in the range of $[0.2, 0.4]$. The figure shows that the gain of both of the mechanisms increase as we increase the number of destinations; this is due to the presence of more receiver nodes. Also, it is clear from the figure that the relationship of the total gain and the number of destinations is linear, which is because of the independence of the links. As a result,

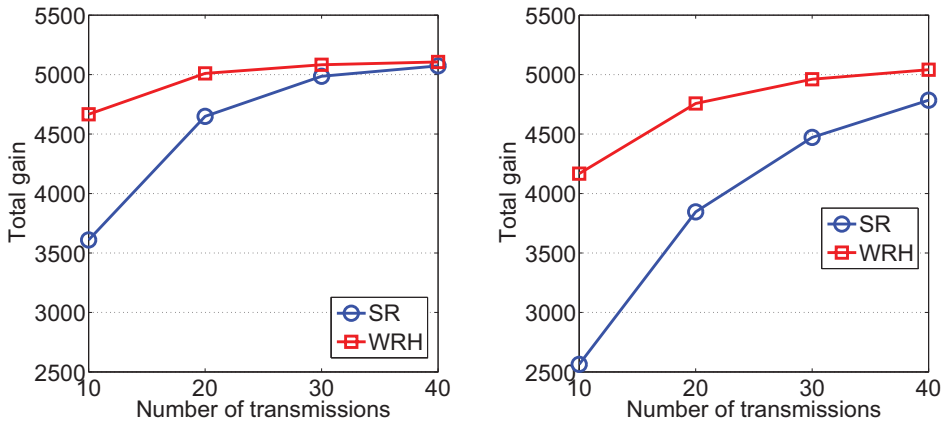


Figure 6.12: Comparison between the gain of the WRH and SR mechanisms in the case of single packet transmission, $m = 10$, $k = 1$, $n = 5$; (a) $p = 0.3$, (b) $p = 0.5$.

the ratio of the gain of the mechanisms is fixed in this figure.

We repeat the previous experiment in Figure 6.13 (b) by increasing the range of the links' error rates to $[0.2, 0.6]$. As it is expected, the gains of the mechanisms in Figure 6.13 (b) are less than that of Figure 6.13 (a). The efficiency of the WR mechanism increases as the error rates increase.

6.6.2.2 Multiple Packets

Figure 6.14 (a) shows the total gain of the WMPR, WMPR-NC, and SR mechanisms. In this figure, the packet size is equal to 5 symbols. Also, the number of destination nodes is equal to 5, and the error rate of the links between the source and the destination node is equal to 0.4. We increase the total number of transmissions as we increase the number of packets, and it is equal to the total number of symbols (total number of symbols is equal to 5 times the number of packets). As it is expected, the gain of the WMPR-NC mechanism is more than that of the other methods. Moreover, the gain of the WMPR mechanism is more than that of the SR method. Figure 6.14 (a) shows that the gain of the WMPR-NC mechanism is up to

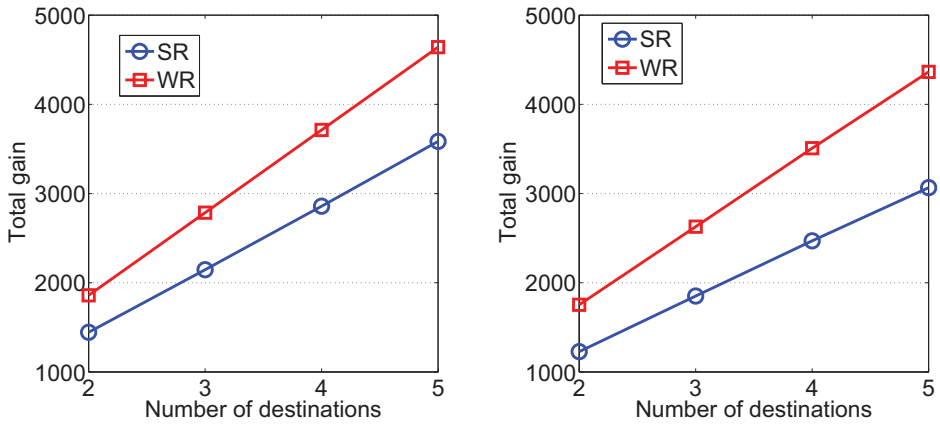


Figure 6.13: Comparison between the gain of the WR and SR mechanisms in the case of single packet transmission, $m = 10$, $k = 1$, $t = 10$; (a) $p \in [0.2, 0.4]$, (b) $p \in [0.2, 0.6]$.

15%, and 45% more than that of the WMPR and SR methods, respectively. Also, the efficiency of the network coding increases as we increase the number of packets, which are coded together.

We evaluate the effect of the link's error rate on the gain in Figure 6.14 (b). The packet size and the number of packets are equal to 5 symbols and 50, respectively. Also, for the total 250 symbols that the source node needs to transmit, we set the total number of transmissions to 250. The figure shows that the total gain of the WMPR and SR mechanisms drop dramatically as we increase the error rate. In contrast with the other methods, WMPR-NC is more robust to the error rate, which is due to the use of network coding.

We repeat the experiment of Figure 6.14 (a) in Figure 6.15 (a) with 5 destination nodes. The packet size is equal to 5 symbols, and the links' error rates are in the range of $[0.3, 0.5]$. Much similar to Figure 6.14 (a), the gain of all of the mechanisms increase as we increase the number of packets. Note that we increase the total number of transmissions as we increase the number of packets. By comparing Figure 6.14 (a) with Figure 6.15 (a), we find that the difference between the WMPR and WMPR-NC

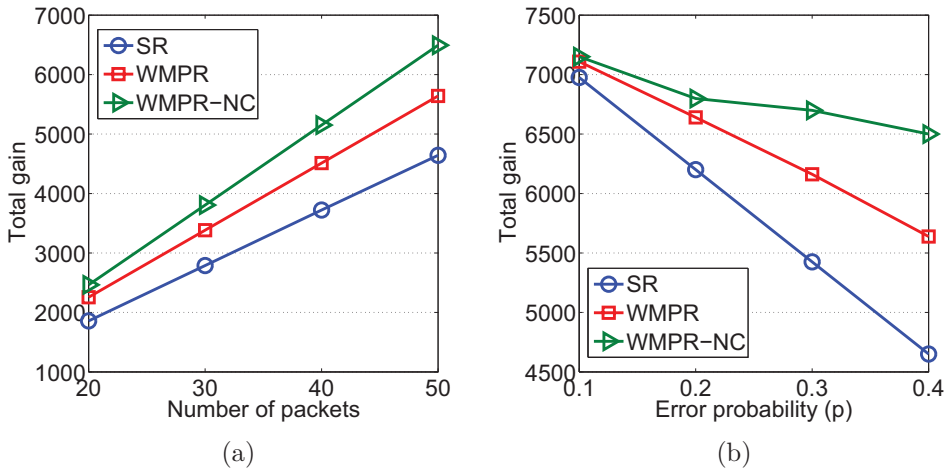
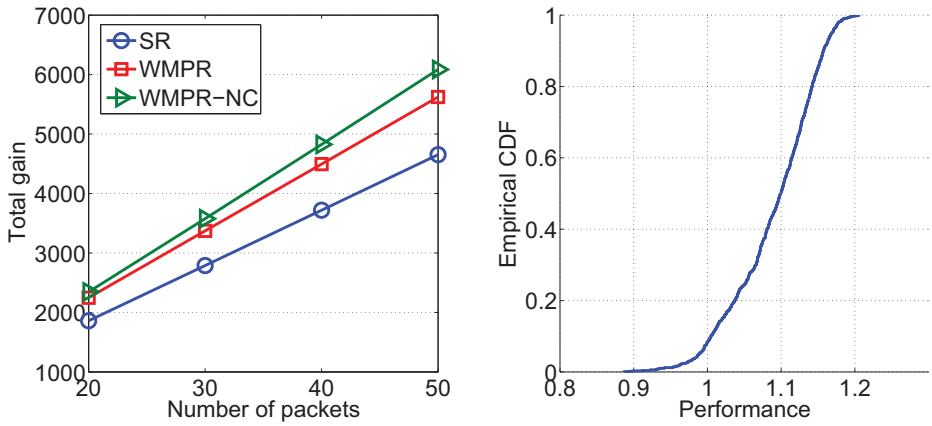


Figure 6.14: Comparison between the gain of WMPR, WMPR-NC, and SR mechanisms, $m = 5$, $n = 5$; (a) $p = 0.4$, $t = 5$ (b) $k = 50$.

decreases in the case of multiple destinations, which is because of the diversity of the links. Consequently, the efficiency of WMPR-NC increases in the case that the error rates of the links are close to each other.

We compare the performance of the WMPR-NC mechanism to the WMPR in Figure 6.15 (b). For this purpose, we divide the gain of the WMPR-NC mechanism by that of the WMPR mechanism, and plot its CDF. In this experiment, the packet size and the number of packets are equal to 5 symbols and 50, respectively. Also, the error rate of the links between the source and the 5 destination nodes are in the range of [0.3,0.5]. This figure shows that, in less than 5% of the cases, the number of delivered symbols in the WMPR-NC mechanism is less than that of the WMPR method. Moreover, in more than 50% of the cases, the number of delivered symbols of the WMPR-NC protocol is more than 10% higher than that of the WMPR mechanism.

Figure 6.16 (a) shows the gain of the WMPR, WMPR-NC, and WMPR-header. We set the header cost of the coded and uncoded packets to 0.07 and 0.05, respectively. The number of packets and symbols in each packet are equal to 20 and 5. Moreover,



(a) (b)

Figure 6.15: Comparison between the gain of WMPR, WMPR-NC, and SR mechanisms, $m = 5$, $n = 5$, $p \in [0.3, 0.5]$; (a) total gain, $t = 5$ (b) Performance of the WMPR-NC mechanism over the WMPR method.

the size of the transmission time window for each packet is set to 5. Figure 6.16 (a) shows that, for more reliable links, performing network coding might not be efficient, as the gain of WMPR-NC is less than that of the WMPR method. The reason is that, for these cases, the advantage of performing network coding over uncoding is less than the increase in the overhead. The WMPR-header considers the header overhead of the packets; as a result, it disables network coding when it finds that coding is not efficient. As we increase the error rate of the links, the difference between the utility of network coding and uncoding increases. Therefore, the utility of WMPR-NC becomes more than that of the WMPR method, and the WMPR-header method automatically switches to coding.

We increase the header cost of the coded packet c^{NC} to 0.09, and repeat the previous experiment. Figure 6.16 (b) shows the simulation result. Increasing c^{NC} reduces the utility of network coding; thus, the WMPR-header method turns on coding in the case of less reliable links. Note that in this simulation the number of packets is fixed, and an alternative way to make network coding more efficient is increasing the number of packets k, as shown in Figure 6.15 (a).

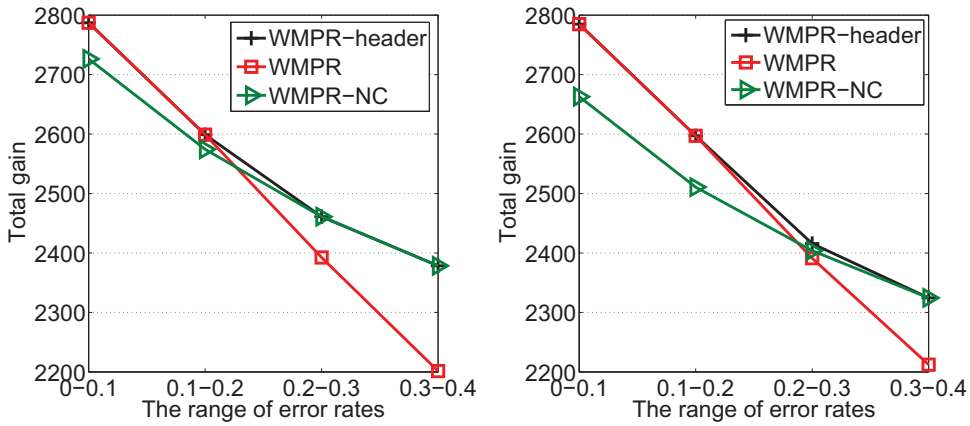


Figure 6.16: Comparison between the gain of WMPR, WMPR-NC, and WMPR-header mechanisms, $m = 5$, $n = 5$, $k = 20$, $t = 5$; (a) $c^{NC} = 0.07$, $c^{UC} = 0.05$ (b) $c^{NC} = 0.09$, $c^{UC} = 0.05$.

6.7 Conclusion

There is much work on reliable transmissions over error-prone wireless channels. In contrast to the previous work on reliable transmission, we consider a novel problem in this chapter. We study the problem of maximizing the total gain in the case of partial data delivery in error-prone wireless networks. In our setting, each set of bits, called symbols, has a different weight. We first address the case of single packet transmission to a homogenous destination nodes, and we show that the optimal solution of this problem has a round-robin pattern. Then, we extend our solution to the case of heterogenous destinations. We also provide a solution for the case of sending multiple packets to multiple destinations, and we enhance the expected gain (utility) using inter-packet random linear network coding.

Our extensive results show that our proposed multiple packets transmission mechanism can increase the gain up to 60%, compared to that of a simple retransmission mechanism. Moreover, using random linear network coding can enhance the gain.

CHAPTER 7

IMPLEMENTATION OF SYMBOL-LEVEL NETWORK CODING

7.1 System Model and Problem Definition

In Chapter 6, we discussed our symbol-level network coding. After proposing the idea of symbol-level network coding, we performed a real experiment using USRP (universal software radio peripheral) devices, and we found that the errors in real environments have a bursty pattern. It means that, if a transmission error happens, likely it will affect more than one symbol. Based on this observation, we extended our symbol-level network coding by considering different possible symbol placements in the packets. The results of our finding are presented in chapter.

7.1.1 System Setting

In our model, we have a single-hop wireless network, in which a source node broadcasts a batch of k packets to n destination nodes d. We assume that m symbols form a packet, and the symbols themselves might contain several bits. The priority of a symbol s_i is defined as the inverse of its weight w_i, and in general $w_i > w_{i+1}, \forall i : 1 \leq i \leq m-1$. The system model is similar to that in Chapter 6. The set of symbols used in this chapter is summarized in Table 7.1.

We assume that each batch of packets has a deadline to be transmitted, and after that the source has another batch of packets ready to be transmitted. Consequently, we cannot use channel coding and hierarchical coding methods in our setting. The

Table 7.1: The set of symbols used in this chapter.

Notation	**Definition**
d_i	The i-th destination node
n	The number of destination nodes
m	The number of symbols inside each packet
k	The number of packets
w_i	The weight of the i-th symbol of each packet
ϵ_i	The error rate of the link between the source and the i-th destination node.
t	The size of the transmission time window (in the terms of the number of symbols)
$s_{j,i}$	The i-th symbol of the j-th packet
S_i	The i-th coded symbol
$x_{j,i}$	The number of transmissions of the coded symbols S_i
y_i	The number of transmissions of the coded i-th symbols S_i
p_i	The i-th packet
u_i	The gain (utility) from the i-th symbols in the case of non-coded symbols
u_i^{NC}	The gain (utility) from the i-th symbols when we use linear NC
u	The total gain (utility) in the case of non-coding
u^{NC}	The total gain (utility) of using linear NC

time assigned for transmitting a single packet is sufficient for t symbol transmissions. As a result, for a batch of k packets, the source node can transmit $t \times k$ symbols. If we do not consider a deadline for the packets, or assume that the source has infinite packets to transmit, the optimal solution can be found by way of a simple extension of the well-known channel coding theory [78].

Our goal in this work is to maximize the total weight of the received symbols of a batch of k packets by the destination nodes. We use boolean variable $z_{j,i,l}$ to represent success and failure in delivering symbol s_i of the k-th packet to destination d_l. The value of the variable is equal to 1 in the case of successful delivery; otherwise the value is 0. We can model our objective function as the following utility function:

$$u = \sum_{j=1}^{k} \sum_{i=1}^{m} \sum_{l=1}^{n} z_{j,i,l} \times w_i$$

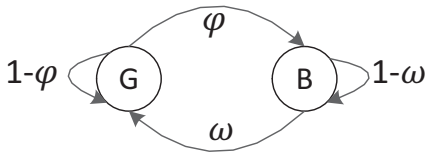
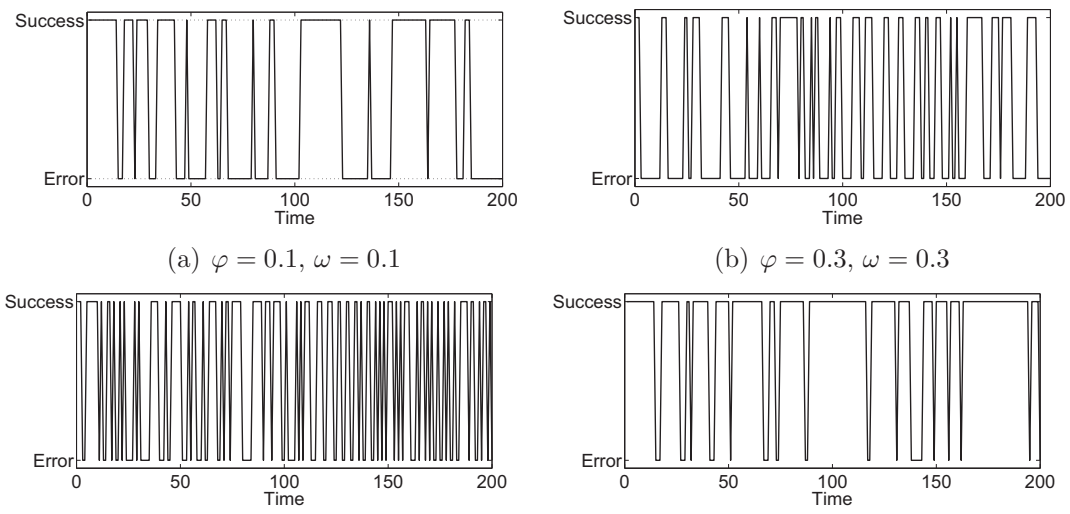


Figure 7.1: Gilbert-Elliott model: modeling transmission errors using a 2-state Markov chain.

7.1.2 Error Model

We assume that the transmission errors are bursty. A burst error is a contiguous sequence of errors. In order to model the burst errors we use the Gilbert-Elliott model [79, 80]. The Gilbert-Elliott model is widely used for describing burst error patterns in data transmission channels. This model is based on a Markov chain with two states *good* and *bad*, represented as G and B, respectively. As shown in Figure 7.1, the system transits from state G to B with a probability equal to φ. The probability of the transition from state B to G is equal to ω. In the general form of the model, there are different error probabilities for the 2 states. However, in the simplified model, the error probability of states G and B are supposed to be 0 and 1, respectively. In the simplified version, the error probability is equal to the stationary state probability of being at state B, which is equal to $\epsilon = \varphi/(\varphi + \omega)$.

Figures 7.2(a)-(d) show the simulation results of 200 transmissions using the simplified version of the Gilbert-Elliott model. In Figure 7.2(a), φ and ω are set to 0.1. As a result, the average length of the burst errors and successful transmissions are the same. Figures 7.2(b) and (c) show that the average length of the burst errors and successful transmissions decrease as we increase φ and ω, which is due to the increase in the transition probability. In Figure 7.2(d), φ and ω are equal to 0.1 and 0.5, respectively. As a result, the lengths of consecutive successful transmissions are more than those of the burst errors.



(a) $\varphi = 0.1$, $\omega = 0.1$

(b) $\varphi = 0.3$, $\omega = 0.3$

(c) $\varphi = 0.5$, $\omega = 0.5$

(d) $\varphi = 0.1$, $\omega = 0.5$

Figure 7.2: Simulation result for 200 transmissions using the Gilbert-Elliott model.

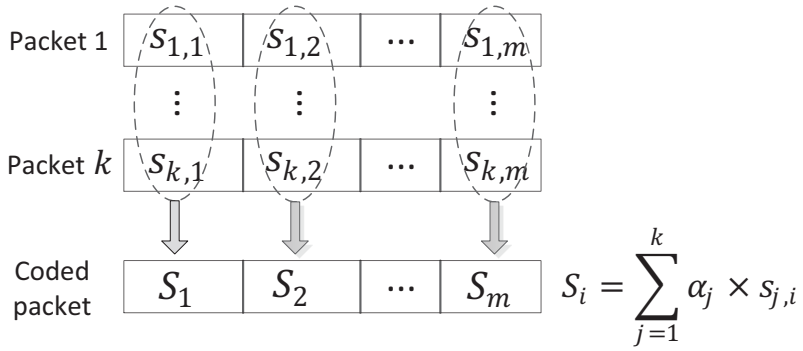


Figure 7.3: Network coding scheme.

7.2 Priority-Based Data Transmission

Without network coding, the probability of symbol $s_{j,i}$ to be received by the l-th destination is equal to $1 - {\epsilon_l}^{x_{j,i}}$, where $x_{j,i}$ is the number of transmissions of symbol $s_{j,i}$. Also, the weight of the i-th symbol of the packets are equal. As a result, in the case of non-coding, the total gain (utility) becomes:

$$u = \sum_{i=1}^{m} \sum_{j=1}^{k} \sum_{l=1}^{n} w_i \times (1 - {\epsilon_l}^{x_{j,i}})$$

The problem of this scheme is that a destination node might receive some of the symbols multiple times, and might not receive the other symbols. This problem is known as the coupon collector problem. In order to solve this problem, we can benefit from random linear NC. We can code the i-th symbols of the k packets together as shown in Figure 7.3. Using this scheme, a destination node will be able to decode the i-th coded symbols and retrieve the original symbols once it receives at least k coded symbols. Thus, each coded symbol contributes the same amount of information to the destination nodes. This is in contrast to the case of non-coding transmissions, in which a destination node might not receive some of the symbols, and might receive the other symbols multiple times. If we represent the number of transmissions assigned to the coded symbols over the i-th symbols of the packets as y_i, the utility becomes:

$$u^{NC} = \sum_{i=1}^{m} w_i \times k \sum_{l=1}^{n} \Big[\sum_{j=k}^{y_i \times k} \binom{k \times y_i}{j} \times (1-\epsilon_l)^j \times \epsilon_l^{y_i \times k - j} \Big]$$

The right most summation calculates the decoding probability of the i-th symbols. A destination node can decode the i-th code symbols if it receives at least k coded symbols. Random linear NC decreases the probability of receiving partial symbols. This is because if a destination node receives less than k coded symbols, it will not be able to decode the coded symbols. As a result, depending on the assigned redundancy to each set of symbols of the k packets, using random linear NC might be efficient or not.

Consider a source node that wants to transmit two packets with a single symbol to a destination node, and the error rate of the link between these nodes is equal to 0.4. Assume that the number of transmissions is equal to 4. In the case of non-coding transmissions, the probability of receiving both of the symbols is equal to $(1-0.4^2) \times (1-0.4^2) = 0.7056$. Also, the probability of receiving just one of them is equal to $2 \times (1-0.4^2) \times 0.4^2 = 0.2688$. Therefore, assuming that the weight of the

symbols is equal to 1, the total gain is $0.7056 + 0.2688 = 0.9744$. Using NC, these probabilities become $1 - 0.4^4 - 4 \times 0.4^3 = 0.7184$ and 0 respectively. Thus, using network coding in this example in not beneficial.

Following the above discussion, it is clear that none of the coding and non-coding schemes give us the optimal solution. Thus, we need a method that switches between them to finds the optimal scheme. The optimization in this case becomes:

$$\begin{aligned} & \max \sum_{i=1}^{m} max(u_i, u_i^{NC}) \\ s.t. \quad & \sum_{j=1}^{k} \sum_{i=1}^{m} x_{j,i} = t \\ & \sum_{i=1}^{m} y_i = t \end{aligned}$$

where:

$$u_i = \sum_{j=1}^{k} \sum_{l=1}^{n} w_i \times (1 - {\epsilon_l}^{x_{j,i}}) \tag{7.1}$$

$$u_i^{NC} = w_i \times k \sum_{l=1}^{n} \Big[\sum_{j=k}^{y_i \times k} \binom{k \times y_i}{j} (1 - \epsilon_l)^j \times \epsilon_l^{y_i \times k - j} \Big] \tag{7.2}$$

A straightforward solution is to apply brute search. For this purpose, we can find the optimal solution by calculating the gain in the case of coding and non-coding for all of the possible transmission distributions, and by selecting the distribution that results to the maximum gain. However, it is clear that the time complexity of the brute search is huge. In order to reduce the time complexity, we can benefit from the following lemma:

Lemma 7.1. *For the distribution that results in the optimal solution we have* $x_i \geq s_{i+1}, \forall 1 \leq i \leq m-1$

Proof. Both of the functions 7.1 and 7.2 are non decreasing functions. Therefore,

assigning more transmissions to the symbols with a greater w_i results in a higher utility. □

Lemma 7.1 helps us to reduce the search space dramatically.

Theorem 7.2. *The number of ways that we can distribute t transmissions among m symbols using lemma 7.1 is in the order of t^m.*

Proof. The number of ways that we can distribute t transmissions among m symbols using lemma 7.1 is equal to the number of ways to partition the integer number t to at most m partitions. The latter problem is referred to as the restricted version of the *integer partitioning* problem. In [81], it is shown that the number of distinct possible partitioning is in the order of t^m. □

Theorem 7.2 shows that the time complexity of checking all the possible transmission distributions that see lemma 7.1 is exponential. However, in our problem, m is fixed. Moreover, a typical number of symbols is usually a small number, e.g. 4 or 5. As a result, the number of possible distributions that need to be checked is polynomial in terms of t.

Based on the discussion, we propose the Priority-Based Transmission (PBT) algorithm as follows. The PBT algorithm checks all the possible distributions that see lemma 7.1. For each possible distribution, we check the gain of the i-th symbols of the k packets in the case of coding and non-coding symbols, and calculate the maximum total possible gain for all of the symbols. We repeat this process for all of the possible transmission distributions and select the distribution that results in the maximum total utility. The details of the PBT method are shown in Algorithm 13.

After finding the optimal number of transmissions for each symbol, the source transmits each symbol several times. As mentioned in Section 7.1, the errors in our model have a burst pattern. Consequently, putting the transmissions of the same symbol beside each other is not logical. The reason is that, if an error happens in a

Algorithm 13 PBT Algorithm

```
1: Initialize: maxGain = 0, argmax = 1
2: for i = 1 : m do
3:    coding(i) = 0
4: for each distribution do
5:    for i=1:m do
6:       calculate u_i^NC and u_i using 7.2 and 7.1
7:       if u_i^NC ≥ u_i then
8:          gain = gain + u_i^NC
9:       else
10:         gain = gain + u_i
11:   if gain ≥ maxGain then
12:      maxGain = gain
13:      argmax = index
14:      for i=1:m do
15:         coding(i)=1
```

transmission, there is a large chance that the next transmission faces an error as well. On the other hand, a successful transmission might be followed by another successful transmission with a high probability. Consequently, if the source node transmits a symbol multiple times, each following the other, it is with a high probability that a destination node might receive the symbol multiple times or might miss all of the transmissions assigned to that symbol. In the former case, the transmissions are wasted, as some of these successful transmissions could be assigned to other symbols. In the later case, the destination node will not receive and gain from that particular symbol.

In order to make the PBT method robust against burst errors, we transmit the different symbols in a round-robin patter. In PBT, the source node starts from the most important symbol of the packets, transmits each symbol of a packet once, and subtracts one transmission from the number of transmissions assigned to each symbol. It then repeats the process until no more transmissions are left. Our results from the implementation on the USRP testbed confirm the effectiveness of the round-robin transmission pattern.

7.3 Experiments and Evaluation

7.3.1 Simulations

7.3.1.1 Setting

In order to evaluate our method, we implemented a simulator in the MATLAB environment. We compare our proposed PBT (priority-based transmission) method with a *simple retransmission* (SR) method and MPT-NC method [82]. In the SR method, we distribute the transmissions evenly to the different symbols. The MPT-NC method, finds the optimal distribution of the transmissions in the case of non-coding. It then checks whether applying network coding among the symbols that have the same weight can increase the gain or not. In the former case, the MPT-NC method enables coding for these set of symbols. We run each simulation for 100 random topologies with different links' error rates. Also, we repeat the simulation of each random topology 100 times. In our simulations, the weight of the i-th symbol of each packet is equal to 2^{m-i}. In the simulations, the number of destinations, packets, and symbols of each packet are equal to 10 and 5, and 5, respectively.

7.3.1.2 Results

In the first experiment, we measure the effect that the number of transmissions has on the total gain in Figure 7.4(a). For each random topology, ω is set to 0.12. Moreover, φ for each link to the destinations is randomly set in the range of $[0.05, 0.12]$. As a result, the error rate of the links are in the range of $[0.2941, 0.5]$. As expected, the total gain in Figure 7.4(a) increases as we increase the number of transmissions. In this figure, the total gain of the PBT method is up to 16% and 50% more than that of the MPT-NC and SR methods, respectively.

In Figure 7.4(b), we reduce the average burst error size by half, and repeat the previous experiment. For this purpose, we multiply ω and the range of φ by 2.

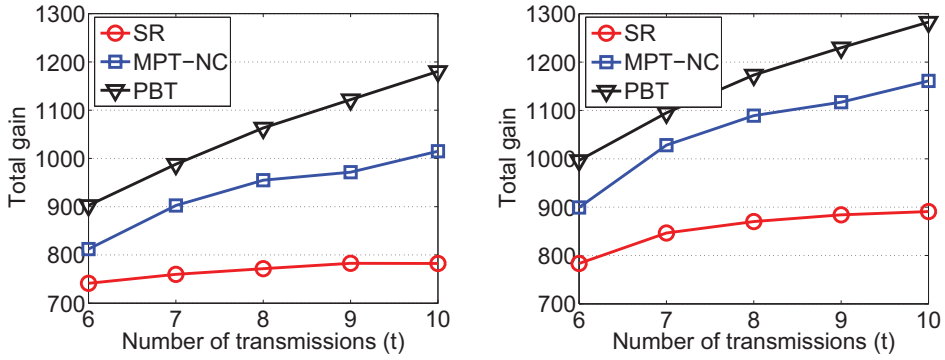


Figure 7.4: Effect of number of transmissions on the total gain . $m = 5$, $k = 5$, $n = 10$. (a) $r = 0.12$, $q \in [0.05, 0.12]$. (b) $r = 0.24$, $q \in [1, 0.24]$.

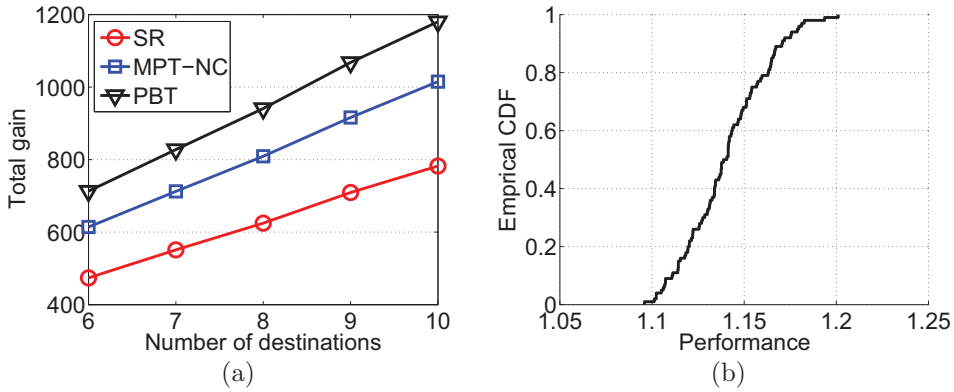


Figure 7.5: $r = 0.12$, $q \in [0.05, 0.12]$, $m = 5$, $k = 5$, $n = 10$. (a) Total gain. (b) Empirical CDF of the performance, $n = 10$.

Therefore, the range of error rates is still $[0.2941, 0.5]$. Although the error rates in Figures 7.4(a) and (b) are the same, the total gain in Figure 7.4(b) is more than that of in Figure 7.4(a). The reason is that, the size of burst errors in Figure 7.4(b) is smaller that in Figure 7.4(a). However, the reduction in the burst size has more of an effect on the gain of MPT-NC and SR methods, as the PBT method is more robust against the burst errors.

In Figure 7.5(a), ω is set to 0.12, and φ is in the range of $[0.05, 0.12]$. Also, the number of symbol transmissions t is equal to 10. The figure shows that the total gain of the methods is almost linear with respect to the number of destination nodes.

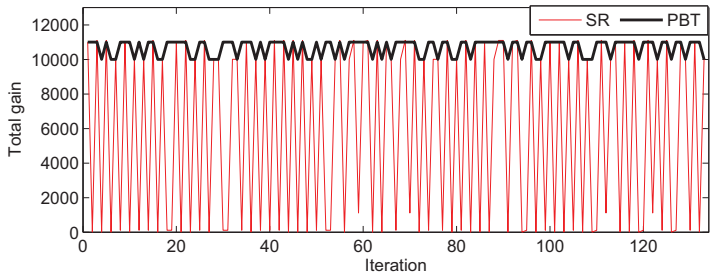
Figure 7.5(b) shows the empirical CDF of our method's performance. We define the performance as the division of the total gain of the PBT method by that of the MPT-NC scheme. The figure depicts that in 20% of the cases, the performance of our method is between 1.16 and 1.2. Moreover, in 50% of the runs, the performance of the PBT method is up to 1.14.

7.3.2 Real Experiment

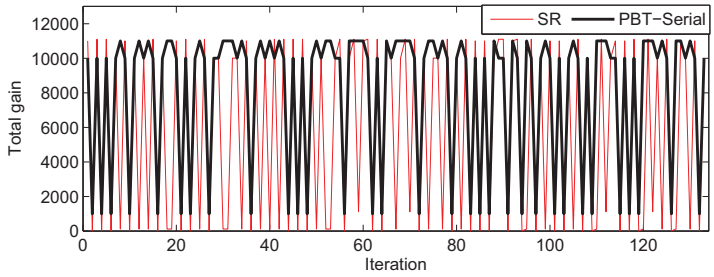
For the real experiments we use 3 USRPs (universal software radio peripheral) to evaluate our proposed PBT method. One USRP is the sender, one is the receiver, and the other one works as an interfering node. The devices work on the narrowband, and the central frequency is 1.26GHz. The antenna gain on each node is 20 db. We send the packets for one minute and compare the received packets using the PBT and the SR approaches.

The source node transmits a 5-digit binary coded decimal (BCD) number, in which the weight of the i-th digit (from left to right) is equal to $10^{5-i}, \forall\ 1 \leq i \leq 5$. We repeat the same experiment several times (iterations). Figure 7.6(a) shows the total gain of the PBT and SR methods for 134 runs. The bold and thin lines show the total gain of the PBT and simple retransmission, respectively. In the PBT method, we put the symbols in a round-robin fashion. In other words, we first put the symbols to be sent beside each other, and then we put the redundant symbols. As depicted in Figure 7.6(a), the total gain of the PBT method is almost always more than or equal to that of the SR. The oscillation in the gains is because of the randomness of the channel, which results in errors happening in different parts of the data in different runs.

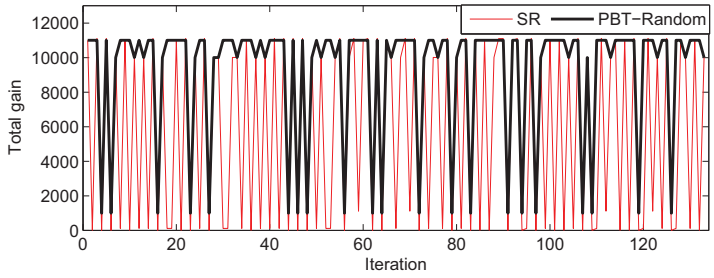
In Figure 7.6(b), we modify the proposed PBT method by changing the round robin patter with a serial pattern. It means that, we put symbol s_1 for x_1 times beside each other. Then, we put the second symbol x_2 times and so on. As Figure 7.6(b)



(a)



(b)



(c)

Figure 7.6: Real experiment. Transmission of a 4-digit BCD number. $k = 1$, $t = 5$, $\epsilon = 0.2442$. (a) PBT method. (b) PBT with serial symbols pattern. (b) PBT with random symbols pattern.

depicts, there are cases in which the gain of the SR method is more than that of the PBT method, which is due to the bursty errors. If a bursty error happens, all of the repeated serial symbols might be lost. On the other hand, if a symbol is received correctly, then the destination might receive that symbol several times, which is useless. It suggests that repeating the symbols in a serial patter is not an appropriate approach.

The total gain of the methods in the case of a random symbol pattern is shown

in Figure 7.6(c). As the figure depicts, in some cases both of the methods results in a similar total gain. However, in many cases, the gain of the PBT method is more than that of the SR method. By comparing the Figures 7.6(a)-(c) we find that the gain of the PBT method with a random symbol patterns is in between that of the serial and round-robin pattern.

7.4 Conclusion

Providing reliable transmission in wireless communication networks is critical. In this chapter, we consider transmission of data with different levels of importance. Instead of ensuring the reception of all of the packets by the destination nodes, we are interested in maximizing the utility that the designation nodes will gain in the case of partial retrieval of data. We propose an optimal solution for assigning transmissions for different parts of the data to be transmitted, which benefits from network coding. In our priority-based transmission method, we consider the possible burst errors. We implemented our method on a USRP testbed. We evaluated our method both through simulations and the results from the real testbed.

CHAPTER 8

CONCLUSION

8.1 Summary of Contributions

In this dissertation, we investigated the applications of network coding in priority-based data transmission. This applications include efficient resource management and robust data transmissions. Our contributions can be summarized as follows:

- The advantage of network coding in multi-layer video transmission is investigated. We use triangular network coding for content distribution on cache nodes, and depict that triangular network coding helps us to use the limited storage of caches efficiently.

- We design a scalable video-on-demand system using helper nodes and network coding. The helper nodes, which are micro servers with limited storage and bandwidth, help us to reduce the load on the servers. On the other hand, we use the benefit of joint inter- and intra-layer network coding in our system to provide more video requests from the helper nodes and reduce the load on the server. We also propose an optimal distributed resource management algorithm.

- We study robust multi-layer video broadcast using network coding. We propose a 2-dimensional triangular network coding, and show the effectiveness of our proposed methods by applying our coding scheme on real video traces.

- We propose a symbol-level data transmission, in which different parts of the packets have different importance. Efficient unequal-error protection methods

are proposed that make the transmissions robust against transmission errors. We implement and evaluate our symbol-level network coding method on universal software radio peripheral (USRP) devices. We introduce a modified version of our symbol-level network coding method to make the transmissions robust against bursty errors.

8.2 Future Work

We proposed a scalable video-on-demand system using helper nodes and network coding. As shown in the examples, joint inter- and intra-layer network coding should result in less server load than the case of pure intra-layer network coding. However, our simulation results show that the server load in the cases of joint inter- and intra-layer network coding and pure intra-layer network coding are close. As a future work, we want to study the reason for this phenomena, and find the cases that joint inter- and intra-layer network coding can be useful.

In this dissertation, we studied the theoretical aspects of a scalable video-on-demand system. However, there should exists many challenges in the implementation phase. Investigating the implementation issues and showing the practicality of the proposed video-on-demand system can be an interesting research topic.

In order to evaluate our proposed 2-dimensional network coding, we have applied our schemes on real video sequences. However, we made artificial packet drops to simulate the transmission errors. A multicasting system that performs real transmissions needs to be implemented to address the challenges of a real system. In such a system, the transmitter and receivers should perform real-time coding and decoding, which might be a challenge.

An interesting application of network coding is in providing data secrecy. When random linear network coding is applied on a set of packets, likely the coded packets cannot be decoded until a sufficient number of coded packets are available. As a

result, network coding is a natural and low cost way to provide security. On the other hand, in order to provide reliable transmissions and data storages, more coded packets need to be transmitted or stored on the storages. Consequently, providing reliability and security are in contrast with each other. Performing a trade-off between security and reliability can be an interesting research topic, specially in the case of multi-layer videos.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] T. Ho, M. Médard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

[2] M. Shao, S. Dumitrescu, and X. Wu, "Layered multicast with inter-layer network coding for multimedia streaming," *IEEE Transactions on Multimedia*, vol. 13, no. 99, pp. 353–365, 2011.

[3] M. Kim, D. Lucani, X. Shi, F. Zhao, and M. Médard, "Network coding for multi-resolution multicast," in *IEEE INFOCOM*, 2010, pp. 1–9.

[4] H. Djandji, "An efficient hybrid arq protocol for point-to-multipoint communication and its throughput performance," *IEEE Transactions on Vehicular Technology*, vol. 48, no. 5, pp. 1688–1698, 1999.

[5] B. Zhao and M. Valenti, "The throughput of hybrid-ARQ protocols for the gaussian collision channel," *IEEE Transactions on Information Theory*, vol. 47, no. 5, pp. 1971–1988, 2001.

[6] L. Rizzo and L. Vicisano, "RMDP: an FEC-based reliable multicast protocol for wireless environments," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 2, no. 2, pp. 23–31, 1998.

[7] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 2, pp. 24–36, 1997.

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

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

[10] D. J. MacKay, "Fountain codes," *IEE Proceedings- Communications*, vol. 152, no. 6, pp. 1062–1068, 2005.

[11] P. Cataldi, M. Shatarski, M. Grangetto, and E. Magli, "Lt codes," in *IIH-MSP'06*, 2006, pp. 263–266.

[12] G. C.Clark and J. B. Cain, *Error-correction coding for digital communications*. Springer, 1981.

[13] S. Lin and D. J. Costello, *Error control coding: Fundamentals and Applications*. Prentice-hall Englewood Cliffs, NJ, 2004.

[14] W. Ryan and S. Lin, *Channel codes: classical and modern.* Cambridge University Press, 2009.

[15] S. L. Howard, C. Schlegel, and K. Iniewski, "Error control coding in low-power wireless sensor networks: When is ecc energy-efficient?" *EURASIP Journal on Wireless Communications and Networking*, vol. 17, no. 2, pp. 29–29, 2006.

[16] M. Vuran and I. F. Akyildiz, "Error control in wireless sensor networks: a cross layer analysis," *ACM Transactions on Networking, IEEE*, vol. 17, no. 4, pp. 1186–1199, 2009.

[17] T. Courtade and R. Wesel, "Optimal allocation of redundancy between packet-level erasure coding and physical-layer channel coding in fading channels," *IEEE Transactions on Communications*, vol. 59, no. 8, pp. 2101–2109, 2011.

[18] C. Berger, S. Zhou, Y. Wen, P. Willett, and K. Pattipati, "Optimizing joint erasure-and error-correction coding for wireless packet transmissions," *IEEE Transactions on Communications*, vol. 7, no. 11, pp. 4586–4595, 2008.

[19] G. Angelopoulos, A. Paidimarri, A. P. Chandrakasan, and M. Médard, "Experimental study of the interplay of channel and network coding in low power sensor applications," in *IEEE ICC*, 2013, pp. 5126–5130.

[20] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "Xors in the air: practical wireless network coding," in *ACM SIGCOMM*, 2006.

[21] D. Koutsonikolas, C. Wang, and Y. Hu, "CCACK: Efficient network coding based opportunistic routing through cumulative coded acknowledgments," in *IEEE INFOCOM*, 2010, pp. 1–9.

[22] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *ACM SIGCOMM*, 2007.

[23] P. Ostovari, J. Wu, and A. Khreishah, "Network coding techniques for wireless and sensor networks," in *The Art of Wireless Sensor Networks*, H. M. Ammari, Ed. Springer, 2013.

[24] ——, "Deadline-aware broadcasting in wireless networks with local network coding," in *IEEE ICNC*, Jan 2012.

[25] P. Ostovari, A. Khreishah, and J. Wu, "Deadline-aware broadcasting in wireless networks with network coding," in *IEEE GLOBECOM*, Dec 2012.

[26] A. Khreishah, I. Khalil, P. Ostovari, and J. Wu, "Flow-based xor network coding for lossy wireless networks," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 2321–2329, 2012.

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

[28] S. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[29] R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782– 795, Oct 2003.

[30] L. Lu, M. Xiao, M. Skoglund, L. Rasmussen, G. Wu, and S. Li, "Efficient network coding for wireless broadcasting," in *Wireless Communications and Networking Conference (WCNC)*, 2010, pp. 1–6.

[31] L. Lu, M. Xiao, and L. Rasmussen, "Relay-aided broadcasting with instantaneously decodable binary network codes," in *ICCCN*, 2011, pp. 1–5.

[32] D. Nguyen, T. Tran, T. Nguyen, and B. Bose, "Wireless broadcast using network coding," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 2, pp. 914–925, 2009.

[33] W. Fang, F. Liu, Z. Liu, L. Shu, and S. Nishio, "Reliable broadcast transmission in wireless networks based on network coding," in *Computer Communications Workshops (INFOCOM WKSHPS)*, 2011, pp. 555–559.

[34] S. Katti, D. Katabi, H. Balakrishnan, and M. Medard, "Symbol-level network coding for wireless mesh networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 401–412, 2008.

[35] M. Li, Z. Yang, and W. Lou, "Codeon: Cooperative popular content distribution for vehicular networks using symbol level network coding," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 1, pp. 223–235, 2011.

[36] Z. Yang, M. Li, and W. Lou, "CodePlay: Live multimedia streaming in vanets using symbol-level network coding," in *ICNP*, 2010, pp. 223–235.

[37] W. Xiumin, W. Jianping, and X. Yinlong, "Data dissemination in wireless sensor networks with network coding," *EURASIP Journal on Wireless Communications and Networking*, vol. 2010, 2010.

[38] Z. Yang, M. Li, and W. Lou, "R-code: Network coding based reliable broadcast in wireless mesh networks with unreliable links," in *IEEE GLOBECOM*, 2009.

[39] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the air: practical wireless network coding," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 243–25, 2006.

[40] C. Fragouli, J. Widmer, and J. L. Boudec, "Efficient broadcasting using network coding," *IEEE/ACM Transactions on Networking*, vol. 16, no. 2, pp. 450–463, Apr 2008.

[41] L. Li, R. Ramjee, M. Buddhikot, and S. Miller, "Network coding-based broadcast in mobile ad-hoc networks," in *IEEE INFOCOM*, May 2007.

[42] W. Lou and J. Wu, "On reducing broadcast redundancy in ad hoc wireless networks," *IEEE Transactions on Mobile Computing*, vol. 1, no. 2, pp. 111– 122, Apr-Jun 2002.

[43] S. Yang and J. Wu, "Efficient broadcasting using network coding and directional antennas in MANETs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 2, pp. 148–161, Feb 2010.

[44] J. Wang, C. Huang, and J. Li, "On ISP-friendly rate allocation for peer-assisted vod," in *ACM Multimedia*, 2008, pp. 279–288.

[45] C. Wu and B. Li, "On meeting P2P streaming bandwidth demand with limited supplies," in *SPIE MMCN*, 2008.

[46] J. Wang, C. Huang, and J. Li, "Challenges, design and analysis of a large-scale P2P-vod system," in *ACM SIGCOMM*, 2008, pp. 375–388.

[47] J. Wang and K. Ramchandran, "Enhancing peer-to-peer live multicast quality using helpers," in *IEEE ICIP*, 2008, pp. 2300–2303.

[48] H. Zhang, J. Wang, M. Chen, and K. Ramchandran, "Scaling peer-to-peer video-on-demand systems using helpers," in *IEEE ICIP*, 2009, pp. 3053–3056.

[49] S. Pawar, S. Rouayheb, H. Zhang, K. Lee, and K. Ramchandran, "Codes for a distributed caching based video-on-demand system," in *ACSSC*, 2011.

[50] H. Hao, M. Chen, A. Parekh, and K. Ramchandran, "A distributed multichannel demand-adaptive P2P VoD system with optimized caching and neighbor-selection," in *SPIE*, 2011.

[51] H. Ha and C. Yim, "Two-dimensional channel coding scheme for MCTF-based scalable video coding," *Y. Wang and T. Fang and L-P. Chau and K. Yap*, vol. 9, no. 1, pp. 37–45, 2007.

[52] ——, "Layer-weighted unequal error protection for scalable video coding extension of h. 264/AVC," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 2, pp. 736–744, 2008.

[53] H. Wang, S. Xiao, and C. Kuo, "Robust video multicast with joint network coding and video interleaving," *Journal of Visual Communication and Image Representation*, vol. 21, no. 2, pp. 77–88, 2010.

[54] ——, "Random linear network coding with ladder-shaped global coding matrix for robust video transmission," *Journal of Visual Communication and Image Representation*, vol. 22, no. 3, pp. 203–212, 2011.

[55] F. Soldo, A. Markopoulou, and A. L. Toledo, "On the performance of network coding in multi-resolution wireless video streaming," in *IEEE NetCod*, 2010, pp. 1–6.

[56] M. Halloush and H. Radha, "Practical network coding for scalable video in error prone networks," in *IEEE PCS*, 2009, pp. 2107–2114.

[57] D. Koutsonikolas, Y. Hu, C. Wang, M. Comer, and A. Mohamed, "Efficient online wifi delivery of layered-coding media using inter-layer network coding," in *IEEE ICDCS*, 2011, pp. 237–247.

[58] N. Golrezaei, K. Shanmugam, A. Dimakis, A. Molisch, and G. Caire, "Femto-caching: Wireless video content delivery through distributed caching helpers," *Arxiv preprint arXiv:1109.4179*, 2011.

[59] D. Wang, J. Liu, Q. Zhang, and F. Chen, "Cooperative coding and caching for streaming data in multihop wireless networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2010, no. 10, pp. 2:1–2:13, 2010.

[60] A. Finamore, M. Mellia, M. Munafò, R. Torres, and S. Rao, "Youtube everywhere: impact of device and infrastructure synergies on user experience," in *ACM IMC*, 2011, pp. 345–360.

[61] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *ACM CCR*, 1996, pp. 117–130.

[62] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h. 264/avc standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

[63] E. Magli, M. Wang, P. Frossard, and A. Markopoulou, "Network coding meets multimedia: A review," *IEEE Transactions on Multimedia*, vol. 15, no. 5, pp. 1195–1212, 2013.

[64] B. Li, Z. Wang, J. Liu, and W. Zhu, "Two decades of internet video streaming: A retrospective view," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 9, no. 1s, pp. 1–20, 2013.

[65] Y. He and L. Guan, "A new polynomial-time algorithm for linear programming," in *ACM STOC*, 1984, pp. 302–311.

[66] P. Ostovari, A. Khreishah, and J. Wu, "Cache content placement using triangular network coding," in *IEEE WCNC*, 2013, pp. 1375–1380.

[67] S. H. Friedberg, A. J. Insel, and L. E. Spence, *Linear Algebra*, 4th ed. Upper Saddle River, NJ (USA); Prentice Hall, 2003.

[68] E. Chong and S. Zak, *An Introduction to Optimization*. John Wiley & Sons, 2013.

[69] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Upper Saddle River, NJ (USA); Prentice Hall Inc., 1989.

[70] X. Lin and N. Shroff, "Utility maximization for communication networks with multipath routing," *IEEE Transactions on Automatic Control*, vol. 5, no. 51, pp. 766–781, 2006.

[71] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge Univ Press, 2004.

[72] A. Khreishah, C.-C. Wang, and N. B. Shroff, "Optimization based rate control for communication networks with inter-session network coding," in *IEEE INFOCOM*, 2008, pp. 81–85.

[73] M. E. J. Newman, "Power laws, Pareto distributions and Zipf's law," *Contemporary physics*, vol. 46, no. 5, pp. 323–351, 2005.

[74] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *IEEE ISIT*, June 2003.

[75] J. Reichel, H. Schwarz, and M. Wien, "Joint scalable video model jsvm-6," Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, Geneva, Switzerland, Tech. Rep., 2006.

[76] "Video sequences," GArizona State University, 2006. [Online]. Available: http://trace.eas.asu.edu/yuv/bus/

[77] "Video sequences," GArizona State University, 2006. [Online]. Available: http://trace.eas.asu.edu/yuv/akiyo/

[78] T. Cover and J. Thomas, *Elements of information theory*. Wiley-InterScience, 2006.

[79] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell system technical journal*, vol. 39, no. 5, pp. 1253–1265, 1960.

[80] E. Elliot, "Estimates of error rates for codes on burst-nise channels," *Bell system technical journal*, vol. 42, no. 5, pp. 1977–1997, 1963.

[81] H. Gupta, *Tables of partitions*. Cambridge University Press, 1958.

[82] P. Ostovari, J. Wu, and A. Khreishahand, "Efficient symbol-level transmission in error-prone wireless networks with network coding," in *IEEE WoWMoM*, 2013, pp. 1–9.

PUBLICATIONS LIST

Journal Papers

1. **Pouya Ostovari**, Jie Wu, Abdallah Khreishah, and Ness B. Shroff, "Scalable Video Streaming with Helper Nodes using Random Linear Network Coding," accepted to appear in *IEEE/ACM Transactions on Networking*.

2. **Pouya Ostovari** and Jie Wu, "Robust Wireless Transmission of Scalable Coded Videos using Inter-Layer Network Coding", accepted to appear in *Journal of Computer Networks*.

3. **Pouya Ostovari**, Abdallah Khreishah, and Jie Wu, "Broadcasting with Hard Deadlines in Wireless Multi-hop Networks Using Network Coding," *Wireless Communications and Mobile Computing*, Vol. 15, No. 5, pp. 983-999, 2015.

4. **Pouya Ostovari**, Jie Wu, and Abdallah Khreishah, "Symbol-Level Reliable Broadcasting of Sensitive Data in Error-Prone Wireless Networks," *Journal of Parallel and Distributed Computing* (JPDC), Vol. 74, No. 7, pp. 2673-2685, 2014.

5. Abdallah Khreishah, Jie Wu and **Pouya Ostovari**, "Flow-Based XOR Network Coding for Lossy Wireless Networks," *IEEE Transactions on Wireless Communications*, Vol.11, No.6, pp. 2321-2329, June 2012.

6. **Pouya Ostovari**, Abdallah Khreishah, and Jie Wu, "Trade-off between Redundancy and Feedbacks in Wireless Network Communication," *Ad Hoc & Sensor Wireless Networks*, Vol. 24, No. 1, pp. 21-47, 2015.

Conference Proceedings

7. **Pouya Ostovari** and Jie Wu "Reliable Broadcast with Joint Forward Error Correction and Erasure Codes in Wireless Communication Networks," *Proc. Of IEEE International Conference on Mobile Ad-hoc and Sensor Systems* (MASS), October 19-22, 2015.

8. **Pouya Ostovari**, Jie Wu, and Abdallah Khreishah, "Cooperative Mobile Internet Access with Opportunistic Scheduling," *Proc. Of International Symposium on Resilient Communication Systems* (ISRCS), August 18-20, 2015.

9. **Pouya Ostovari** and Jie Wu, "Reliable Videos Broadcast with Network Coding and Coordinated Multiple Access Points," *Proc. Of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks* (WoWMoM), June 14-17, 2015.

10. **Pouya Ostovari**, Abdallah Khreishah, and Jie Wu, "Robust Wireless Delivery of Scalable Videos using Inter-layer Network Coding," *Proc. Of IEEE Wireless Communications and Networking Conference* (WCNC), March 9-12, 2015.

11. **Pouya Ostovari**, Jie Wu, and Ying Dai, "Priority-Based Broadcasting of Sensitive Data in Error-Prone Wireless Networks." *Proc. of the 2nd International Symposium on Resilient Communication Systems* (ISRCS), August 19- 21, 2014.

12. **Pouya Ostovari**, Jie Wu, and Abdallah Khreishah, "The Benefits of Cooperation between the Cloud and Private Data Centers for Multi-rate Video Streaming," *Proc. of the he 23rd International Conference on Computer Communications and Networks* (ICCCN 2014), August 4-7, 2014.

13. **Pouya Ostovari**, Abdallah Khreishah, and Jie Wu, "Multi-Layer Video Streaming with Helper Nodes using Network Coding," *Proc. of the 10h IEEE International Conference on Mobile Ad-hoc and Sensor Systems* (MASS 2013), October 14-16, 2013.

14. **Pouya Ostovari**, Jie Wu, and Abdallah Khreishah, "Efficient Symbol-Level Transmission in Error-Prone Wireless Networks With Network Coding," *Proc. of the 14th International Symposium on a World of Wireless, Mobile and Multimedia Networks* (WoWMoM 2013), June 2013.

15. **Pouya Ostovari**, Abdallah Khreishah, and Jie Wu, "Cache Content Placement Using Triangular Network Coding," *Proc. of IEEE Wireless Communications and Networking Conference* (WCNC), April 7-10, 2013.

16. **Pouya Ostovari** and Jie Wu, "Throughput and Fairness Aware Dynamic Network Coding in Wireless Communication Networks," *Proc. of the 1st International Symposium on Resilient Communication Systems* (ISRCS), August 13-15, 2013.

17. **Pouya Ostovari**, Abdallah Khreishah, and Jie Wu, "Deadline-aware Broadcasting in Sensor Networks with Network Coding," *Proc. of the IEEE Global Telecommunications Conference* (GLOBECOM), December 2012.

18. **Pouya Ostovari**, Jie Wu, and Abdallah Khreishah, "Deadline-aware Broadcasting in Wireless Networks with Local Network Coding," *Proc. of the International Conference on Computing, Networking and Communications, Wireless Networks Symposium* (ICNC), January 30 - February 2, 2012.

19. Abdallah Khreishah, Jie Wu and **Pouya Ostovari**, "Flow Based XOR Network Coding for Lossy Wireless Networks," *Proc. of the IEEE Global Telecommunications Conference* (GLOBECOM), December 2011.

Book Chapter

20. **Pouya Ostovari**, Jie Wu, and Abdallah Khreishah, "Network Coding Techniques for Wireless and Sensor Networks," accepted to appear in The Art of Wireless Sensor Networks, Vol. 1, pp. 129-162, H. M. Ammari (ed), Springer, 2014.

Under Review

21. **Pouya Ostovari**, Jie Wu, and Abdallah Khreishah, "Cooperative Internet Access using Network Coding and Opportunistic Scheduling," submitted to the Journal of Computer Communications.

22. **Pouya Ostovari**, Jie Wu, and Abdallah Khreishah, "Efficient Online Collaborative Caching for Cellular Networks."

23. **Pouya Ostovari** and Jie Wu, "Network Coding Technique: Security Challenges and Applications."