WIRELESS NETWORK CODING: ANALYSIS, CONTROL MECHANISMS, AND INCENTIVE DESIGN

A Dissertation

by

YU-PIN HSU

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Alex Sprintson
Committee Members,	P. R. Kumar
	Krishna Narayanan
	Natarajan Gautam
Head of Department,	Chanan Singh

May 2014

Major Subject: Electrical Engineering

Copyright 2014 Yu-Pin Hsu

ABSTRACT

The access to information *anywhere* and *anytime* is becoming a necessity in our daily life. Wireless technologies are expected to provide ubiquitous access to information and to support a broad range of emerging applications, such as multimedia streaming and video conferencing. The need to support the explosive growth in wireless traffic requires new tools and techniques that maximize the spectrum efficiency, as well as minimize delays and power consumption.

This dissertation aims at novel approaches for the design and analysis of efficient and reliable wireless networks. We plan to propose efficient solutions that leverage user collaboration, peer-to-peer data exchange, and the novel technique of *network coding*. Network coding improves the performance of wireless networks by exploiting the broadcast nature of the wireless spectrum. The new techniques, however, pose significant challenges in terms of *control*, *scheduling*, and *mechanism design*. The proposed research will address these challenges by developing novel *network controllers*, *packet schedulers*, and *incentive mechanisms* that would encourage the clients to collaborate and contribute resources to the information transfer.

Our contributions can be broadly divided into three research thrusts: (1) stochastic network coding; (2) incentive mechanism design; (3) joint coding and scheduling design. In the first thrust we consider a single-relay network and propose an optimal controller for the stochastic setting as well as a universal controller for the on-line setting. We prove that there exist an optimal controller for the stochastic setting which is stationary, deterministic, and threshold type based on the queue length. For the on-line setting we present a randomized algorithm with the competitive ratio of $e/(e-1)$. In the second thrust, we propose incentive mechanisms for both centralized

and distributed settings. In the third thrust, we propose joint coding and scheduling algorithms for time-varying wireless networks.

The outcomes of our research have both theoretical and practical impact. We design and validate efficient algorithms, as well as provide insights on the fundamental properties of wireless networks. We believe these results are valuable for the industry as they are instrumental for the design and analysis of future wireless and cellular networks that are more efficient and robust.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Professor Alex Sprintson. Only his continued support, brilliant ideas and encouragements could make this work possible. I have enjoyed working with him. He was always open for questions and enthusiastic for discussions. Besides all his great supervision in this work, his patience and positive attitude help me find my interests. I greatly appreciate and would like to thank him in advance for all of his future advice.

I am also very thankful to my thesis committee members: Professor Alex Sprintson, Professor P. R. Kumar, Professor Krishna Narayanan, and Professor Natarajan Gautam. Their feedbacks and ideas helped me a lot on completing this dissertation. It was a great honor to have them as my PhD committee. I am also deeply grateful to Professor Sergiy Butenko for his suggestions at my final defense. I had also a great fortune to collaborate with Professor Srinivas Shakkottai, Professor I-Hong Hou, Professor Daren B. H. Cline, Professor Eytan H. Modiano at MIT, and Professor Lingjie Duan at SUTD.

During my five years of PhD studies at Texas A&M University, I have found the admirable friends and colleagues; thank you very much! Finally, I express my deepest gratitude to my beloved family. My achievements (if any) were only because of their great support. Specially, I would like to express my true love to Min Her. Min has been always around to give excitement and meaning to my life, and I cannot imagine where I would be without her love and support.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Communication systems and communication networks are vital for our day-to-day live. Wireless technologies provide ubiquitous access to networking services. Billions of people around the globe rely on wireless networks for voice and video conferencing, video streaming, and files downloading. It has been predicted that the amount of wireless traffic will keep increasing exponentially in the coming years.

Unfortunately, wireless networks are inherently less reliable than wired networks due to interference and fading. Furthermore, the limited usable spectrum and energy constrains pose significant challenges for network designers. Despite recent contributions from research initiatives spanning many years, several important issues, such as providing robust performance guarantees to wireless applications, remain unresolved.

The novel technique of *network coding* has a significant potential for improving the throughput and reliability of wireless networks by taking advantage of the broadcast nature of the wireless medium. To fully realize the benefits offered by the network coding technique, there is a need to address a broad range of challenges related to control, scheduling, mechanism design, and performance analysis. The goal of this dissertation is to address these challenges through developing novel network controllers, scheduling algorithms, and incentive mechanisms for a broad range of wireless network coding settings.

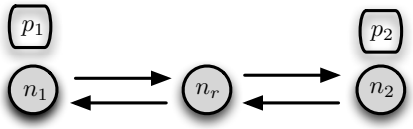

Figure 1.1: Nodes n_1 and n_2 exchange packets p_1 and p_2 via the relay node n_r.

The idea of network coding is to generalize the traditional routing algorithm by allowing the intermediate network nodes to create and forward the the new packets by combining the received packets. Different from the *store-and-forward* approach in the traditional routing algorithm, the network coding solution uses a *store-code-and-forward approach*. This technique has several important benefits such as an increase in network throughput, as well as an improvement in the reliability and robustness of networks. Fig. 1.1 illustrates a simple example, where two nodes n_1 and n_2 need to exchange packets p_1 and p_2 through a relay n_r. The traditional store-and-forward approach needs four transmissions, i.e., n_1 sends p_1 to n_r, n_2 sends p_2 to n_r, n_r sends p_1 to n_2, and n_r sends p_2 to n_1. In contrast, employing network coding, packets p_1 and p_2 are combined by means of a bit-wise XOR operation as $p_1 + p_2$ at n_r and are broadcast to nodes n_1 and n_2 simultaneously; as such, only three transmissions are required, i.e., n_1 sends p_1 to n_r, n_2 sends p_2 to n_r, and n_r sends $p_1 + p_2$ to n_1, n_2. Nodes n_1 and n_2 can decode the packets they need by combining the received coded packet with the packet they have, e.g., n_1 performs the bit-wise XOR $(p_1 + p_2) + p_1$ to recover the desired packet p_2. Consequently, by means of the network coding approach, the network throughput can be upgraded.

Since wireless transmissions are broadcast in nature, a wireless node can overhear the information from the nodes in its vicinity. For example, in Fig. 1.2, there is an information flow $n_1 \rightarrow n_2 \rightarrow n_3$. Node n_4 within the transmission range of n_1, n_2 can overhear the information sent by n_1 and n_2. Such information is referred to as *side information*. While the side information is not utilized by the traditional methods, the network coding schemes leverage the side information by broadcasting linear combinations of the packets so that multiple nodes can decode the information they need from a single transmission.

Fig. 1.3 depicts a scenario, in which the network coding decreases the transmis-

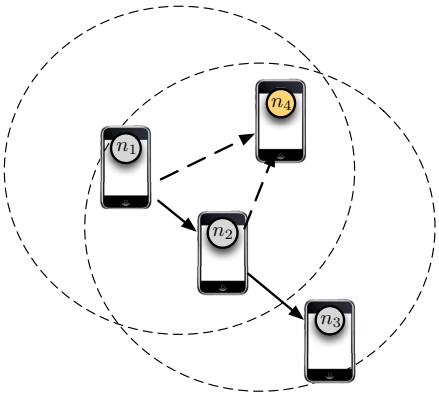

Figure 1.2: Information flow $n_1 \to n_2 \to n_3$, where n_4 lies in the transmission ranges of n_1 and n_2.

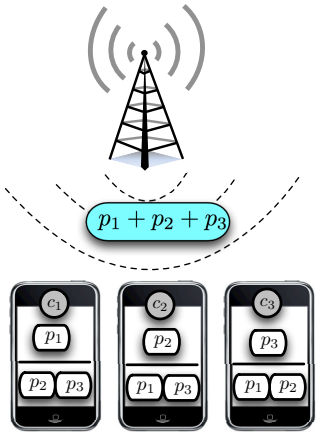

Figure 1.3: Server delivers p_1, p_2, p_3 to clients c_1, c_2, c_3 over a noiseless broadcast channel. Clients c_1, c_2, c_3 want packets p_1, p_2, p_3 and have the side information $\{p_2, p_3\}$, $\{p_1, p_3\}$, $\{p_1, p_2\}$, respectively.

sion counts; this scenario is an instance of the *Index Coding* problem [1, 2]. The server has three packets p_1, p_2, p_3 that needs to be transmitted to clients c_1, c_2, c_3 respectively over a noiseless broadcast channel. We assume that at most one packet can be sent for each time. Moreover, each of clients c_1, c_2, c_3 has a set of packets $\{p_2, p_3\}$, $\{p_1, p_3\}$, $\{p_1, p_2\}$, respectively, available to it as the side information. It can be verified that the demands of all clients can be satisfied by broadcasting one packet $p_1 + p_2 + p_3$. Note that the traditional approach without coding requires the transmissions of all three packets p_1, p_2, p_3.

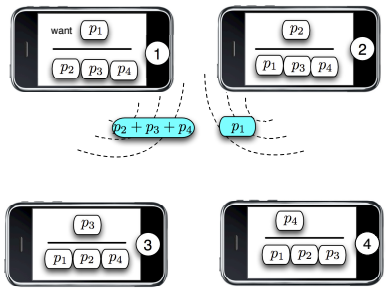

Figure 1.4: Clients c_1, c_2, c_3, c_4 exchange packets over a noiseless broadcast channel.

Moreover, in Fig. 1.4, we show the potential of network coding and side information in peer-to-peer (P2P) communications, which is an instance of the *Direct Data Exchange* problem [3–5]. Without network coding, at least four transmissions are necessary to satisfy the demands of all clients. Using the network coding technique, all clients can be satisfied by just two transmissions: client c_1 transmits a linear combination of p_2, p_3 p_4, and client c_2 transmit p_1.

Network coding research was initiated by the seminal work of Ahlswede et al. [6] and since then it has attracted major interest from the research community. Moreover, the network coding technique for wireless networks has been implemented

and demonstrated by MIT's COPE [7] and MORE [8]. COPE contains a special network coding layer between the IP and MAC layers. In MORE, an opportunistic routing protocol is proposed, that randomly mixes packets that belong to the same flow before forwarding them to the next hop.

In spite of recent advances for the wireless network coding, many important settings, such as *time-varying networks* and *selfish* network users, have received little attention from the research community. The mainstream network coding research has focused on *static* environments, where the communication channels are perfect, and all packets are given in advance. However, the wireless environment is intrinsically dynamic (or time-varying). The time variation occurs because the data information is randomly generated by the wireless users, the mobile users are moving when connecting to the network, channels are assigned dynamically (e.g., in *cognitive radio* networks), or radio signals reach the wireless users by more than one path (i.e., multipath effect). The traditional network coding schemes for the static network are not applicable in such environment; accordingly, there is a paramount need to develop new coding and scheduling approaches that are adaptive to the dynamic communication environment. The main question in this context is *how to design a network coding control scheme that maximizes the system performance in the presence of uncertainties, e.g., data loss, user mobility, and time-varying channel conditions.*

Moreover, network users will potentially be *selfish*; they only try to maximize the utilities of themselves, rather than the total social welfare of the overall system. However, the traditional network coding techniques are designed under the assumption that the users are cooperative, hence they are not applicable for the settings in which network users are selfish. Accordingly, there is an increasing need for the design of an *incentive design*, which provides an incentive for each network user to cooperate, in order to achieve the expected network performance. The central question in this

context is *how to develop an incentive mechanism that motivates the selfish users to participate in coding design and cooperative transfer of data.*

The control theory and game theory play a central role in designing robust wireless network coding schemes for time-varying networks and networks with selfish users. The control theory facilitates the design of a *centralized* controller that adaptively constructs the code and schedules the coded packets to be transmitted over the channel, while game theory is instrumental for designing efficient mechanisms for a *distributed* setting in the presence of selfish users. Accordingly, the overarching goal of this dissertation is to create an integrative framework that uses tools and techniques of control theory, game theory, and network coding theory to facilitate efficient information exchanges in wireless environments. Our research includes algorithm design, analysis of the optimality, complexity, and approximation ratio of the proposed solutions, as well as distributed implementation. Our goal is to provide algorithms with provable performance guarantees that are applicable to a broad range of practical settings.

Our contributions can be broadly divided into three research thrusts. First, in a stochastic environment some of the packets might not have coding pairs, which limits the number of available coding opportunities. In this context, an important decision is whether to delay the transmission of a packet in hope that a coding pair will be available in the future or transmit a packet without coding. We addresses this issue from the viewpoints of both stochastic optimization (average case analysis) and competitive analysis (worst case analysis). By formulating a stochastic dynamic program, we identify optimal control actions that would balance between costs of transmission against the costs incurred due to the delays. We also show that a stationary and deterministic threshold policy based on queue lengths is optimal.

The algorithm for stochastic dynamic optimization assumes an independent and

identical distributed arrival process. Therefore, we take a second look on this problem with the objective of designing a universal algorithm that works for any input distribution. We present an on-line algorithm based on the primal-dual approach. The performance of the proposed algorithm is analyzed using the competitive analysis, which characterize the performance of the algorithm in the worst-case scenario. We show that our algorithm achieves the competitive ratio of $e/(e-1)$.

Second, we focus on the scenarios in which clients are selfish. Our objective is to motivate selfish clients to contribute to information transfer and coding design. For both centralized communications and distributed P2P communications, we propose incentive-compatible network coding mechanisms that achieve the social optimum. In centralized communications, the classic Vickery-Clarke-Groves (VCG) mechanism can be applied but it is not tractable; hence we propose approximate incentive-compatible mechanisms that have provable performance guarantees. However, for P2P topologies, the VCG mechanism is no longer effective. Accordingly, we propose alternative distributed incentive-compatible mechanisms.

Third, we consider random arrivals to the server in centralized communications with ON/OFF time-varying channels. We assume that packets arrive to the server following a random arrival process. The problem can be viewed as a generalization of the Index Coding problem with lossy channels and random arrivals. We characterize the decodable capacity region, which captures the maximal arrival rates that can be handled by the system without compromising the stability of its arrival queues, as well as guarantee all clients can decode the packets they need. Specifically, we present an universal scheduling scheme that can work with any given coding scheme to handle all arrival rates within the decodable capacity region.

2. OPPORTUNITIES FOR NETWORK CODING: TO WAIT OR NOT TO WAIT*

2.1 Introduction

In recent years, there has been a growing interest in the applications of network coding techniques in wireless networks. It was shown that network coding can result in significant improvements in the performance in terms of delay and transmission count. For example, consider a wireless network coding scheme depicted in Fig. 2.1(a). Here, wireless nodes 1 and 2 need to exchange packets x_1 and x_2 through a relay node (node 3). A simple *store-and-forward* approach needs four transmissions. In contrast, the network coding solution uses a *store-code-and-forward* approach in which the two packets x_1 and x_2 are combined by means of a bitwise XOR operation at the relay and are broadcast to nodes 1 and 2 simultaneously. Nodes 1 and 2 can then decode this coded packet to obtain the packets they need.

Effros et al. [9] introduced the strategy of reverse carpooling that allows two information flows traveling in opposite directions to share a path. Fig. 2.1(b) shows an example of two connections, from n_1 to n_4 and from n_4 to n_1 that share a common path (n_1, n_2, n_3, n_4). The wireless network coding approach results in a significant (up to 50%) reduction in the number of transmissions for two connections that use reverse carpooling. In particular, once the first connection is established, the second connection (of the same rate) can be established in the opposite direction with little additional cost.

In this section, we will focus on the design and analysis of scheduling protocols

* Part of the data reported in this section is reprinted with permission from "Opportunities for Network Coding: To Wait or Not to Wait" by Yu-Pin Hsu, Navid Abedini, Natarajan Gautam, Alex Sprintson, and Srinivas Shakkottai, 2011. In *IEEE International Symposium on Information Theory Proceedings (ISIT)*, 791-795, Copyright 2011 by IEEE.

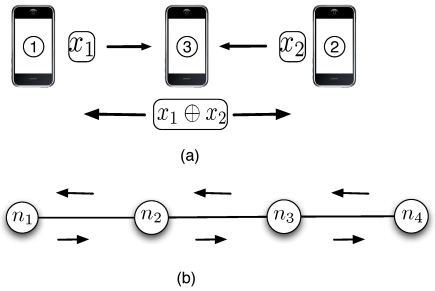

Figure 2.1: (a) Wireless network coding (b) Reverse carpooling.

that exploit the fundamental trade-off between the number of transmissions and delay in the reverse carpooling schemes. In particular, to cater to delay-sensitive applications, the network must be aware that savings achieved by coding may be offset by delays incurred in waiting for such opportunities. Accordingly, we design delay-aware controllers that use local information to decide whether or not to wait for a coding opportunity, or to go ahead with an uncoded transmission. By sending uncoded packets we do not take advantage of network coding, resulting in a penalty in terms of transmission count, and, as a result, energy-inefficiency. However, by waiting for a coding opportunity, we might be able to achieve energy efficiency at the cost of a small delay increase.

Consider a relay node that transmits packets between two of its adjacent nodes with flows in opposite directions, as depicted in Fig. 2.2. The relay maintains two queues q_1 and q_2, such that q_1 and q_2 store packets that need to be delivered to node 2 and node 1, respectively. If both queues are not empty, then it can relay two packets from both queues by performing an XOR operation. However, what should the relay do if one of the queues has packets to transmit, while the other queue is empty? Should the relay wait for a coding opportunity or just transmit a packet from a non-empty queue without coding? This is the fundamental question we seek

to answer. In essence we would like to trade off efficiently transmitting the packets against high quality of service (i.e., low delays).

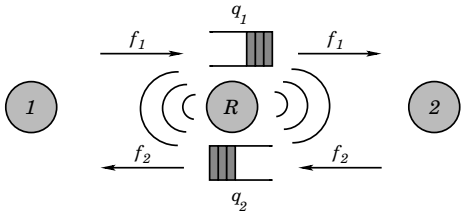

Figure 2.2: 3 nodes relay network.

2.1.1 Related work

Network coding research was initiated by the seminal work of Ahlswede et al. [6] and since then attracted major interest from the research community. Network coding technique for wireless networks has been considered by Katti et al. [7]. They propose an architecture, referred to as COPE, which contains a special network coding layer between the IP and MAC layers. In [8], an opportunistic routing protocol is proposed, referred to as MORE, that randomly mixes packets that belong to the same flow before forwarding them to the next hop. In addition, several works, e.g., [10–15], investigate the scheduling and/or routing problems in the network coding enabled networks. Sagduyu and Ephremides [10] focus on the network coding in the tandem networks and formulate related cross-layer optimization problems, while Khreishah et al. [11] devise a joint coding-scheduling-rate controller when the pairwise intersession network coding is allowed. Reddy et al. [12] have showed how to design coding-aware routing controllers that would maximize coding opportunities in multihop networks. References [13] and [14] attempt to schedule the network coding between multiple-session flows. Xi and Yeh [15] propose a distributed algorithm that

minimizes the transmission cost of a multicast session.

References [16–18] analyze the similar trade-off between power consumption and packet delays from different perspectives. Ciftcioglu et al. [16] propose a threshold policy using the Lyapunov technique. The threshold policy in [16] is an approximate solution with some performance guarantees. Nguyen and Yang [17] present a basic Markov decision process (MDP) framework for the problem at hand. Huang et al. [18] analyze the performance of the transport protocols over meshed networks as well as several implementation issues. In contrast, we focus on the detailed theoretical analysis of the problem at hand, present a provably optimal control policy, and identify its structure.

In this section, we consider a stochastic arrival process and address the decision problem of whether or not a packet should wait for a coding opportunity. Our objective is therefore to study the delicate trade-off between the energy consumption and the queueing delay when network coding is an option. We use the Markov decision process (MDP) framework to model this problem and formulate a stochastic dynamic program that determines the optimal control actions in various states. While there exists a large body of literature on the analysis of MDPs (see, e.g., [19–22]), there is no clear methodology to find optimal policies for the problems that possess the proprieties of infinite horizon, average cost optimization, and with a countably infinite state space. Indeed, [22] remarks that it is difficult to analyze and obtain optimal policies for such problems. The works in [23–26] contribute to the analysis of MDPs with countably infinite state space. Moreover, reference [27] that surveys the recent results on the monotonic structure of optimal policy, states that while one dimensional MDP with convex cost functions has been extensively studied, limited models for multi-dimensional spaces are dealt with due to the correlations between dimensions. In many high-dimension cases, one usually directly investigates the properties

of the cost function. As we will see later, this section poses precisely such a problem, and showing the properties of optimal solution is one of our main contributions.

2.1.2 Main results

We first consider the case illustrated in Fig. 2.2, in which we have a single relay node with two queues that contain packets traversing in opposite directions. We assume that time is slotted, and the relay can transmit at most one packet via noiseless broadcast channels during each time slot. We also assume that the arrivals into each queue are independent and identically distributed. Each transmission by the relay incurs a cost, and similarly, each time slot when a packet waits in the queue incurs a certain cost. Our goal is to minimize the weighted sum of the transmission and waiting costs.

We can think of the system state as the two queue lengths. We find that the optimal policy is a simple *queue-length threshold* policy with one threshold for each queue at the relay, and whose action is simply: if a coding opportunity exists, code and transmit; else transmit a packet if the threshold for that queue is reached. We then show how to find the optimal thresholds.

We examine three general models afterward. In the first model, the service capacity of the relay is not restricted to one packet per time slot. Then, if the relay can serve a batch of packets, we find that the optimal controller is of the threshold type for one queue, when the queue length of the other queue is fixed. Secondly, we study an arrival process with memory, i.e., *Markov modulated* arrival process. Here, we discover that the optimal policy has multiple thresholds. Finally, we extend our results for time-varying channels.

We then perform a numerical study of a number of policies that are based on waiting time and queue length, waiting time only, as well as the optimal deterministic

queue-length threshold policy to indicate the potential of our approach. We also evaluate the performance of a deterministic queue length based policy in the line network topology via simulations.

Our contributions can be summarized as follows. We consider the problem of delay versus coding efficiency trade-off, as well as formulate it as an MDP problem and obtain the structure of the optimal policy. It turns out that the optimal policy does not use the waiting time information. Moreover, we prove that the optimal policy is stationary and based on the queue-length threshold, and therefore is easy to implement. While it is easy to analyze MDPs that have a finite number of states, or involve a discounted total cost optimization with a single communicating class, our problem does not possess any of these properties. Hence, although our policy is simple, the proof is extremely intricate. Furthermore, our policy and proof techniques can be extended to other scenarios such as batched service and Markov-modulated arrival process.

2.2 System overview

2.2.1 System model

Our first focus is on the case of a single relay node of interest, which has the potential for network coding packets from flows in opposing directions. Consider Fig. 2.2 again. We assume that there is a flow f_1 that goes from node 1 to 2 and another flow f_2 from node 2 to 1, both of which are through the relay under consideration. The packets from both flows are stored at separate queues, q_1 and q_2, at relay node R.

For clarity of presentation, we assume a simple time division multiple access (TDMA) scheme, however or results are easy to generalize to more involved settings. We assume that time is divided into slots and each slot is further divided into three

mini-slots. In each slot, each node is allowed to transmit in its assigned mini-slot: node 1 uses the first mini-slot and node 2 uses the second mini-slot, while the last mini-slot in a slot is used by the relay. In particular, the time period between transmission opportunities for the relay is precisely one slot. Our model is consistent with the scheduled and time synchronized scheme such as LTE. Moreover, we use slot as the unit of packet delays. We assume if a packet is transmitted in the same slot when it arrived at the relay, its latency is zero.

The number of arrivals between consecutive slots to both flows is assumed to be independent of each other and also independent and identically distributed (i.i.d.) over time, with the random variables $\mathcal{A}_i$ for $i = 1, 2$ respectively. In each slot, n packets arrive at q_i with the probability $\mathbb{P}(\mathcal{A}_i = n) = p_n^{(i)}$ for $n \in \mathbb{N} \cup \{0\}$. Afterward, the relay gets an opportunity to transmit. Initially we assume that the relay can transmit a maximum of one packet in each time slot.

2.2.2 Markov decision process model

We use a Markov decision process (MDP) model to develop a strategy for the relay to decide its best course of action at every transmission opportunity. For $i = 1, 2$ and $t = 0, 1, 2, \cdots$, let $Q_t^{(i)}$ be the number of packets in q_i at the t^{th} time slot just before an opportunity to transmit. Let a_t be the action chosen at the end of the t^{th} time slot with $a_t = 0$ implying the action is to do nothing and $a_t = 1$ implying the action is to transmit. Clearly, if $Q_t^{(1)} + Q_t^{(2)} = 0$, then $a_t = 0$ because that is the only feasible action. Also, if $Q_t^{(1)} Q_t^{(2)} > 0$, then $a_t = 1$ because the best option is to transmit as a coded XOR packet as it reduces both the number of transmissions as well as latency. However, when exactly one of $Q_t^{(1)}$ and $Q_t^{(2)}$ is non-zero, it is unclear what the best action is.

To develop a strategy for that, we first define the costs for latency and transmis-

sion. Let C_T be the cost for transmitting a packet and C_H be the cost of holding a packet for a length of time equal to one slot. The power for transmitting a packet is much higher than the processing energy for network coding because of the simple XOR operation. We therefore ignore the effect of the processing cost. However, to include the processing cost is a small extension and will not change the analytical approach. Hence we assume that the cost of transmitting a coded packet is the same as that of a uncoded packet.

We define the MDP$\{(Q_t, a_t), t \geq 0\}$ where $Q_t = (Q_t^{(1)}, Q_t^{(2)})$ is the state of the system and a_t is the control action chosen by the relay at the t^{th} slot. The state space (i.e., all possible values of Q_t) is the set $\{(i, j) : i = 0, 1, \cdots ; j = 0, 1, \cdots\}$.

Let $C(Q_t, a_t)$ be the *immediate cost* if action a_t is taken at time t when the system is in state $Q_t = (Q_t^{(1)}, Q_t^{(2)})$. Therefore,

$$C(Q_t, a_t) = C_H([Q_t^{(1)} - a_t]^+ + [Q_t^{(2)} - a_t]^+) + C_T a_t, \tag{2.1}$$

where $[x]^+ = \max(x, 0)$.

2.2.3 Average-optimal policy

A *policy* θ specifies the decisions at all decision epoch, i.e., $\theta = \{a_0, a_1, \cdots\}$. A policy is *history dependent* if a_t depends on $a_0, \cdots a_{t-1}$ and $Q_0 \cdots, Q_t$, while that is *Markov* if a_t only depends on Q_t. A policy is *stationary* if $a_{t_1} = a_{t_2}$ when $Q_{t_1} = Q_{t_2}$ for some t_1, t_2. In general, a policy belongs to one of the following sets [19]:

- Π^{HR}: a set of randomized history dependent policies;
- Π^{MR}: a set of randomized Markov policies;
- Π^{SR}: a set of randomized stationary policies;

- Π^{SD}: a set of deterministic stationary policies.

The long-run average cost for some policy $\theta \in \Pi^{\mathrm{HR}}$ is given by

$$V(\theta) = \lim_{K\to\infty} \frac{1}{K+1}\mathbb{E}_\theta \left[\sum_{t=0}^{K} C(Q_t, a_t)|Q_0 = (0,0)\right], \tag{2.2}$$

where $\mathbb{E}_\theta$ is the expectation operator taken for the system under policy θ. We consider our initial state to be an empty system, since if we view our system as an ad-hoc network with some initial energy, then the initial state of all queue would be zero to begin with.

Our goal is to characterize and obtain the *average-optimal policy*, i.e., the policy that minimizes $V(\theta)$. It is not hard to see (as shown in [19]) that

$$\Pi^{\mathrm{SD}} \subset \Pi^{\mathrm{SR}} \subset \Pi^{\mathrm{MR}} \subset \Pi^{\mathrm{HR}}.$$

As in [19,21] there might not exist a SR or SD policy that is optimal, in what regime does the average-optimal policy lie?

We first describe the probability law for our MDP and then develop a methodology to obtain the average-optimal policy. For the MDP$\{(Q_t, a_t), t \geq 0\}$, let $P_{a_t}(Q_t, Q_{t+1})$ be the transition probability from state Q_t to Q_{t+1} associated with action $a_t \in \{0,1\}$. Then the probability law can be derived as $P_0\left((i,j),(k,l)\right) = p^{(1)}_{k-i}p^{(2)}_{l-j}$ for all $k \geq i$ and $l \geq j$; otherwise, $P_0\left((i,j),(k,l)\right) = 0$. Also, $P_1\left((i,j),(k,l)\right) = p^{(1)}_{k-[i-1]^+}p^{(2)}_{l-[j-1]^+}$ for all $k \geq [i-1]^+$ and $l \geq [j-1]^+$; otherwise, $P_1\left((i,j),(k,l)\right) = 0$.

A list of important notation used in this section is summarized in Table 2.1.

2.2.4 Waiting time information

Intuition tells us that if a packet has not been waiting for a long time then perhaps it could afford to wait a little more, but if a packet has waited for long, it might be

Table 2.1: Notation table

$\mathcal{A}_i$	Random variable that represents the number of packets that arrives at q_i for each time slot
$p_n^{(i)}$	Probability that n packets arrive at q_i, i.e., $\mathbb{P}(\mathcal{A}_i = n)$
$Q_t^{(i)}$	The number of packets in q_i at time t
Q_t	System state, i.e., $(Q_t^{(1)}, Q_t^{(2)})$
a_t	Action chosen by relay at time t
C_T	Cost of transmitting one packet
C_H	Cost of holding a packet for one time slot
$C(Q_t, a_t)$	Immediate cost if action a_t is taken at time t when the system is in state Q_t
$V(\theta)$	Time average cost under the policy θ
$P_{a_t}(Q_t, Q_{t+1})$	Transition probability from state Q_t to Q_{t+1} when action a_t is chosen
$V_\alpha(i, j, \theta)$	Total expected discounted cost under the policy θ when the initial state is (i, j)
$V_\alpha(i, j)$	Minimum total expected discounted cost when the initial state is (i, j), i.e., $\min_\theta V_\alpha(i, j, \theta)$
$v_\alpha(i, j)$	Difference of the minimum total expected discounted cost between the states (i, j) and $(0, 0)$, i.e., $V_\alpha(i, j) - V_\alpha(0, 0)$
$V_{\alpha,n}(i, j)$	Iterative definition for the optimality equation of $V_\alpha(i, j)$
$\mathcal{V}_\alpha(i, j, a)$	$V_\alpha(i, j) = \min_{a\in\{0,1\}} \mathcal{V}_\alpha(i, j, a)$, which is the optimality equation of $V_\alpha(i, j)$
$\Delta\mathcal{V}(i, j)$	$\mathcal{V}_\alpha(i, j, 1) - \mathcal{V}_\alpha(i, j, 0)$

better to just transmit it. That seems logical considering that we try our best to code but we cannot wait too long because it hurts in terms of holding costs. It is easy to keep track of waiting time information using time-stamps on packets when they are issued. Let $T^{(i)}$ be the arrival time of i^{th} packet and $\mathcal{D}_\theta^{(i)}$ be its delay (i.e., the waiting time before it is transmitted) while policy θ is applied. We also denote by $\mathcal{T}_{t,\theta}$ the number of transmissions by time t under policy θ. Then Eq. (2.2) can be written as

$$V(\theta) = \lim_{K\to\infty} \frac{1}{K+1} \mathbb{E}_\theta \left[\sum_{i:T^{(i)}\leq K} C_H \mathcal{D}_\theta^{(i)} + C_T \mathcal{T}_{K,\theta} \right]. \tag{2.3}$$

Would we be making better decisions by also keeping track of waiting times of each packet? We can answer this question by applying [19, Theorem 5.5.3].

Proposition 1.

(i) For the MDP$\{(Q_t, a_t), t \geq 0\}$, if there exists a randomized history dependent policy that is average-optimal then there exists a randomized Markov policy $\theta^* \in \Pi^{MR}$ that minimizes $V(\theta)$.

(ii) Further, one cannot find a policy which also uses waiting time information that would yield a better solution than $V(\theta^*)$.

2.2.5 Remark

To inform nodes 1 and 2 whether the transmitted packet is coded or not, we can just put one bit in front of each packet, where 0 for a uncoded packet and 1 for a coded packet. See [7] for more implementation issues.

In Subsections 2.3 and 2.4, we prove that there exists an optimal policy that is stationary, deterministic, and queue-length threshold for the system model of this subsection. The result will be generalized in Subsection 2.7.

- In Subsection 2.7.1, we consider the batched service, where more than one packet can be served for each time.
- In Subsection 2.7.2, instead of i.i.d. arrivals, we consider the Markov-modulated arrival process.
- In Subsection 2.7.3, we consider time-varying channels.

2.3 Structure of the average-optimal policy: stationary and deterministic property

In the previous subsection, we showed that there exists an average-optimal policy that does not include the waiting time in the state of the system. Next, we focus on queue length based and randomized Markov policies, as well as determine the structure of the average-optimal policy. In this subsection, we will show that there exists an average-optimal policy that is stationary and deterministic.

We begin by considering the infinite horizon α-discounted cost case, where $0 < \alpha < 1$, which we then tie to the average cost case. This method is typically used in the MDP literature (e.g., [26]), where the conditions for the structure of the average-optimal policy usually rely on the results of the infinite horizon α-discounted cost case. For our MDP$\{(Q_t, a_t), t \geq 0\}$, the total expected discounted cost incurred by a policy $\theta \in \Pi^{\mathrm{HR}}$ is

$$V_\alpha(i, j, \theta) = \mathbb{E}_\theta \left[\sum_{t=0}^{\infty} \alpha^t C(Q_t, a_t) | Q_0 = (i, j) \right]. \tag{2.4}$$

In addition, we define $V_\alpha(i, j) = \min_\theta V_\alpha(i, j, \theta)$ as well as $v_\alpha(i, j) = V_\alpha(i, j) - V_\alpha(0, 0)$. Define the α-*optimal policy* as the policy θ that minimizes $V_\alpha(i, j, \theta)$.

2.3.1 Preliminary results

In this subsection, we introduce the important properties of $V_\alpha(i,j)$, which are mostly based on the literature [26]. We first show that $V_\alpha(i,j)$ is finite (Proposition 2) and then introduce the *optimality equation* of $V_\alpha(i,j)$ (Lemma 3).

Proposition 2. *If $\mathbb{E}[\mathcal{A}_i] < \infty$ for $i = 1, 2$, then $V_\alpha(i,j) < \infty$ for every state (i,j) and α.*

Proof. Let $\tilde{\theta}$ be a stationary policy of waiting (i.e., $a_t = 0$ for all t) in each time slot. By definition of optimality, $V_\alpha(i,j) \leq V_\alpha(i,j,\tilde{\theta})$. Hence, if $V_\alpha(i,j,\tilde{\theta}) < \infty$, then $V_\alpha(i,j) < \infty$. Note that

$$\begin{aligned} V_\alpha(i,j,\tilde{\theta}) &= \mathbb{E}_{\tilde{\theta}}\Big[\sum_{t=0}^{\infty} \alpha^t C(Q_t, a_t) | Q_0 = (i,j)\Big] \\ &= \sum_{t=0}^{\infty} \alpha^t C_H \left(i + j + t\mathbb{E}[\mathcal{A}_1 + \mathcal{A}_2]\right) \\ &= \frac{C_H(i+j)}{1-\alpha} + \frac{\alpha C_H}{(1-\alpha)^2}\mathbb{E}[\mathcal{A}_1 + \mathcal{A}_2] < \infty. \end{aligned}$$

□

The next lemma follows from Propositions 1 in [26] and the fact that $V_\alpha(i,j)$ is finite (by Proposition 2).

Lemma 3 ([26], Proposition 1)**.** *If $\mathbb{E}[\mathcal{A}_i] < \infty$ for $i = 1, 2$, then the optimal expected discounted cost $V_\alpha(i,j)$ satisfies the following optimality equation:*

$$\begin{aligned} V_\alpha(i,j) = \min_{a \in \{0,1\}} [&C_H([i-a]^+ + [j-a]^+) + C_T a + \\ &\alpha \sum_{k=0}^{\infty}\sum_{l=0}^{\infty} P_a\big((i,j),(k,l)\big) V_\alpha(k,l)]. \end{aligned} \tag{2.5}$$

Moreover, the stationary policy that realizes the minimum of right hand side of (2.5) will be an α-optimal policy.

We define $V_{\alpha,0}(i,j) = 0$ and for $n \geq 0$,

$$V_{\alpha,n+1}(i,j) = \min_{a\in\{0,1\}} [C_H([i-a]^+ + [j-a]^+) + C_T a + \alpha \sum_{k=0}^{\infty}\sum_{l=0}^{\infty} P_a\big((i,j),(k,l)\big)V_{\alpha,n}(k,l)]. \tag{2.6}$$

Lemma 4 below follows from Proposition 3 in [26].

Lemma 4 ([26], Proposition 3)**.** *$V_{\alpha,n}(i,j) \to V_\alpha(i,j)$ as $n \to \infty$ for every i, j, and α.*

Eq. (2.6) will be helpful for identifying the properties of $V_\alpha(i,j)$, e.g., to prove that $V_\alpha(i,j)$ is a non-decreasing function.

Lemma 5. *$V_\alpha(i,j)$ is a non-decreasing function with respect to (w.r.t.) i for fixed j, and vice versa.*

Proof. The proof is by induction on n in Eq. (2.6). The result clearly holds for $V_{\alpha,0}(i,j)$. Now, assume that $V_{\alpha,n}(i,j)$ is non-decreasing. First, note that $C_H([i-a]^+ + [j-a]^+) + C_T a$ is a non-decreasing function of i and j (since C_H is non-negative). Next, we note that

$$\begin{aligned} &\alpha \sum_{k=0}^{\infty}\sum_{l=0}^{\infty} P_a\big((i,j),(k,l)\big)V_{\alpha,n}(k,l) \\ =&\alpha \sum_{r=0}^{\infty}\sum_{s=0}^{\infty} p_r^{(1)} p_s^{(2)} V_{\alpha,n}([i-a]^+ + r, [j-a]^+ + s), \end{aligned}$$

which is also a non-decreasing function in i and j separately due to the inductive assumption. Since the sum and the minimum (in Eq. (2.6)) of non-decreasing func-

tions are a non-decreasing function, we conclude that $V_{\alpha,n+1}(i,j)$ is a non-decreasing function as well. □

The next two lemmas, which can be proven via the similar arguments in [26], specify the conditions for the existence of the optimal stationary and deterministic policy.

Lemma 6 ([26], Theorem (i)). *There exists a stationary and deterministic policy that is average-optimal for the MDP*$\{(Q_t, a_t), t \geq 0\}$ *if the following conditions are satisfied:*

(i) $V_\alpha(i,j)$ *is finite for all* i, j, *and* α;

(ii) There exists a nonnegative N *such that* $v_\alpha(i,j) \geq -N$ *for all* i, j, *and* α;

(iii) There exists a nonnegative $M_{i,j}$ *such that* $v_\alpha(i,j) \leq M_{i,j}$ *for every* i, j, *and* α. *Moreover, for each state* (i,j) *there is an action* $a(i,j)$ *such that*

$$\sum_{k=0}^{\infty}\sum_{l=0}^{\infty} P_{a(i,j)}\big((i,j),(k,l)\big)M_{k,l} < \infty$$

.

Lemma 7 ([26], Proposition 5). *Assume there exists a stationary policy* θ *inducing an irreducible and ergodic Markov chain with the following properties: there exists a nonnegative function* $F(i,j)$ *and a finite nonempty subset* $G \subseteq (\mathbb{N} \cup \{0\})^2$ *such that for* $(i,j) \in (\mathbb{N} \cup \{0\})^2 - G$ *it holds that*

$$\sum_{k=0}^{\infty}\sum_{l=0}^{\infty} P_{a(\theta)}((i,j),(k,l))F(k,l) - F(i,j) \leq -C((i,j),a(\theta)), \tag{2.7}$$

where $a(\theta)$ is the action when the policy θ is applied. Moreover, for $(i,j) \in G$ it holds that

$$\sum_{k=0}^{\infty}\sum_{l=0}^{\infty} P_{a(\theta)}((i,j),(k,l))F(k,l) < \infty.$$

Then, the condition (iii) in Lemma 6 holds.

2.3.2 Main result

Using lemmas 6 and 7, we show next that the MDP defined in this section has an average-optimal policy that is stationary and deterministic.

Theorem 8. *For the MDP$\{(Q_t, a_t), t \geq 0\}$, there exists a stationary and deterministic policy θ^* that minimizes $V(\theta)$ if $\mathbb{E}[\mathcal{A}_i^2] < \infty$ and $\mathbb{E}[\mathcal{A}_i] < 1$ for $i = 1, 2$.*

Proof. As described earlier it is sufficient to show that the three conditions in Lemma 6 are satisfied. Proposition 2 implies that the condition (i) holds, while the condition (ii) is satisfied due to Lemma 5 (i.e., $N = 0$ in Lemma 6). We denote by $\tilde{\theta}$ the stationary policy of transmitting at each time slot. We use this policy for each of the three cases described below and show that condition (iii) of Lemma 6 holds.

Case (i): $p_0^{(i)} + p_1^{(i)} < 1$ for $i = 1, 2$, i.e., the probability that two or more packets arrive for each time slot is non-zero. This policy $\tilde{\theta}$ results in an irreducible and ergodic Markov chain, and therefore Lemma 7 can be applied. Let $F(i,j) = B(i^2 + j^2)$ for some positive B. Then, for all states $(i,j) \in (\mathbb{N} \cup \{0\})^2 - \{(0,0),(0,1),(1,0)\}$, it

holds that

$$
\begin{aligned}
&\sum_{k=0}^{\infty}\sum_{l=0}^{\infty} P_{a(\tilde{\theta})}\left((i,j),(k,l)\right)\left[F(k,l)-F(i,j)\right] \\
=&\sum_{r=0}^{\infty}\sum_{s=0}^{\infty} P_1\left((i,j),([i-1]^+ + r,[j-1]^+ + s)\right)\cdot\left[F([i-1]^+ + r,[j-1]^+ + s)-F(i,j)\right] \\
=&\sum_{r=0}^{\infty}\sum_{s=0}^{\infty} p_r^{(1)} p_s^{(2)} B\left[2i(r-1)+(r-1)^2+2j(s-1)+(s-1)^2\right] \\
=&B\left(2i\sum_{r=0}^{\infty} p_r^{(1)}(r-1)+\sum_{r=0}^{\infty} p_r^{(1)}(r-1)^2+2j\sum_{s=0}^{\infty} p_s^{(2)}(s-1)+\sum_{s=0}^{\infty} p_s^{(2)}(s-1)^2\right) \\
=&2B\Big(i(\mathbb{E}[\mathcal{A}_1]-1)+j(\mathbb{E}[\mathcal{A}_2]-1)\Big)+B\Big(\mathbb{E}[(\mathcal{A}_1-1)^2]+\mathbb{E}[(\mathcal{A}_2-1)^2]\Big).
\end{aligned}
$$

Note that $\mathbb{E}[\mathcal{A}_i] < 1$, hence $2B(\mathbb{E}[\mathcal{A}_i]-1) < -C_H$ for sufficiently large B. Moreover, since $\mathbb{E}[\mathcal{A}_i^2] < \infty$, it holds that

$$
\sum_{k=0}^{\infty}\sum_{l=0}^{\infty} P_{a(\tilde{\theta})}\left((i,j),(k,l)\right)\left[F(k,l)-F(i,j)\right] \le -C((i,j),a(\tilde{\theta})),
$$

when i,j are large enough, where

$$
C((i,j),a(\tilde{\theta})) = C_H([i-1]^+ + [j-1]^+) + C_T.
$$

We observe that there exists a finite set G that contains states $\{(0,0),(0,1),(1,0)\}$ such that Eq. (2.7) is satisfied for $(i,j) \in (\mathbb{N}\cup\{0\})^2 - G$. Then, for $(i,j) \in G$, it

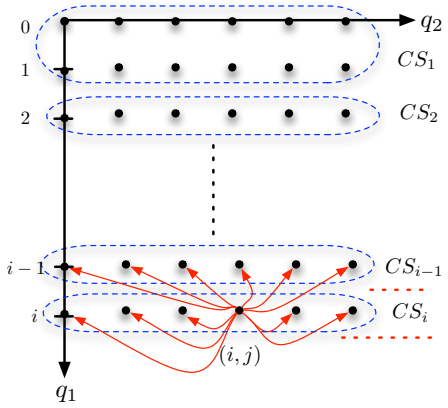

Figure 2.3: Case (ii) in the proof of Theorem 8: state (i,j) can only transit to the states in the CS_i and CS_{i-1}.

holds that

$$\begin{aligned}
&\sum_{k=0}^{\infty}\sum_{l=0}^{\infty} P_{a(\tilde{\theta})}\left((i,j),(k,l)\right) F(k,l) \\
=&B\sum_{r=0}^{\infty}\sum_{s=0}^{\infty} p_r^{(1)} p_s^{(2)} \left[([i-1]^+ + r)^2 + ([j-1]^+ + s)^2\right] \\
=&B\Big\{(i-1)^2 + 2[i-1]^+\mathbb{E}[\mathcal{A}_1] + \mathbb{E}[\mathcal{A}_1^2] + \\
&(j-1)^2 + 2[j-1]^+\mathbb{E}[\mathcal{A}_2] + \mathbb{E}[\mathcal{A}_2^2]\Big\} < \infty.
\end{aligned}$$

Therefore, the condition of Lemma 7 is satisfied, which implies, in turn, that condition (iii) in Lemma 6 is satisfied as well.

Case (ii): $p_0^{(1)} + p_1^{(1)} = 1$ and $p_0^{(2)} + p_1^{(2)} < 1$. Note that $\tilde{\theta}$ results in a reducible Markov chain. That is, there are several communicating classes [28]. We define the classes $\mathrm{CS}_1 = \{(a,b) : a = 0,1 \text{ and } b \in \mathbb{N}\cup\{0\}\}$ and $\mathrm{CS}_i = \{(a,b) : a = i, b \in \mathbb{N}\cup\{0\}\}$ for $i \geq 2$, as shown in Fig. 2.3. Then each CS_i is a communicating class under the policy $\tilde{\theta}$. The states in CS_1 are positive-recurrent, and each CS_i for $i \geq 2$

is a transient class (see [28]).

For $i \geq 2$, let $\overline{C}_{i,j}$ be the expected cost of the first passage from state (i,j) (in class CS_i) to a state in class CS_{i-1}. Moreover, we denote the expected cost of a first passage from state (i,j) to (k,l) by $\overline{C}_{(i,j),(k,l)}$. Let $T_0 = \min\{t \geq 1 : (Q_t^{(1)}, Q_t^{(2)}) = (0,0)\}$ and for $i \geq 1$, $T_i = \min\{t \geq 1 : Q_t^{(1)} = i\}$. Then we can express the expected cost of the first passage from state (i,j) to $(0,0)$ as follows.

$$\overline{C}_{(i,j),(0,0)} = \overline{C}_{i,j} + \sum_{k=1}^{i-2} \overline{C}_{i-k,Q^{(2)}_{T_{i-k}}} + \overline{C}_{(1,Q^{(2)}_{T_1}),(0,0)}.$$

Note that state (i,j) has the probability of $p_0^{(1)}$ to escape to class CS_{i-1} and $p_1^{(1)}$ to remain in class CS_i. By considering all the possible paths, we compute $\overline{C}_{i,j}$ as follows.

$$\begin{aligned}
\overline{C}_{i,j} =& \mathbb{E}\left[\sum_{k=0}^{\infty}(p_1^{(1)})^k p_0^{(1)} \sum_{t=0}^{k} C((i,Q_t^{(2)}),1) | (Q_0^{(1)}, Q_0^{(2)}) = (i,j)\right] \\
=& p_0^{(1)} \mathbb{E}\left[\sum_{t=0}^{\infty} C((i,Q_t^{(2)}),1) \sum_{k=t}^{\infty}(p_1^{(1)})^k | (Q_0^{(1)}, Q_0^{(2)}) = (i,j)\right] \\
=& \mathbb{E}\left[\sum_{t=0}^{\infty}(p_1^{(1)})^t C((i,Q_t^{(2)}),1) | (Q_0^{(1)}, Q_0^{(2)}) = (i,j)\right],
\end{aligned}$$

where $C((i,Q_t^{(2)}),1) = C_T + C_H([i-1]^+ + [Q_t^{(2)} - 1]^+)$. Following the similar argument to the proof of Proposition 2, we conclude that $\overline{C}_{i,j} < \infty$. Moreover, Proposition 4 in [26] implies that $\overline{C}_{(1,j),(0,0)} < \infty$ for any j, where the intuition is that the expected traveling time from state $(1,j)$ to $(0,0)$ is finite due to the positive recurrence of CS_1. Therefore, we conclude that $\overline{C}_{(i,j),(0,0)} < \infty$.

Let $\hat{\theta}$, be a policy that always transmits until time slot T_0 after which the α-

optimal policy is employed. Then, $V_\alpha(i,j)$ can be bounded by

$$\begin{aligned} V_\alpha(i,j) \leq \quad & \mathbb{E}_{\hat{\theta}}\left[\sum_{t=0}^{T_0-1} \alpha^t C(Q_t,a_t)|Q_0=(i,j)\right] + \mathbb{E}_{\hat{\theta}}\left[\sum_{t=T_0}^{\infty} \alpha^t C(Q_t,a_t)|Q_0=(i,j)\right] \\ \leq \quad & \overline{C}_{(i,j),(0,0)} + V_\alpha(0,0). \end{aligned}$$

Then the condition (iii) of Lemma 6 is satisfied by choosing $M_{i,j} = \overline{C}_{(i,j),(0,0)}$. In particular, it holds that $v_\alpha(i,j) = V_\alpha(i,j) - V_\alpha(0,0) \leq M_{i,j}$ and $M_{i,j} < \infty$. Moreover, $\sum_{k=0}^{\infty}\sum_{l=0}^{\infty} P_1\big((i,j),(k,l)\big)M_{k,l} = \sum_{k=0}^{\infty}\sum_{l=0}^{\infty} P_1\big((i,j),(k,l)\big)\overline{C}_{(k,l),(0,0)} \leq \overline{C}_{(i,j),(0,0)} < \infty$.

Case (iii): $p_0^{(i)} + p_1^{(i)} = 1$ for $i = 1, 2$, i.e., Bernoulli arrivals to both queues. Note that in this case $\tilde{\theta}$ also results in a reducible Markov chain. The proof is similar to case (ii); we can define $M_{i,j} = \overline{C}_{(i,j),(0,0)}$, and show that $\overline{C}_{(i,j),(0,0)}$ is finite for this case. □

According to Borkar [29], it is possible to find the randomized policy that is closed to the average-optimal by applying linear programming methods for an MDP of a very generic setting, where randomized stationary policies are average-optimal. However, since the average-optimal policy has further been shown in Theorem 8 to be deterministic, in the next subsection we investigate the structural properties of the average-optimal policy and using a Markov-chain based enumeration to find the average-optimal polity that would be deterministic stationary.

2.4 Structure of the average-optimal policy: threshold type

Now that we know the average-optimal policy is stationary and deterministic, the question is how do we find it? If we know that the average-optimal policy satisfies the structural properties, then it is possible to search through the space of stationary deterministic policies and obtain the optimal one. We will study the

α-optimal policy first and then discuss how to correlate it with the average-optimal policy. Before investigating the general i.i.d. arrival model, we study a special case, namely Bernoulli process. Our objective is to determine the α-optimal policy for the Bernoulli arrival process.

Lemma 9. *For the i.i.d. Bernoulli arrival process and the system starting from the empty queues, the α-optimal policy is of threshold type. In particular, there exist optimal thresholds $L^*_{\alpha,1}$ and $L^*_{\alpha,2}$ so that the optimal deterministic action in state $(i,0)$ is to wait if $i \leq L^*_{\alpha,1}$, and to transmit without coding if $i > L^*_{\alpha,1}$; while in state $(0,j)$ is to wait if $j \leq L^*_{\alpha,2}$, and to transmit without coding if $j > L^*_{\alpha,2}$.*

Proof. We define

$$\mathcal{V}_\alpha(i,0,a) = C_H([i-a]^+) + C_T a + \alpha \sum_{k,l} P_a\big((i,0),(k,l)\big) V_\alpha(k,l).$$

According to Eq. (2.5),

$$V_\alpha(i,0) = \min_{a\in\{0,1\}} \mathcal{V}_\alpha(i,0,a).$$

Let $L^*_{\alpha,1} = \min\{i \in \mathbb{N}\cup\{0\} : \mathcal{V}_\alpha(i,0,1) > \mathcal{V}_\alpha(i,0,0)\} - 1$. Then the α-optimal policy is $a_t = 0$ for the states $(i,0)$ with $i \leq L^*_{\alpha,1}$, and $a_t = 1$ for the state $(L^*_{\alpha,1}+1, 0)$. However, the system starts with empty queues; as such, the states $(i,0)$ for $i > L^*_{\alpha,1} + 1$ are not accessible as $(L^*_{\alpha,1}+1, 0)$ only transits to $(L^*_{\alpha,1}, 0)$, $(L^*_{\alpha,1}+1, 0)$, $(L^*_{\alpha,1}, 1)$, and $(L^*_{\alpha,1}+1, 1)$. Hence, we do not need to define the policy of the states $(i,0)$ for $i > L^*_{\alpha,1} + 1$. The similar argument is applicable for the states $(0,j)$. Consequently, there exists a policy of threshold type that is α-optimal. □

Here we are providing an intuition of the threshold policy. If a packet is transmitted immediately without coding, the system cost increases significantly due to a

large transmission cost. To wait at present for a coding opportunity in the future incurs a smaller waiting cost. Therefore, the packet might be delayed until the delay cost cannot be compensated by the saving from coding. An optimal policy might be as follows: to increase as time goes the probability to transmit the packet. Moreover, we have shown in Subsection 2.3 that there is an optimal policy that is stationary and deterministic; as such the optimal policy could be threshold type.

2.4.1 General i.i.d. arrival process

For the i.i.d. Bernoulli arrival process, we have just shown that the α-optimal policy is threshold based. Our next objective is to extend this result to any i.i.d. arrival process. We define that

$$\mathcal{V}_\alpha(i,j,a) = C_H\left([i-a]^+ + [j-a]^+\right) + C_T \cdot a + \alpha\mathbb{E}[V_\alpha\left([i-a]^+ + \mathcal{A}_1, [j-a]^+ + \mathcal{A}_2\right)];$$

$$\mathcal{V}_{\alpha,n}(i,j,a) = C_H\left([i-a]^+ + [j-a]^+\right) + C_T a + \alpha\mathbb{E}[V_{\alpha,n}\left([i-a]^+ + \mathcal{A}_1, [j-a]^+ + \mathcal{A}_2\right)].$$

Then Eq. (2.5) can be written as $V_\alpha(i,j) = \min_{a\in\{0,1\}} \mathcal{V}_\alpha(i,j,a)$, while Eq. (2.6) can be written as $V_{\alpha,n+1}(i,j) = \min_{a\in\{0,1\}} \mathcal{V}_{\alpha,n}(i,j,a)$. For every discount factor α, we want to show that there exists an α-optimal policy that is of threshold type. To be precise, let the α-optimal policy for the first dimension be $a^*_{\alpha,i} = \min\{a' \in \arg\min_{a\in\{0,1\}} \mathcal{V}_\alpha(i,0,a)\}$. This notation also used in [19] combines two operations: First we let $\Lambda = \{a \in \{0,1\} : \min \mathcal{V}_{\alpha,n}(i,0,a)\}$, and then do $\min\Lambda$. In other words, we choose $a = 0$ when both $a = 0$ and $a = 1$ result in the same $\mathcal{V}_{\alpha,n}(i,j,a)$. We will show that $a^*_{\alpha,i}$ is non-decreasing as i increases, and so is the second dimension. We start with a number of definitions that describe the properties of $V_\alpha(i,j)$.

Definition 10 ([27], Submodularity)**.** A function $f : (\mathbb{N}\cup\{0\})^2 \to \mathbb{R}$ is submodular

if for all $i, j \in \mathbb{N} \cup \{0\}$

$$f(i,j) + f(i+1,j+1) \leq f(i+1,j) + f(i,j+1).$$

Definition 11 ($\mathcal{K}$-Convexity)**.** A function $f : (\mathbb{N} \cup \{0\})^2 \to \mathbb{R}$ is $\mathcal{K}$-convex (where $\mathcal{K} \in \mathbb{N}$) if for every $i, j \in \mathbb{N} \cup \{0\}$

$$f(i+\mathcal{K},j) - f(i,j) \leq f(i+\mathcal{K}+1,j) - f(i+1,j);$$
$$f(i,j+\mathcal{K}) - f(i,j) \leq f(i,j+\mathcal{K}+1) - f(i,j+1).$$

Definition 12 ($\mathcal{K}$-Subconvexity)**.** A function $f : (\mathbb{N} \cup \{0\})^2 \to \mathbb{R}$ is $\mathcal{K}$-subconvex (where $\mathcal{K} \in \mathbb{N}$) if for all $i, j \in \mathbb{N} \cup \{0\}$

$$f(i+\mathcal{K},j+\mathcal{K}) - f(i,j) \leq f(i+\mathcal{K}+1,j+\mathcal{K}) - f(i+1,j);$$
$$f(i+\mathcal{K},j+\mathcal{K}) - f(i,j) \leq f(i+\mathcal{K},j+\mathcal{K}+1) - f(i,j+1).$$

Remark 13. If a function $f : (\mathbb{N} \cup \{0\})^2 \to \mathbb{R}$ is submodular and $\mathcal{K}$-subconvex, then it is $\mathcal{K}$-convex, and for every $r \in \mathbb{N}$ with $1 \leq r < \mathcal{K}$,

$$f(i+\mathcal{K},j+r) - f(i,j) \leq f(i+\mathcal{K}+1,j+r) - f(i+1,j);$$
$$f(i+r,j+\mathcal{K}) - f(i,j) \leq f(i+r,j+\mathcal{K}+1) - f(i,j+1).$$

For simplicity, we will ignore $\mathcal{K}$ in definitions 11 and 12 when $\mathcal{K} = 1$. We will show in Subsection 2.4.3 that $V_\alpha(i,j)$ is non-decreasing, submodular, and subconvex, that result in the threshold base of α-optimal policy. Note that the definition of $\mathcal{K}$-Convexity (Definition 11) is dimension-wise, which is different from the definition of convexity for the continuous function in two dimensions.

2.4.2 Proof overview

Before the technical proofs in Subsection 2.4.3, in this subsection, we overview why submodularity and subconvexity of $V_\alpha(i,j)$ lead to the α-optimality of the threshold based policy.

- *We claim that to show that α-optimal policy is monotonic w.r.t. state $(i,0)$, it suffices to show that $\mathcal{V}_\alpha(i,0,1)-\mathcal{V}_\alpha(i,0,0)$ is a non-increasing function w.r.t. i:* Suppose that $\mathcal{V}_\alpha(i,0,1)-\mathcal{V}_\alpha(i,0,0)$ is non-increasing, i.e., $\mathcal{V}_\alpha(i+1,0,1)-\mathcal{V}_\alpha(i+1,0,0)\leq\mathcal{V}_\alpha(i,0,1)-\mathcal{V}_\alpha(i,0,0)$. If the α-optimal policy for state $(i,0)$ is $a^*_{\alpha,i}=1$, i.e., $\mathcal{V}_\alpha(i,0,1)-\mathcal{V}_\alpha(i,0,0)\leq 0$, then the α-optimal policy for state $(i+1,0)$ is also $a^*_{\alpha,i+1}=1$ according to $\mathcal{V}_\alpha(i+1,0,1)-\mathcal{V}_\alpha(i+1,0,0)\leq\mathcal{V}_\alpha(i,0,1)-\mathcal{V}_\alpha(i,0,0)\leq 0$. Similarly, if the α-optimal policy for state $(i+1,0)$ is $a^*_{\alpha,i+1}=0$ then the α-optimal policy for state $(i,0)$ is $a^*_{\alpha,i}=0$. Hence, the α-optimal policy is monotonic in i.

- *We claim that to prove that $\mathcal{V}_\alpha(i,0,1)-\mathcal{V}_\alpha(i,0,0)$ is non-increasing, it is sufficient to show that $V_\alpha(i,j)$ is convex:* When $i\geq 1$, the claim is true since

$$\begin{aligned}&\mathcal{V}_\alpha(i,0,1)-\mathcal{V}_\alpha(i,0,0)\\ =&C_T-C_H+\alpha\mathbb{E}[V_\alpha(i-1+\mathcal{A}_1,\mathcal{A}_2)-V_\alpha(i+\mathcal{A}_1,\mathcal{A}_2)].\end{aligned}$$

- *Similarly, to show that α-optimal policy of state (i,j) is monotonic w.r.t. i for fixed j and vice versa, it suffices to show that $V_\alpha(i,j)$ is subconvex:* When

$i, j \geq 1$, we observe that

$$\begin{aligned}
&\mathcal{V}_\alpha(i,j,1) - \mathcal{V}_\alpha(i,j,0) \\
=&C_t - 2C_h + \alpha\mathbb{E}[V_\alpha(i-1+\mathcal{A}_1, j-1+\mathcal{A}_2) - V_\alpha(i+\mathcal{A}_1, j+\mathcal{A}_2)].
\end{aligned}$$

- *We claim that $V_\alpha(i,j)$ is submodular:* We intend to prove the convexity and subconvexity of $V_\alpha(i,j)$ by induction, which will require the relation between $V_\alpha(i,j) + V_\alpha(i+1,j+1)$ and $V_\alpha(i+1,j) + V_\alpha(i,j+1)$. There will be two choices: (i) $V_\alpha(i,j) + V_\alpha(i+1,j+1) \leq V_\alpha(i+1,j) + V_\alpha(i,j+1)$, or (ii) $V_\alpha(i,j) + V_\alpha(i+1,j+1) \geq V_\alpha(i+1,j) + V_\alpha(i,j+1)$. First, We might assume that $V_\alpha(i,j)$ satisfies (i). Then (i) and the subconvexity of $V_\alpha(i,j)$ implies the convexity of $V_\alpha(i,j)$. In the contrary, the convexity of $V_\alpha(i,j)$ and (ii) lead to the subconvexity of $V_\alpha(i,j)$. In other words, both choices are possible since they do not violate the convexity and subconvexity of $V_\alpha(i,j)$. However, we are going to argue that the choice (ii) is wrong as follows. Suppose that the actions of α-optimal policy for the states (i,j), $(i+1,j)$, $(i,j+1)$, $(i+1,j+1)$ are $0,0,1,1$ respectively. If the choice (ii) is true, then when $i \geq 1$, we have

$$\begin{aligned}
&C_H(i+j) + \mathbb{E}[V_{\alpha,n}(i+\mathcal{A}_1, j+\mathcal{A}_2)]+ \\
&C_T + C_H(i+j) + \mathbb{E}[V_{\alpha,n}(i+\mathcal{A}_1, j+\mathcal{A}_2)] \\
\geq&C_H(i+1+j) + \mathbb{E}[V_{\alpha,n}(i+1+\mathcal{A}_1, j+\mathcal{A}_2)]+ \\
&C_T + C_H(i-1+j) + \mathbb{E}[V_{\alpha,n}(i-1+\mathcal{A}_1, j+\mathcal{A}_2)].
\end{aligned}$$

By simplifying the above inequality, we can observe the contradiction to the fact that $V_{\alpha,n}(i,j)$ is convex. Therefore, $V_\alpha(i,j)$ is submodular.

Based on the above discussion, we understand that if we show $V_\alpha(i,j)$ is submodular and subconvex, then the α-optimal policy of state (i,j) is non-decreasing separately in the direction of i and j (i.e., threshold type). Next, we briefly discuss how Lemmas 14-17 and Theorem 18 in the next subsection work together. Theorem 18 states that the α-optimal policy is of threshold type, while the proof is based on an induction on n in Eq. (2.6). First, when $n = 0$ we observe that $V_{\alpha,0}(i,j)$ is non-decreasing, submodular, and subconvex. Second, based on Lemma 14 and Corollary 15, $\min\{a' \in \arg\min_{a\in\{0,1\}} \mathcal{V}_{\alpha,0}(i,j,a)\}$ is non-decreasing w.r.t. i for fixed j, and vice versa. Third, according to Lemmas 5, 16, and 17, we know that $V_{\alpha,1}(i,j)$ is non-decreasing, submodular, and subconvex. Therefore, as n goes to infinity, we conclude that $V_\alpha(i,j)$ is non-decreasing, submodular, and subconvex, as well as $\min\{a' \in \arg\min_{a\in\{0,1\}} \mathcal{V}_\alpha(i,j,a)\}$ is non-decreasing w.r.t. i for fixed j, and vice versa.

2.4.3 Main results

Lemma 14. *Given $0 < \alpha < 1$ and $n \in \mathbb{N} \cup \{0\}$. If $V_{\alpha,n}(i,j)$ is non-decreasing, submodular, and subconvex, then $\mathcal{V}_{\alpha,n}(i,j,a)$ is submodular for i and a when j is fixed, and so is for j and a when i is fixed.*

Proof. We define $\Delta\mathcal{V}_{\alpha,n}(i,j) = \mathcal{V}_{\alpha,n}(i,j,1) - \mathcal{V}_{\alpha,n}(i,j,0)$. We claim that $\Delta\mathcal{V}_{\alpha,n}(i,j)$ is non-increasing, i.e., $\Delta\mathcal{V}_{\alpha,n}(i,j)$ is a non-increasing function w.r.t. i while j is fixed, and vice versa (we will focus on the former part). Notice that

$$\begin{aligned}\Delta\mathcal{V}_{\alpha,n}(i,j) &= C_H([i-1]^+ + (j-1)^+) + C_T + \\ &\quad \alpha\mathbb{E}[V_{\alpha,n}([i-1]^+ + \mathcal{A}_1, [j-1]^+ + \mathcal{A}_2)] - \\ &\quad C_H(i+j) - \alpha\mathbb{E}[V_{\alpha,n}(i+\mathcal{A}_1, j+\mathcal{A}_2)].\end{aligned}$$

To be precise, when $i \geq 1$,

$$\begin{aligned}\Delta\mathcal{V}_{\alpha,n}(i,j) =& C_T - 2C_H + \alpha\mathbb{E}[V_{\alpha,n}(i-1+\mathcal{A}_1, j-1+\mathcal{A}_2) - \\ & V_{\alpha,n}(i+\mathcal{A}_1, j+\mathcal{A}_2)] \text{ for } j \geq 1; \qquad (2.8)\\ \Delta\mathcal{V}_{\alpha,n}(i,j) =& C_T - C_H + \alpha\mathbb{E}[V_{\alpha,n}(i-1+\mathcal{A}_1, \mathcal{A}_2) - \\ & V_{\alpha,n}(i+\mathcal{A}_1, \mathcal{A}_2)] \text{ for } j = 0. \qquad (2.9)\end{aligned}$$

Because of the subconvexity of $V_{\alpha,n}(i,j)$ in Eq. (2.8), when $i \geq 1$ and $j \geq 1$, $\Delta\mathcal{V}_{\alpha,n}(i,j)$ does not increase as i increases. The same is for $i \geq 1$ and $j = 0$ in Eq. (2.9) due to the convexity of $V_{\alpha,n}(i,j)$.

We proceed to establish the boundary conditions. When $j \geq 1$,

$$\begin{aligned}\Delta\mathcal{V}_{\alpha,n}(1,j) &= C_T - 2C_H + \alpha\mathbb{E}[V_{\alpha,n}(\mathcal{A}_1, j-1+\mathcal{A}_2) - V_{\alpha,n}(1+\mathcal{A}_1, j+\mathcal{A}_2)];\\ \Delta\mathcal{V}_{\alpha,n}(0,j) &= C_T - C_H + \alpha\mathbb{E}[V_{\alpha,n}(\mathcal{A}_1, j-1+\mathcal{A}_2) - V_{\alpha,n}(\mathcal{A}_1, j+\mathcal{A}_2)].\end{aligned}$$

Note that $\mathbb{E}[V_{\alpha,n}(1+\mathcal{A}_1, j+\mathcal{A}_2)] \geq \mathbb{E}[V_{\alpha,n}(\mathcal{A}_1, j+\mathcal{A}_2)]$ according to non-decreasing $V_{\alpha,n}(i,j)$ and then $\Delta\mathcal{V}_{\alpha,n}(1,j) \leq \Delta\mathcal{V}_{\alpha,n}(0,j)$ when $j \geq 1$. Finally, when $j = 0$ we have

$$\begin{aligned}\Delta\mathcal{V}_{\alpha,n}(1,0) &= C_T - C_H + \alpha\mathbb{E}[V_{\alpha,n}(\mathcal{A}_1, \mathcal{A}_2) - V_{\alpha,n}(1+\mathcal{A}_1, \mathcal{A}_2)];\\ \Delta\mathcal{V}_{\alpha,n}(0,0) &= C_T.\end{aligned}$$

Here, $\Delta\mathcal{V}_{\alpha,n}(1,0) \leq \Delta\mathcal{V}_{\alpha,n}(0,0)$ since $\mathbb{E}[V_{\alpha,n}(\mathcal{A}_1, \mathcal{A}_2) - V_{\alpha,n}(1+\mathcal{A}_1, \mathcal{A}_2)] \leq 0$ as $V_{\alpha,n}(i,j)$ is non-decreasing. Consequently, $\Delta\mathcal{V}_{\alpha,n}(i,j)$ is a non-increasing function w.r.t. i while j is fixed. □

Submodularity of $\mathcal{V}_{\alpha,n}(i,j,a)$ implies the monotonicity of the optimal minimizing

policy [19, Lemma 4.7.1] as described in the following Corollary. This property will simplify the proofs of Lemmas 16 and 17.

Corollary 15. *Given* $0 < \alpha < 1$ *and* $n \in \mathbb{N} \cup \{0\}$. *If* $V_{\alpha,n}(i,j)$ *is non-decreasing, submodular, and subconvex, then* $\min\{a' \in \arg\min_{a\in\{0,1\}} \mathcal{V}_{\alpha,n}(i,j,a)\}$ *is non-decreasing w.r.t.* i *for fixed* j*, and vice versa.*

Lemma 16. *Given* $0 < \alpha < 1$ *and* $n \in \mathbb{N} \cup \{0\}$. *If* $V_{\alpha,n}(i,j)$ *is non-decreasing, submodular, and subconvex, then* $V_{\alpha,n+1}(i,j)$ *is submodular.*

Proof. We intend to show that $V_{\alpha,n+1}(i+1,j+1) - V_{\alpha,n+1}(i+1,j) \leq V_{\alpha,n+1}(i,j+1) - V_{\alpha,n+1}(i,j)$ for all $i,j \in \mathbb{N} \cup \{0\}$. According to Corollary 15, only 6 cases of $(a^*_{i,j}, a^*_{i+1,j}, a^*_{i,j+1}, a^*_{i+1,j+1})$ are considered, where $a^*_{i,j} = \min\{a' \in \arg\min_{a\in\{0,1\}} \mathcal{V}_{\alpha,n}(i,j,a)\}$.

Case (i): if $(a^*_{i,j}, a^*_{i+1,j}, a^*_{i,j+1}, a^*_{i+1,j+1}) = (1,1,1,1)$, we claim that

$$\begin{aligned} & \mathbb{E}[V_{\alpha,n}(i+\mathcal{A}_1, j+\mathcal{A}_2) - V_{\alpha,n}(i+\mathcal{A}_1, [j-1]^+ + \mathcal{A}_2)] \\ \leq\ & \mathbb{E}[V_{\alpha,n}([i-1]^+ + \mathcal{A}_1, j+\mathcal{A}_2) - V_{\alpha,n}([i-1]^+ + \mathcal{A}_1, [j-1]^+ + \mathcal{A}_2)]. \end{aligned}$$

When $i,j \neq 0$, it is true according to submodularity of $V_{\alpha,n}(i,j)$. Otherwise, both sides of the inequality are 0.

Case (ii): if $(a^*_{i,j}, a^*_{i+1,j}, a^*_{i,j+1}, a^*_{i+1,j+1}) = (0,0,0,0)$, we claim that

$$\begin{aligned} & \mathbb{E}[V_{\alpha,n}(i+1+\mathcal{A}_1, j+1+\mathcal{A}_2) - V_{\alpha,n}(i+1+\mathcal{A}_1, j+\mathcal{A}_2)] \\ \leq & \mathbb{E}[V_{\alpha,n}(i+\mathcal{A}_1, j+1+\mathcal{A}_2) - V_{\alpha,n}(i+\mathcal{A}_1, j+\mathcal{A}_2)]. \end{aligned}$$

This is obvious from the submodularity of $V_{\alpha,n}(i,j)$.

Case (iii): if $(a^*_{i,j}, a^*_{i+1,j}, a^*_{i,j+1}, a^*_{i+1,j+1}) = (0,0,0,1)$, we claim that

$$\begin{aligned} & C_T - C_H + \alpha\mathbb{E}[V_{\alpha,n}(i+\mathcal{A}_1, j+\mathcal{A}_2) - V_{\alpha,n}(i+1+\mathcal{A}_1, j+\mathcal{A}_2)] \\ \leq & C_H + \alpha\mathbb{E}[V_{\alpha,n}(i+\mathcal{A}_1, j+1+\mathcal{A}_2) - V_{\alpha,n}(i+\mathcal{A}_1, j+\mathcal{A}_2)]. \end{aligned}$$

From the submodularity of $V_{\alpha,n}(i,j)$, it is obtained that

$$\begin{aligned} & V_{\alpha,n}(i,j) - V_{\alpha,n}(i+1,j) + V_{\alpha,n}(i,j) - V_{\alpha,n}(i,j+1) \\ \leq & V_{\alpha,n}(i,j) - V_{\alpha,n}(i+1,j) + V_{\alpha,n}(i+1,j) - V_{\alpha,n}(i+1,j+1) \\ = & V_{\alpha,n}(i,j) - V_{\alpha,n}(i+1,j+1). \end{aligned}$$

Since $a^*_{i+1,j+1} = 1$, we have $\Delta\mathcal{V}_{\alpha,n}(i+1,j+1) \leq 0$, i.e.,

$$C_T - 2C_H + \alpha\mathbb{E}[V_{\alpha,n}(i+\mathcal{A}_1, j+\mathcal{A}_2) - V_{\alpha,n}(i+1+\mathcal{A}_1, j+1+\mathcal{A}_2)] \leq 0.$$

The claim follows from the following equation:

$$\begin{aligned} & C_T - 2C_H + \alpha\mathbb{E}[V_{\alpha,n}(i+\mathcal{A}_1, j+\mathcal{A}_2) - V_{\alpha,n}(i+1+\mathcal{A}_1, j+\mathcal{A}_2) + \\ & \qquad V_n(i+\mathcal{A}_1, j+\mathcal{A}_2) - V_{\alpha,n}(i+\mathcal{A}_1, j+1+\mathcal{A}_2)] \\ \leq & \Delta\mathcal{V}_{\alpha,n}(i+1,j+1) \leq 0. \end{aligned}$$

Case (iv): if $(a^*_{i,j}, a^*_{i+1,j}, a^*_{i,j+1}, a^*_{i+1,j+1}) = (0,0,1,1)$, we claim that

$$\begin{aligned} & -C_H + \alpha\mathbb{E}[V_{\alpha,n}(i+\mathcal{A}_1, j+\mathcal{A}_2) - V_{\alpha,n}(i+1+\mathcal{A}_1, j+\mathcal{A}_2)] \\ \leq & C_H([i-1]^+ - i) + \alpha\mathbb{E}[V_{\alpha,n}([i-1]^+ + \mathcal{A}_1, j+\mathcal{A}_2) - V_{\alpha,n}(i+\mathcal{A}_1, j+\mathcal{A}_2)] \end{aligned}$$

When $i \neq 0$, it is satisfied because $V_{\alpha,n}(i,j)$ is convex. Otherwise, it is true since

$V_{\alpha,n}(i,j)$ is non-decreasing.

Case (v): if $(a^*_{i,j}, a^*_{i+1,j}, a^*_{i,j+1}, a^*_{i+1,j+1}) = (0,1,0,1)$, we claim that

$$\begin{aligned}
&C_H(j-[j-1]^+) + \alpha\mathbb{E}[V_{\alpha,n}(i+\mathcal{A}_1, j+\mathcal{A}_2) - V_{\alpha,n}(i+\mathcal{A}_1, [j-1]^+ + \mathcal{A}_2)] \\
\leq & C_H + \alpha\mathbb{E}[V_{\alpha,n}(i+\mathcal{A}_1, j+1+\mathcal{A}_2) - V_{\alpha,n}(i+\mathcal{A}_1, j+\mathcal{A}_2)].
\end{aligned}$$

When $j \neq 0$, it holds since $V_{\alpha,n}(i,j)$ is convex. It is true for other cases because of the non-decreasing $V_{\alpha,n}(i,j)$.

Case (vi): if $(a^*_{i,j}, a^*_{i+1,j}, a^*_{i,j+1}, a^*_{i+1,j+1}) = (0,1,1,1)$, we claim that

$$\begin{aligned}
&C_H(j-[j-1]^+) + \alpha\mathbb{E}[V_{\alpha,n}(i+\mathcal{A}_1, j+\mathcal{A}_2) - V_{\alpha,n}(i+\mathcal{A}_1, [j-1]^+ + \mathcal{A}_2)] \\
\leq & C_T + C_H([i-1]^+ - i) + \alpha\mathbb{E}[V_{\alpha,n}([i-1]^+ + \mathcal{A}_1, j+\mathcal{A}_2) - V_{\alpha,n}(i+\mathcal{A}_1, j+\mathcal{A}_2)].
\end{aligned}$$

Based on the submodularity of $V_{\alpha,n}(i,j)$, we have

$$\begin{aligned}
&V_{\alpha,n}([i-1]^+, j) - V_{\alpha,n}(i,j) + V_{\alpha,n}(i, [j-1]^+) - V_{\alpha,n}(i,j) \\
\geq & V_{\alpha,n}([i-1]^+, [j-1]^+) - V_{\alpha,n}(i, [j-1]^+) + V_{\alpha,n}(i, [j-1]^+) - V_{\alpha,n}(i,j) \\
= & V_{\alpha,n}([i-1]^+, [j-1]^+) - V_{\alpha,n}(i,j).
\end{aligned}$$

It is noted that $a^*_{i,j} = 0$ and hence $\Delta\mathcal{V}_{\alpha,n}(i,j) \geq 0$, i.e.,

$$\begin{aligned}
&C_T + C_H([i-1]^+ + [j-1]^+ - i - j) + \\
&\alpha\mathbb{E}[V_{\alpha,n}([i-1]^+ + \mathcal{A}_1, [j-1]^+ + \mathcal{A}_1) - V_{\alpha,n}(i+\mathcal{A}_1, j+\mathcal{A}_1)] \geq 0.
\end{aligned}$$

Therefore, it can be concluded that

$$\begin{aligned}
&C_T + C_H([i-1]^+ + [j-1]^+ - i - j) + \\
&\alpha\mathbb{E}[V_{\alpha,n}([i-1]^+ + \mathcal{A}_1, j + \mathcal{A}_2) - V_{\alpha,n}(i + \mathcal{A}_1, j + \mathcal{A}_2) + \\
&\quad V_{\alpha,n}(i + \mathcal{A}_1, [j-1]^+ + \mathcal{A}_2) - V_{\alpha,n}(i + \mathcal{A}_1, j + \mathcal{A}_2)] \\
\geq &\Delta\mathcal{V}_{\alpha,n}(i,j) \geq 0.
\end{aligned}$$

□

Lemma 17. *Given $0 < \alpha < 1$ and $n \in \mathbb{N} \cup \{0\}$. If $V_{\alpha,n}(i,j)$ is non-decreasing, submodular, and subconvex, then $V_{\alpha,n+1}(i,j)$ is subconvex.*

Proof. We want to show that $V_{\alpha,n+1}(i+1,j+1) - V_{\alpha,n+1}(i,j) \leq V_{\alpha,n+1}(i+2,j+1) - V_{\alpha,n+1}(i+1,j)$ for all i and j. There will be 5 cases of $(a^*_{i,j}, a^*_{i+1,j}, a^*_{i+1,j+1}, a^*_{i+2,j+1})$ that need to be considered.

Case (i): if $(a^*_{i,j}, a^*_{i+1,j}, a^*_{i+1,j+1}, a^*_{i+2,j+1}) = (1,1,1,1)$, we claim that

$$\begin{aligned}
&C_H(i - [i-1]^+) + \alpha\mathbb{E}[V_{\alpha,n}(i + \mathcal{A}_1, j + \mathcal{A}_2) - V_{\alpha,n}([i-1]^+ + \mathcal{A}_1, [j-1]^+ + \mathcal{A}_2)] \\
\leq &C_H + \alpha\mathbb{E}[V_{\alpha,n}(i + 1 + \mathcal{A}_1, j + \mathcal{A}_2) - V_{\alpha,n}(i + \mathcal{A}_1, [j-1]^+ + \mathcal{A}_2)].
\end{aligned}$$

When $i, j \neq 0$, it is true according to the subconvexity of $V_{\alpha,n}(i,j)$. The argument is satisfied for $i = 0, j \neq 0$ due to the the non-decreasing $V_{\alpha,n}(i,j)$, and for the case $i \neq 0, j = 0$ due to the convexity of $V_{\alpha,n}(i,j)$. Otherwise, it holds according to the non-decreasing property.

Case (ii): if $(a^*_{i,j}, a^*_{i+1,j}, a^*_{i+1,j+1}, a^*_{i+2,j+1}) = (0,0,0,0)$, we claim that

$$\begin{aligned}&\mathbb{E}[V_{\alpha,n}(i+1+\mathcal{A}_1, j+1+\mathcal{A}_2) - V_{\alpha,n}(i+\mathcal{A}_1, j+\mathcal{A}_2)]\\ \leq&\mathbb{E}[V_{\alpha,n}(i+2+\mathcal{A}_1, j+1+\mathcal{A}_2) - V_{\alpha,n}(i+1+\mathcal{A}_1, j+\mathcal{A}_2)].\end{aligned}$$

The above results from the subconvexity of $V_{\alpha,n}(i,j)$.

Case (iii): if $(a^*_{i,j}, a^*_{i+1,j}, a^*_{i+1,j+1}, a^*_{i+2,j+1}) = (0,0,0,1)$, we claim that

$$\begin{aligned}&2C_H + \alpha\mathbb{E}[V_{\alpha,n}(i+1+\mathcal{A}_1, j+1+\mathcal{A}_2) -\\ &V_{\alpha,n}(i+\mathcal{A}_1, j+\mathcal{A}_2)] \leq C_T.\end{aligned}$$

Since $a^*_{i+1,j+1} = 0$, we have $\Delta\mathcal{V}_{\alpha,n}(i+1,j+1) \geq 0$, i.e.,

$$C_T - 2C_H + \alpha\mathbb{E}[V_{\alpha,n}(i+\mathcal{A}_1, j+\mathcal{A}_2) - V_{\alpha,n}(i+1+\mathcal{A}_1, j+1+\mathcal{A}_2)] \geq 0.$$

Hence the claim is verified.

Case (iv): if $(a^*_{i,j}, a^*_{i+1,j}, a^*_{i+1,j+1}, a^*_{i+2,j+1}) = (0,0,1,1)$, it is trivial since the both $V_{\alpha,n+1}(i+1,j+1) - V_{\alpha,n+1}(i,j)$ and $V_{\alpha,n+1}(i+2,j+1) - V_{\alpha,n+1}(i+1,j)$ are zeros.

Case (v): if $(a^*_{i,j}, a^*_{i+1,j}, a^*_{i+1,j+1}, a^*_{i+2,j+1}) = (0,1,1,1)$, we claim that

$$\begin{aligned}C_T \;\leq\; & C_H(1+j-[j-1]^+) +\\ & \alpha\mathbb{E}[V_{\alpha,n}(i+1+\mathcal{A}_1, j+\mathcal{A}_2) - V_{\alpha,n}(i+\mathcal{A}_1, [j-1]^+ + \mathcal{A}_2)].\end{aligned}$$

Notice that $a^*_{i+1,j} = 1$, so $\Delta\mathcal{V}_{\alpha,n}(i+1,j) \leq 0$, i.e.,

$$\begin{aligned}&C_T - C_H(1+j-[j-1]^+) +\\ &\alpha\mathbb{E}[V_{\alpha,n}(i+\mathcal{A}_1, [j-1]^+ + \mathcal{A}_2) - V_{\alpha,n}(i+1+\mathcal{A}_1, j+\mathcal{A}_2)] \leq 0.\end{aligned}$$

□

Based on the properties of $V_\alpha(i,j)$, we are ready to state the optimality of the threshold type policy in terms of the total expected discounted cost.

Theorem 18. *For the MDP$\{(Q_t, a_t), t \geq 0\}$ with any i.i.d. arrival processes to both queues, there exists an α-optimal policy that is of threshold type. Given $Q_t^{(2)}$, the α-optimal policy is monotone w.r.t. $Q_t^{(1)}$, and vice versa.*

Proof. We prove by induction. $V_{\alpha,0}(i,j) = 0$ is non-decreasing, submodular, and sub-convex, that leads to the non-decreasing $\min\{a' \in \arg\min_{a\in\{0,1\}} \mathcal{V}_{\alpha,0}(i,j,a)\}$ based on Corollary 15. These properties propagate as n goes to infinity according to lemmas 5, 16, 17, and Corollary 15. □

Thus far, the α-optimal policy is characterized. A useful relation between the average-optimal policy and the α-optimal policy is described in the following lemma.

Lemma 19 ([26], Lemma and Theorem (i))**.** *Consider MDP$\{(Q_t, a_t), t \geq 0\}$. Let $\{\alpha_n\}$ converging to 1 be any sequence of discount factors associated with the α-optimal policy $\{\theta_{\alpha_n}(i,j)\}$. There exists a subsequence $\{\beta_n\}$ and a stationary policy $\theta^*(i,j)$ that is the limit point of $\{\theta_{\beta_n}(i,j)\}$. If the three conditions in Lemma 6 are satisfied, $\theta^*(i,j)$ is the average-optimal policy for Eq. (2.2).*

Theorem 20. *Consider an i.i.d. arrival to both queues. For the MDP$\{(Q_t, a_t), t \geq 0\}$, the average-optimal policy is of threshold type. There exist the optimal thresholds L_1^* and L_2^* so that the optimal deterministic action in states $(i,0)$ is to wait if $i \leq L_1^*$, and to transmit without coding if $i > L_1^*$; while in state $(0,j)$ is to wait if $j \leq L_2^*$, and to transmit without coding if $j > L_2^*$.*

Proof. Let $(\tilde{i}, 0)$ be any state which average-optimal policy is to transmit, i.e., $\theta^*(\tilde{i}, 0) = 1$ in Lemma 19. Since there is a sequence of discount factors $\{\beta_n\}$ such that

$\theta_{\beta_n}(i,j) \to \theta^*(i,j)$, then there exists $N > 0$ so that $\theta_{\beta_n}(\tilde{i},0) = 1$ for all $n \geq N$. Due to the monotonicity of α-optimal policy in Theorem 18, $\theta_{\beta_n}(i,0) = 1$ for all $i \geq \tilde{i}$ and $n \geq N$. Therefore, $\theta^*(i,0) = 1$ for all $i \geq \tilde{i}$. To conclude, the average-optimal policy is of threshold type. □

2.5 Obtaining the optimal deterministic stationary policy

We have shown in the previous subsections that the average-optimal policy is stationary, deterministic and of threshold type, so we only need to consider the subset of deterministic stationary policies. Given the thresholds of the both queues, the MDP is reduced to a Markov chain. The next step is to find the optimal threshold. First note that the condition $\mathbb{E}[\mathcal{A}_i] < 1$ might not be sufficient for the stability of the queues since the threshold based policy leads to an average service rate lower than 1 packet per time slot. In the following theorem, we claim that the conditions $\mathbb{E}[\mathcal{A}_i^2] < \infty$ and $\mathbb{E}[\mathcal{A}_i] < 1$ for $i = 1, 2$ are enough for the stability of the queues.

Theorem 21. *For the MDP$\{(Q_t, a_t), t \geq 0\}$ with $\mathbb{E}[\mathcal{A}_i^2] < \infty$ and $\mathbb{E}[\mathcal{A}_i] < 1$ for $i = 1, 2$. The reduced Markov chain from applying the stationary and deterministic threshold based policy to MDP is positive recurrent, i.e., the stationary distribution exists.*

Proof. The proof is based on Foster-Lyapunov theorem [30] associated with the Lyapunov function $\mathcal{L}(x, y) = x^2 + y^2$. Notice that

$$\begin{aligned} Q_{t+1}^{(i)} &= [Q_t^{(i)} - a_t]^+ + \mathcal{A}_i \\ &= Q_t^{(i)} - a_t + U_t^{(i)} + \mathcal{A}_i, \end{aligned}$$

where

$$U_t^{(i)} = \begin{cases} 0 & \text{if } Q_t^{(i)} - a_t \geq 0 \\ 1 & \text{if } Q_t^{(i)} - a_t = -1. \end{cases}$$

Then it can be observed that

$$\begin{aligned}
&\mathbb{E}\left[\mathcal{L}(Q_{t+1}^{(1)}, Q_{t+1}^{(2)}) - \mathcal{L}(Q_t^{(1)}, Q_t^{(2)})|Q_t^{(1)} = x, Q_t^{(2)} = y\right] \\
=&\mathbb{E}\left[\sum_{i=1}^{2}(Q_t^{(i)} - a_t + U_t^{(i)} + \mathcal{A}_i)^2|Q_t^{(1)} = x, Q_t^{(2)} = y\right] - (x^2 + y^2) \\
=&\sum_{i=1}^{2}\mathbb{E}\left[(Q_t^{(i)} - a_t + \mathcal{A}_i)^2|Q_t^{(1)} = x, Q_t^{(2)} = y\right] + \\
&\sum_{i=1}^{2}\mathbb{E}\left[(U_t^{(i)})^2|Q_t^{(1)} = x, Q_t^{(2)} = y\right] + \\
&\sum_{i=1}^{2}\mathbb{E}\left[2U_t^{(i)}(Q_t^{(i)} - a_t + \mathcal{A}_i)|Q_t^{(1)} = x, Q_t^{(2)} = y\right] - (x^2 + y^2) \\
\leq&\sum_{i=1}^{2}\mathbb{E}\left[(Q_t^{(i)} - a_t + \mathcal{A}_i)^2|Q_t^{(1)} = x, Q_t^{(2)} = y\right] + 2 + 2\mathbb{E}[\mathcal{A}_1] + 2\mathbb{E}[\mathcal{A}_2] - (x^2 + y^2)
\end{aligned} \tag{2.10}$$

$$\begin{aligned}
=&2x\mathbb{E}\left[\mathcal{A}_1 - a_t|Q_t^{(1)} = x, Q_t^{(2)} = y\right] + 2y\mathbb{E}\left[\mathcal{A}_2 - a_t|Q_t^{(1)} = x, Q_t^{(2)} = y\right] + \\
&\mathbb{E}\left[(\mathcal{A}_1 - a_t)^2|Q_t^{(1)} = x, Q_t^{(2)} = y\right] + \mathbb{E}\left[(\mathcal{A}_2 - a_t)^2|Q_t^{(1)} = x, Q_t^{(2)} = y\right] + \\
&2\mathbb{E}[\mathcal{A}_1] + 2\mathbb{E}[\mathcal{A}_2] + 2 \\
\leq&2x\mathbb{E}\left[\mathcal{A}_1 - a_t|Q_t^{(1)} = x, Q_t^{(2)} = y\right] + 2y\mathbb{E}\left[\mathcal{A}_2 - a_t|Q_t^{(1)} = x, Q_t^{(2)} = y\right] + \\
&\mathbb{E}[\mathcal{A}_1^2] + 1 + \mathbb{E}[\mathcal{A}_2^2] + 1 + 2\mathbb{E}[\mathcal{A}_1] + 2\mathbb{E}[\mathcal{A}_2] + 2
\end{aligned} \tag{2.11}$$

$$
\begin{aligned}
=&\mathbb{E}[\mathcal{A}_1^2] + \mathbb{E}[\mathcal{A}_2^2] + 2\mathbb{E}[\mathcal{A}_1] + 2\mathbb{E}[\mathcal{A}_2] + 4 + \\
&\begin{cases} 2x(\mathbb{E}[\mathcal{A}_1] - 1) + 2y(\mathbb{E}[\mathcal{A}_2] - 1) & \text{if } (x,y) \in \mathcal{B}^c \\ 2x\mathbb{E}[\mathcal{A}_1] + 2y\mathbb{E}[\mathcal{A}_2] & \text{if } (x,y) \in \mathcal{B}, \end{cases}
\end{aligned} \tag{2.12}
$$

where $\mathcal{B} = \{(x,y) : (x = 0, y \leq L_2) \text{ or } (x \leq L_1, y = 0)\}$. The inequality (2.10) comes from $(U_t^{(i)})^2 \leq 1$ and $(Q_t^{(i)} - a_t)U_t^{(i)} \leq 0$, while $\mathbb{E}[a_t] \leq 1$ results in Eq. (2.11). Since $\mathbb{E}[\mathcal{A}_i^2] < \infty$ and $\mathbb{E}[\mathcal{A}_i] < 1$ for $i = 1, 2$, the value in Eq. (2.12) is negative for $(x,y) \in \mathcal{B}^c$ and is bounded for $(x,y) \in \mathcal{B}$. Then the result immediately follows from Foster-Lyapunov theorem. □

We realize that if $\mathbb{E}[\mathcal{A}_i^2] < \infty$ and $\mathbb{E}[\mathcal{A}_i] < 1$ for $i = 1, 2$, then there exists a stationary threshold type policy that is average-optimal and can be obtained from the reduced Markov chain. The following theorem gives an example of how to compute the optimal thresholds.

Theorem 22. *Consider the Bernoulli arrival process. The optimal thresholds L_1^* and L_2^* are*

$$
(L_1^*, L_2^*) = \arg\min_{L_1, L_2} C_T \mathcal{T}(L_1, L_2) + C_H \mathcal{H}(L_1, L_2),
$$

where

$$
\begin{aligned}
\mathcal{T}(L_1, L_2) &= p_1^{(1)} p_1^{(2)} \pi_{0,0} + p_1^{(2)} \sum_{i=1}^{L_1} \pi_{i,0} + p_1^{(1)} \sum_{j=1}^{L_2} \pi_{0,j} + p_1^{(1)} p_0^{(2)} \pi_{L_1,0} + p_0^{(1)} p_1^{(2)} \pi_{0,L_2}; \\
\mathcal{H}(L_1, L_2) &= \sum_{i=1}^{L_1} i\pi_{i,0} + \sum_{j=1}^{L_2} j\pi_{0,j},
\end{aligned}
$$

for which

$$\begin{aligned}
\pi_{0,0} &= \frac{1}{\left(\frac{1-\zeta^{L_1+1}}{1-\zeta}\right)+\left(\frac{1-1/\zeta^{L_2+1}}{1-1/\zeta}\right)-1};\\
\pi_{i,0} &= \zeta^i\pi_{0,0};\\
\pi_{0,j} &= \pi_{0,0}/\zeta^j;\\
\zeta &= \frac{p_1^{(1)}p_0^{(2)}}{p_0^{(1)}p_1^{(2)}}.
\end{aligned}$$

Proof. Let $Y_t^{(i)}$ be the number of type i packets at the t^{th} slot *after* transmission. It is crucial to note that this observation time is different from when the MDP is observed. Then the bivariate stochastic process $\{(Y_t^{(1)}, Y_t^{(2)}), t \geq 0\}$ is a discrete-time Markov chain which state space is smaller than the original MDP, i.e., $(0,0)$, $(1,0)$, $(2,0)$, $\cdots$, $(L_1,0)$, $(0,1)$, $(0,2)$, $\cdots$, $(0,L_2)$. Define ζ as a parameter such that

$$\zeta = \frac{p_1^{(1)}p_0^{(2)}}{p_0^{(1)}p_1^{(2)}}.$$

Then, the balance equations for $0 < i \leq L_1$ and $0 < j \leq L_2$ are:

$$\begin{aligned}
\pi_{i,0} &= \zeta\pi_{i-1,0};\\
\zeta\pi_{0,j} &= \pi_{0,j-1}.
\end{aligned}$$

Since $\pi_{0,0} + \sum_{i,j}\pi_{i,0} + \pi_{0,j} = 1$, we have

$$\pi_{0,0} = \frac{1}{\left(\frac{1-\zeta^{L_1+1}}{1-\zeta}\right)+\left(\frac{1-1/\zeta^{L_2+1}}{1-1/\zeta}\right)-1}.$$

The expected number of transmissions per slot is

$$\mathcal{T}(L_1, L_2) = p_1^{(1)} p_1^{(2)} \pi_{0,0} + p_1^{(2)} \sum_{i=1}^{L_1} \pi_{i,0} + p_1^{(1)} \sum_{j=1}^{L_2} \pi_{0,j} + p_1^{(1)} p_0^{(2)} \pi_{L_1,0} + p_0^{(1)} p_1^{(2)} \pi_{0,L_2}.$$

The average number of packets in the system at the beginning of each slot is

$$\mathcal{H}(L_1, L_2) = \sum_{i=1}^{L_1} i\pi_{i,0} + \sum_{j=1}^{L_2} j\pi_{0,j}.$$

Thus upon minimizing we get the optimal thresholds L_1^* and L_2^*. □

Whenever $C_H > 0$, it is relatively straightforward to obtain L_1^* and L_2^*. Since it costs C_T to transmit a packet and C_H for a packet to wait for a slot, it would be better to transmit a packet than make a packet wait for more than C_T/C_H slots. Thus L_1^* and L_2^* would always be less than C_T/C_H. Hence, by completely enumerating between 0 and C_T/C_H for both L_1 and L_2, we can obtain L_1^* and L_2^*. One could perhaps find faster techniques than complete enumeration, but it certainly serves the purpose.

Subsequently, we study a special case, $p_1^{(1)} = p_1^{(2)} \triangleq p$, in Theorem 22. Then $L_1 = L_2 \triangleq L$ as both arrival processes are identical. It can be calculated that $\zeta = 1$ and $\pi_{i,j} = 1/(2L+1)$ for all i, j, and

$$\mathcal{T}(L) = \frac{2pL + 2p - p^2}{2L+1};$$

$$\mathcal{H}(L) = \frac{L^2 + L}{2L+1}.$$

Define $\nu = C_T/C_H$. The optimal threshold is

$$L^*(p,\nu) = \arg\min_L \frac{\nu(2pL + 2p - p^2) + L + L^2}{2L+1}.$$

By taking the derivative, we obtain that $L^* = 0$ if $\nu < 1/(2p - 2p^2)$ and otherwise,

$$L^*(p,\nu) = \frac{-1 + \sqrt{1 - 2(1 - 2\nu p + 2\nu p^2)}}{2}.$$

We can observe that $L^*(p,\nu)$ is a concave function w.r.t. p. Given ν fixed, $L^*(1/2,\nu) = (\sqrt{\nu - 1} - 1)/2$ is the largest optimal threshold among various values of p. When $p < 1/2$, the optimal-threshold decreases as there is a relatively lower probability for packets in one queue to wait for a coding pair in another queue. When $p > 1/2$, there will be a coding pair already in the relay node with a higher probability, and therefore the optimal-threshold also decreases. Moreover, $L^*(1/2,\nu) = \mathcal{O}(\sqrt{\nu})$, so the maximum optimal threshold grows with the square root of ν, but not linearly. When p is very small, $L^*(p,\nu) = \mathcal{O}(\sqrt{\nu p})$ grows slower than $L^*(1/2,\nu)$. Figure 2.4 depicts the optimal threshold $L^*(p,\nu)$ for various values of arrival rate p, and $\nu = C_T/C_H$.

2.6 Numerical results

In this subsection we present several numerical results to compare the performance of different policies in the single relay setting as well as in the line network. We analyzed the following policies:

1. Opportunistic Coding (OC): this policy does not waste any opportunities for transmitting the packets. That is when a packet arrives, coding is performed if a coding opportunity exists, otherwise transmission takes place immediately.

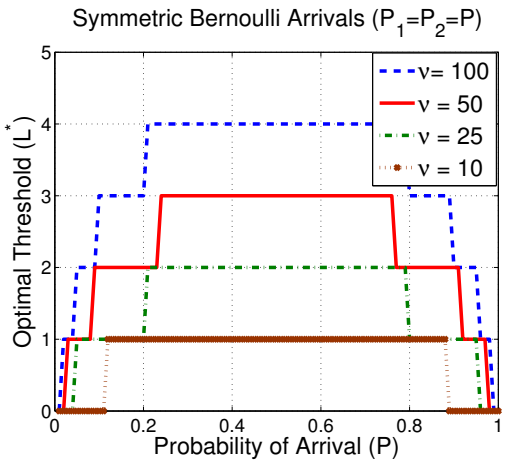

Figure 2.4: Optimal queue length threshold for a single relay with symmetric Bernoulli arrivals.

2. Queue-length based threshold (QLT): this stationary deterministic policy applies the thresholds, proposed by Theorem 22, on the queue lengths.

3. Queue-length-plus-Waiting-time-based (QL+WT) thre-sholds: this is a history dependent policy which takes into account the waiting time of the packets in the queues as well as the queue lengths. That is a packet will be transmitted (without coding), if the queue length hits the threshold or the head-of-queue packet has been waiting for at least some pre-determined amount of time. The optimal waiting-time thresholds are found using exhaustive search through stochastic simulations for the given arrival distributions.

4. Waiting-time (WT) based threshold: this is another history dependent policy that *only* considers the waiting times of the packets, in order to schedule the transmissions. The optimum waiting times of the packets are found through exhaustive search.

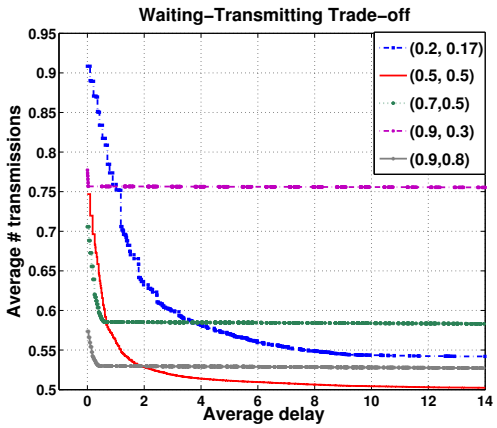

Figure 2.5: Trade-off between average delay and number of transmissions in a single relay using queue-length based threshold (QLT) policy for different Bernoulli arrival rates (p_1, p_2).

We simulate these policies on two different cases: (i) the single relay network with Bernoulli arrivals (Fig. 2.5, 2.6, and 2.7) and (ii) a line network with 4 nodes, in which the sources are Bernoulli (Fig. 2.8, and 2.9). Note that in case (ii), since the departures from one queue determine the arrivals into the other queue, the arrival processes are significantly different from Bernoulli. As expected, for the single relay network, the QLT policy has the optimal performance and the QL+WT policy does not have any advantage.

Moreover, there are results (see [31]) that indicate that the independent arrivals model is accurate under heavy traffic for multi-hop networks. Hence, our characterization of the optimal policy does have value in a more general case. Our simulation results indicate that QLT policy also exhibits a near optimal performance for the line network. We also observe, from the simulation results for the waiting-time-based policy, that making decisions based on waiting time alone leads to a suboptimal performance. In all experiments, the opportunistic policy has the worst possible

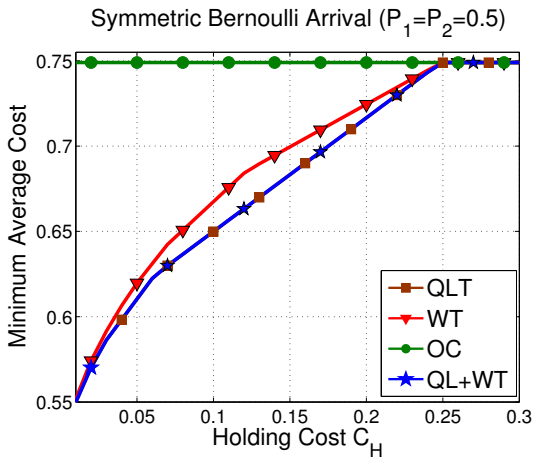

Figure 2.6: Comparison of the minimum average cost (per slot) in a single relay with Bernoulli arrival rates $(0.5, 0.5)$, for different policies, where the costs are normalized by the transmission cost.

performance.

The results are intriguing as they suggest that achieving a near-perfect trade-off between waiting and transmission costs is possible using simple policies; moreover, coupled with optimal network-coding aware routing policies like the one in our earlier work [12], have the potential to exploit the positive externalities that network coding offers.

2.7 Extensions

We have seen that the average-optimal policy is stationary and threshold based for the i.i.d. arrival process with the service rate of 1 packet per time slot. Two more general models are discussed here. We focus on the character of the optimality equation which results in the structure of the average-optimal policy.

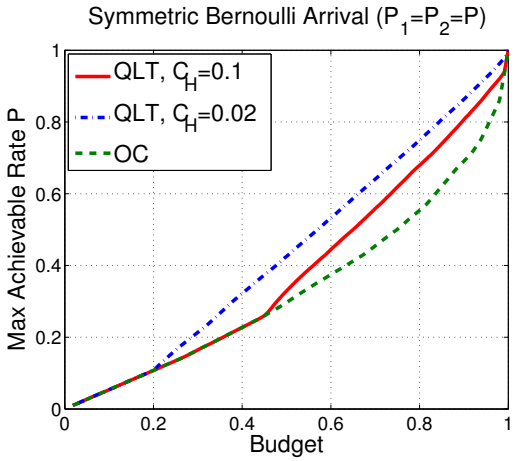

Figure 2.7: Achievable arrival rate versus average budget (per slot) in a single relay with Bernoulli arrivals, using opportunistic coding (OC) and QLT policies, where the costs are normalized by the transmission cost.

2.7.1 Batched service

Assume that the relay R can serve a group of packets with the size of $\mathcal{M}$ at end of the time slot. At the end of every time slot, relay R decides to transmit, $a_t = 1$, or to wait $a_t = 0$. The holding cost per unit time for a packet is C_H, while C_T is the cost to transmit a batched packet. Then the immediate cost is

$$C^{(\mathcal{M})}(Q_t, a_t) = C_H([Q_t^{(1)} - a_t\mathcal{M}]^+ + [Q_t^{(2)} - a_t\mathcal{M}]^+) + C_T a_t.$$

We also want to find the optimal policy θ^* that minimizes the long-time average cost $V^{(\mathcal{M})}(\theta)$, called $\mathcal{M}$-MDP$\{(Q_t, a_t), t \geq 0\}$ problem,

$$V^{(\mathcal{M})}(\theta) = \lim_{K\to\infty} \frac{1}{K+1}\mathbb{E}_\theta \left[\sum_{t=0}^{K} C^{(\mathcal{M})}(Q_t, a_t)|Q_0 = (0,0)\right].$$

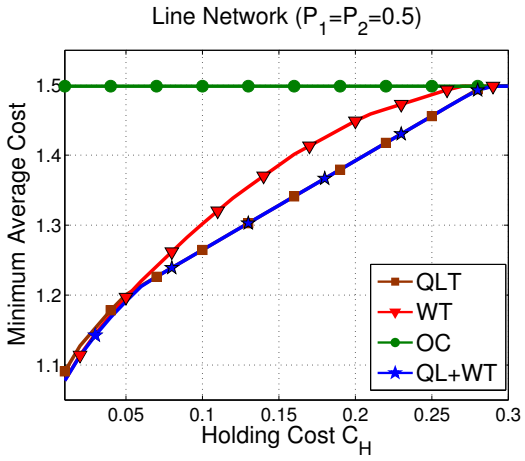

Figure 2.8: Comparison of different policies in a line network with two intermediate nodes and two Bernoulli flows with mean arrival rates $(0.5, 0.5)$.

Notice that the best policy might not just transmit when both queues are non-empty. When $\mathcal{M} > 1$, R might also want to wait even if $Q_t^{(1)}Q_t^{(2)} > 0$ because the batched service of size less than $\mathcal{M}$ has the same transmission cost C_T. The optimality equation of the expected α-discounted cost is revised as

$$V_\alpha^{(\mathcal{M})}(i,j) = \min_{a\in\{0,1\}}\Big[C_H([i-a\mathcal{M}]^+ + [j-a\mathcal{M}]^+) + C_T a + \mathbb{E}[V_\alpha^{(\mathcal{M})}([i-a\mathcal{M}]^+ + \mathcal{A}_1, [j-a\mathcal{M}]^+ + \mathcal{A}_2)]\Big].$$

We can get the following results.

Theorem 23. *Given α and $\mathcal{M}$, $V_\alpha^{(\mathcal{M})}(i,j)$ is non-decreasing, submodular, and $\mathcal{M}$-subconvex. Moreover, there is an α-optimal policy that is of threshold type. Fixed j, the α-optimal policy is monotone w.r.t. i, and vice versa.*

Theorem 24. *Consider any i.i.d. arrival processes to both queues. For the $\mathcal{M}$-MDP$\{(Q_t, a_t), t \geq 0\}$, the average-optimal policy is of threshold type. Given $j =$*

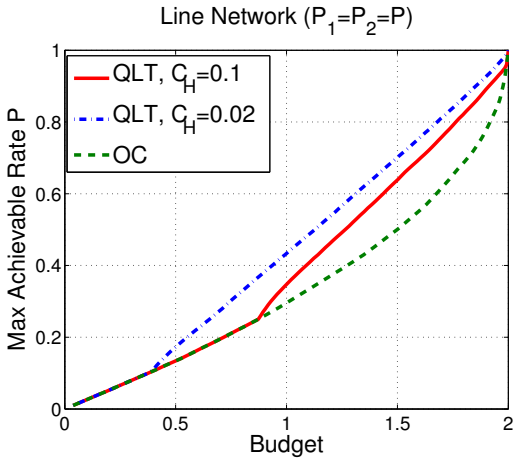

Figure 2.9: Achievable arrival rate versus average budget (per slot) in a line network with two intermediate nodes and two Bernoulli flows.

$\tilde{j}$ fixed, there exists the optimal threshold $L^*_{\tilde{j}}$ such that the optimal stationary and deterministic policy in state $(i, \tilde{j})$ is to wait if $i \leq L^*_{\tilde{j}}$, and to transmit if $i > L^*_{\tilde{j}}$. Similar argument holds for the other queue.

2.7.2 Markov-modulated arrival process

While the i.i.d. arrival process is examined so far, a specific arrival process with memory is studied here, i.e., Markov-modulated arrival process (MMAP). The service capacity of R is focused on $\mathcal{M} = 1$ packet. Let $\mathcal{N}^{(i)} = \{0, 1, \cdots, N^{(i)}\}$ be the state space of MMAP at node i, with the transition probability $p^{(i)}_{k,l}$ where $k, l \in \mathcal{N}^{(i)}$. Then the number of packets generated by the node i at time t is $\mathcal{N}^{(i)}_t \in \mathcal{N}^{(i)}$. Then the decision of R is made based on the observation of $(Q^{(1)}_t, Q^{(2)}_t, \mathcal{N}^{(1)}_t, \mathcal{N}^{(2)}_t)$. Similarly, the objective is to find the optimal policy that minimizes the long-term average cost, named MMAP-MDP$\{((Q^{(1)}_t, Q^{(2)}_t, \mathcal{N}^{(1)}_t, \mathcal{N}^{(2)}_t), a_t) : t \geq 0\}$ problem. The optimality

equation of the expected α-discounted cost becomes

$$
\begin{aligned}
& V_{\alpha}^{\mathrm{MMAP}}(i,j,n_1,n_2) \\
&= \min_{a\in\{0,1\}} [C_H([i-a]^+ + [j-a]^+) + C_T a + \\
&\quad \alpha \sum_{k=0}^{N^{(1)}} \sum_{l=0}^{N^{(2)}} p_{n_1,k}^{(1)} p_{n_2,l}^{(2)} V_{\alpha}^{\mathrm{MMAP}}([i-a]^+ + k, [j-a]^+ + l, k, l)].
\end{aligned}
$$

Then we can conclude the following results.

Theorem 25. *Given $n_1 \in \mathcal{N}^{(1)}$ and $n_2 \in \mathcal{N}^{(2)}$, $V_{\alpha}^{MMAP}(i,j,n_1,n_2)$ is non-decreasing, submodular, and subconvex w.r.t. i and j. Moreover, there is an α-optimal policy that is of threshold type. Fixed n_1 and n_2, the α-optimal policy is monotone w.r.t. i when j is fixed, and vice versa.*

Theorem 26. *Consider an MMAP arrival. For the MMAP-MDP$\{((Q_t^{(1)}, Q_t^{(2)}, \mathcal{N}_t^{(1)}, \mathcal{N}_t^{(2)}), a_t) : t \geq 0\}$, the average-optimal policy is of multiple thresholds type. There exists a set of optimal thresholds $\{L^*_{1,n_1,n_2}\}$ and $\{L^*_{2,n_1,n_2}\}$, where $n_1 \in \mathcal{N}^{(1)}$ and $n_2 \in \mathcal{N}^{(2)}$, so that the optimal stationary decision in states $(i,0,n_1,n_2)$ is to wait if $i \leq L^*_{1,n_1,n_2}$, and to transmit without coding if $i > L^*_{1,n_1,n_2}$; while in state $(0,j,n_1,n_2)$ is to wait if $j \leq L^*_{2,n_1,n_2}$, and to transmit without coding if $j > L^*_{2,n_1,n_2}$.*

2.7.3 Time-varying channel

We will examine the scenario in which the relay transmits packets over time-varying ON/OFF channels, while we assume that the arrivals are i.i.d. and the relay can serve at most one packet for each time slot. Let $S_t = (S_t^{(1)}, S_t^{(2)})$ be the channel state at time t, where $S_t^{(i)} \in \{0(\mathrm{OFF}), 1(\mathrm{ON})\}$ indicates the channel condition from the relay to node i. We assume that the channel states are i.i.d. over time. Moreover, when $S_t^{(i)} = 1$, to transmit a packet from the relay to node i takes the cost of $C_T^{(i)}$.

Then the immediate cost $C(Q_t, S_t, a_t)$ is

$$C(Q_t, S_t, a_t) = C_H([i - a_t S_t^{(1)}]^+ + [j - a_t S_t^{(2)}]^+) + \max(a_t S_t^{(1)} C_T^{(1)}, a_t S_t^{(2)} C_T^{(2)}).$$

The objective is also to find the optimal policy that minimizes the long-run average cost. The optimality equation of the expected α-discounted cost becomes

$$\begin{aligned} & V_\alpha(i, j, s_1, s_2) \\ = & \min_{a \in \{0,1\}} \big[C_H([i - a s_1]^+ + [j - a s_2]^+) + \max(a s_1 C_T^{(1)}, a s_2 C_T^{(2)}) + \\ & \qquad \alpha \mathbb{E}[V_\alpha([i - a s_1]^+ + \mathcal{A}_1, [j - a s_2]^+ + \mathcal{A}_2), S_t^{(1)}, S_t^{(2)}] \big]. \end{aligned}$$

Then we conclude the following results.

Theorem 27. *$V_\alpha(i, j, 1, 1)$ is non-decreasing, submodular, and subconvex. $V_\alpha(i, j, 1, 0)$ is convex in i for any fixed j and $V_\alpha(i, j, 0, 1)$ is convex in j for any fixed i. Moreover, there is an α-optimal policy that is of threshold type. For each channel state, the α-optimal policy is monotone in i for fixed j, and vice versa.*

Theorem 28. *Consider any i.i.d. arrivals to both queues and time-varying ON/OFF channels. The average-optimal policy is of threshold type. For state (s_1, s_2), there exist the optimal thresholds L^*_{1,s_1,s_2} and L^*_{2,s_1,s_2} so that the optimal deterministic action in states $(i, 0)$ is to wait if $i \leq L^*_{1,s_1,s_2}$, and to transmit without coding if $i > L^*_{1,s_1,s_2}$; while in state $(0, j)$ is to wait if $j \leq L^*_{2,s_1,s_2}$, and to transmit without coding if $j > L^*_{2,s_1,s_2}$.*

3. OPPORTUNISTIC NETWORK CODING: COMPETITIVE ANALYSIS*

3.1 Introduction

In recent years, there has been a growing interest in the applications of network coding techniques in multi-hop wireless networks. It was shown that network coding can significantly minimize the number of transmissions resulting in energy savings and, in turn, longer battery life. The *reverse carpooling* technique allows to realize the benefits of network coding while requiring only simple encoding and decoding algorithms.

The reverse carpooling technique for a simple relay network is demonstrated in Fig. 3.1(a). In this example, nodes n_1 and n_2 exchange packets through a relay node R. The traditional approach that does not involve network coding requires four transmissions (two for each packet). In contrast, the network coding approach only requires three transmissions. In particular, with the network coding approach nodes n_1 and n_2 send packets x_1 and x_2 to the relay node, and the relay node broadcasts the linear combination $x_1 + x_2$ (e.g., over field $GF(2^n)$) to both n_1 and n_2. Each of the nodes n_1 and n_2 can then obtain the packet it needs by subtracting the packet it transmitted previously from the mixed packet $x_1 + x_2$.

In a more general case, reverse carpooling includes two flows that traverse a path in opposite directions. For example, Fig. 3.1(b) shows two flows, from n_1 to n_4 and from n_4 to n_1 that share a common path (n_1, n_2, n_3, n_4) with two relay nodes n_2 and n_3. It is easy to verify that the network coding approach can save up to 50% of transmissions. Effectively, once one connection has been established, another

* Part of the data reported in this section is reprinted with permission from "Opportunistic Network Coding: Competitive Analysis" by Yu-Pin Hsu and Alex Sprintson, 2012. In *International Symposium on Network Coding (NETCOD)*, 191-196, Copyright 2012 by IEEE.

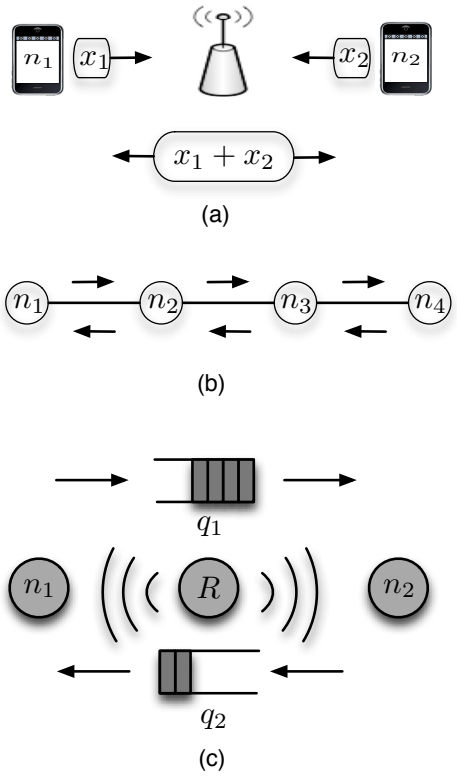

Figure 3.1: (a) Wireless network coding (b) Reverse carpooling (c) 3 nodes relay network.

connection in the opposite direction can be supported with small additional cost.

To achieve the maximum benefit from the network coding, we need to make sure that the relay nodes can create a sufficient number of coded packets. Note that a relay node can only create a coded packet when it has at least one packet to transmit in each direction. More specifically, consider relay node R depicted in Fig. 3.1(c). Relay R maintains two queues, q_1 and q_2 that store packets that need to be delivered to nodes n_2 and n_1, respectively. If both queues are not empty, then R can construct a coded packet by combining the packets from the top of both queues. However, what should the relay do if there are packets waiting to be transmitted in one of the queues,

while the second queue is empty? Should the relay wait for a coding opportunity or just transmit a packet from a non-empty queue without coding? Waiting for a coding opportunity would reduce the number of transmissions, but incur a certain penalty in terms of delay. Transmitting an uncoded packet would minimize the delay, but would result in a larger number of transmissions.

In this section we present an on-line algorithm for this problem that strikes a trade-off between the average delay and the number of transmissions. We analyze the performance of this algorithm using the *competitive analysis* [32] techniques which characterize the performance of the algorithm in the worst-case scenario.

3.1.1 Related work

Network coding research was initiated by the seminal work of Ahlswede et al. [6] and since then attracted major interest from the research community. Network coding technique for wireless networks has been considered by Katti et al. [7]. They propose an architecture, referred to as COPE, which contains a special network coding layer between the IP and MAC layers. In [8], an opportunistic routing protocol is proposed, referred to as MORE, that randomly mixes packets that belong to the same flow before forwarding them to the next hop. In addition, several works, e.g., [10–15], investigate the scheduling and/or routing problems in the network coding enabled networks. Sagduyu and Ephremides [10] focus on the network coding in tandem networks and formulate a cross-layer optimization problem, while Khreishah et al. [11] devise a joint coding-scheduling-rate controller when the pairwise intersession network coding is allowed. The paper [12] shows how to design coding-aware routing controllers that would maximize coding opportunities in multihop networks. References [13] and [14] attempt to schedule the network coding between multiple-session flows. Xi and Yeh [15] propose a distributed algorithm that minimizes the

transmission cost of a multicast session.

The works of [16–18] analyze the similar trade-off between power consumption and packet delays from different perspec- tives. The authors in [16] propose a threshold policy using the Lyapunov technique. The threshold policy in [16] is an approximate solution with some performance guarantees. The paper [17] investigates the problem using an Markov decision process (MDP) framework. Some implementation issues are discussed in [18]. In previous section, we theoretically prove the structure of the optimal policy by formulating a stochastic optimization problem. However, the relevant papers [16–18] and the previous section assumes an independent and identical distributed (i.i.d.) arrival process. In contrary, our purpose is to come up with an on-line and universal algorithm that works with any arrival process.

The ski-rental problem [32] is a classical on-line problem, in which there is a decision for each time epoch about either continuing renting skis or buy skis. Our wait/transmission decision problem can be viewed as a generalization of the classcial ski-rental problem.

3.1.2 Main results

We develop on-line algorithms that make a decision on whether to transmit or to wait at each time epoch. The objective of the algorithm is to minimize the sum of the transmission and waiting time costs. Thus, at any time slot, the decision of transmitting a packet would incur a certain transmission cost, while the decision to wait would incur a waiting cost, but will possibly allow to share the transmission cost between two packets in the future.

In contrary to the previous section, where the statistical characteristics of the arrival process are assumed to be known, in this section we analyze the problem from the viewpoint of competitive analysis. We propose an on-line algorithm for

the relay that makes wait/transmit decisions without the information of the future arrivals.

The performance of the on-line algorithm is analyzed using the *competitive analysis techniques* [33]. The proposed algorithms can achieve the competitive ratio of $e/(e-1)$.

3.2 System overview

3.2.1 Scenario from a relay's perspective

Our first focus is on the case of a single relay node of interest, which has the potential for network coding packets from flows in opposing directions. Consider Fig. 3.1(c) again. We assume that there is a flow f_1 that goes from node 1 to 2 and another flow f_2 from node 2 to 1, both of which are through the relay under consideration. The packets from both flows are stored at separate queues, q_1 and q_2, at relay node R.

In this section, we consider a time division multiple access (TDMA) scheme. Time is divided into slots that are further divided into 3 mini-slots. In each slot, each node is allowed to transmit in its assigned mini-slot: node 1 uses the first mini-slot and node 2 uses the second mini-slot, while the last mini-slot in a slot is used by the relay. In particular, the time period between transmission opportunities for the relay is precisely one slot. Our model is consistent with the scheduled and time synchronized scheme, such as LTE. Moreover, we use slot as the unit of packet delays. We assume if a packet is transmitted in the same slot when it arrived at the relay, its latency is zero.

Note that the relay gets an opportunity to transmit at the end of the slot. When both queues are non-empty, the relay pulls one packet from q_1 and one from q_2 and transmits a linear combination of the packets (e.g., their sum over $GF(2^n)$). Note

that in this case, the relay is able to take the advantage of the network coding, reducing the number of transmitted packets by one. We refer to this case a *coding opportunity*. Note that if both of the queues are empty, that the relay will not able to transmit during its mini-slot. In the case when one of the queues is empty and the second one is not, the relay has to make a decision between the following two options:

1. transmit a packet from an non-empty queue without coding;
2. wait for the next slot in the hope to receive a matching packet in the other queue and be able to utilize the coding opportunity.

Note that the first option increases the number of transmissions, while the second option increases packet delay in the hope of reducing the number of transmissions.

3.2.2 Competitive analysis

We develop an on-line algorithm for the decision problem at the relay and analyze it using the competitive analysis techniques. Note that relay has to make its decisions without the knowledge of the future arrivals. For $i \in \{1, 2\}$ and $t = 0, 1, 2, \cdots$, let $Q_t^{(i)}$ be the number of packets stored at queue q_i at slot t, just before the beginning of the relay's mini-slot. We denote by $Q_t = (Q_t^{(1)}, Q_t^{(2)})$ the state of the system. We also introduce the indicator $D_t \in \{0, 1\}$ such that $D_t = 0$ if the decision at time slot t was to wait and $D_t = 1$ if the decision at time slot t was to transmit. As discussed above, if $Q_t^{(1)} = Q_t^{(2)} = 0$, then $D_t = 0$. Also, if $Q_t^{(1)} > 0$ and $Q_t^{(2)} > 0$, then $D_t = 1$. When exactly one of $Q_t^{(1)}$ and $Q_t^{(2)}$ is non-zero, then the on-line algorithm has to make a decision on whether to transmit an uncoded packet or to wait.

We use $A_t^{(i)} \in \{0, 1\}$ to indicate an arrival of the packet to q_i at slot t, such that $A_t^{(i)} = 0$ if no packet arrived at slot t and $A_t^{(i)} = 1$ otherwise. The *arrival pattern*

$I = \{(A_t^{(1)}, A_t^{(2)})\}_{t=0}^{\infty}$ captures the arrival of packets to both queues. We assume that there will be a finite number of packet arrivals to both queues.

The number of packets in at slot $t+1$ can be expressed as follows:

$$Q_{t+1}^{(i)} = [Q_t^{(i)} - D_t]^+ + A_{t+1}^{(i)},$$

where $[x]^+ = \max(x, 0)$.

We proceed to define transmission and delay costs. We define by C_T the cost of transmitting a packet and C_H be the cost for holding a packet for one slot. We assume that if a packet is transmitted in the same slot it arrived, its delay is zero. Clearly, the cost of transmitting a coded packet is the same as that of an uncoded packet.

Let $C(Q_t, D_t)$ be the *immediate cost* incurred at time t if action D_t is taken when the system is in state Q_t:

$$C(Q_t, D_t) = C_H([Q_t^{(1)} - D_t]^+ + [Q_t^{(2)} - D_t]^+) + C_T D_t. \tag{3.1}$$

Without loss of generality, we assume that $C_H = 1$.

We define an *algorithm* $\theta = \{D_0, D_1, \dots\}$ as a sequence of decisions made at different time slots. An algorithm θ can be deterministic or randomized, where a randomized algorithm is a distribution over deterministic algorithms. Note that with a randomized algorithm, $\{D_t | t = 0, 1, \cdots\}$ are random variables. For the given arrival pattern $I = \{(A_t^{(1)}, A_t^{(2)})\}_{t=0}^{\infty}$ to q_1 and q_2, the total cost $V(I, \theta)$ of a deterministic algorithm θ at time T is defined as:

$$V(I, \theta) = \sum_{t=0}^{\infty} C(Q_t, D_t).$$

For a randomized algorithm, we define the *expected cost* $V(I, \theta)$ of θ as follows:

$$V(I, \theta) = \mathbb{E}_\theta \left[\sum_{t=0}^{\infty} C(Q_t, D_t) \right],$$

where the expectation is over the randomized policy θ.

An *off-line* algorithm knows the arrival pattern in advance. We denote by $OPT(I)$ the cost of an optimal off-line algorithm, i.e., algorithm that minimizes $V(I, \theta)$. In contrast to off-line algorithms, an *on-line* algorithm makes decisions at time t without knowing $A_j^{(i)}, i \in \{1, 2\}$ for $j > t$.

An on-line algorithm θ is said to be c-competitive if for every input pattern I it holds that

$$V(I, \theta) \leq c \cdot OPT(I) + \alpha,$$

where α is a constant independent of I.

3.2.3 Remark

In Subsections 3.3 - 3.5, we focus on the following setting illustrated in Fig. 3.2.

- q_1 has $\mathcal{K}$ packets at time 0 (i.e., $Q_0^{(1)} = \mathcal{K}$);
- there are no arrivals to q_1 (i.e., $A_t^{(1)} = 0$ for all t);
- q_2 has zero packets at time 0, (i.e., $Q_0^{(2)} = 0$).
- the adversary controls the arrival process of exactly $\mathcal{K}$ packets to q_2.

We assume that $\mathcal{K} \leq C_T$; otherwise, there is no potential benefit for the relay to wait for a coding opportunity. We propose an on-line algorithm for this setting that achieves the competitive ratio of $e/(e-1)$. Subsections 3.6 and 3.7 extend the

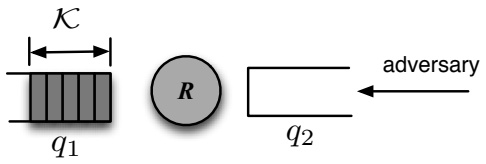

Figure 3.2: One adversarial process to q_1.

algorithm for a more general model in which the adversary controls arrivals to the both queues.

3.3 Primal-dual formulation

We need to introduce some additional notation. Consider a deterministic algorithm θ. First, let x be the number of packets in q_1 transmitted by the algorithm without coding. We also denote by z_t the number of packets waiting in q_1 at time t and by $\eta_t = \sum_{\tau=0}^{t} A_\tau^{(2)}$ we define the number of packets added to q_2 by the adversary by time t.

We observe that without loss of generality, we can assume that the algorithm will never delay packets that arrive to q_2. Indeed, since there are no arrivals to q_1 delaying a packet in q_2 only contributes to the increase in the overall waiting time.

The cost $C(I, \theta)$ of θ can be expressed as:

$$\sum_{t=0}^{\infty} C(Q_t, D_t) = C_T \cdot \mathcal{K} + C_T \cdot x + \sum_{t=0}^{\infty} z_t. \tag{3.2}$$

Note that the first term in (3.2) covers the cost of transmitting all packets in q_2 that were sent uncoded, as well as the cost of coded packets. The second term, $C_T \cdot x$ covers the cost of transmitting uncoded packets from q_1. Finally, the term $\sum_{t=0}^{\infty} z_t$ covers the cost associated with waiting time of packets in q_1.

The off-line decision problem can be formulated as the following integer program:

Primal program:

$$\min \quad C_T \cdot x + \sum_{t=0}^{\infty} z_t \tag{3.3}$$

$$\text{s.t.} \quad x + z_t \geq \mathcal{K} - \eta_t \quad \forall t \geq 0 \tag{3.4}$$

$$x, z_t \in \{0, 1, 2, \cdots\} \ \forall t \geq 0 \tag{3.5}$$

Note that the objective function (3.3) is identical to (3.2), excluding the constant term $C_T \cdot \mathcal{K}$. Constraint (3.4) specifies that the number of packets stored at q_1 at time slot t is at least $\mathcal{K} - \eta_t - x$. Indeed, out of $\mathcal{K}$ packets that were at queue q_1 in the beginning of the algorithm, at most x could be sent by time t without coding and at most η_t could be sent as a combination of packets from q_2.

The integer formulation can be relaxed by substituting the integrality constraint (3.5) with a non-negativity constraint $x, z_t > 0$ for all $t \geq 0$.

Proposition 29. *The relaxation has no integrality gap.*

Proof. Let x^* and z_t^* be the optimal solution to the relax-LP. Then $z_t^* = \mathcal{K} - \eta_t - x^*$. Suppose the optimal solution is non-integral $x^* = \overline{x} + \epsilon$, where $\overline{x} \in \mathbb{N} \cup \{0\}$ and $0 < \epsilon < 1$. Then the optimal value in Eq. (3.3) is $C_T \cdot (\overline{x} + \epsilon) + \sum_{t=0}^{\infty} \mathcal{K} - \eta_t - \epsilon$. Because of $\epsilon > 0$, the optimal solution is $-\infty$, which leads to a contradiction. We can get the similar contradiction when we assume that $x^* = \overline{x} - \epsilon$. Hence, we conclude that the optimal solution to the relax-LP is integral, and therefore there is no integrality gap when relaxing the integer program. □

The dual program of the resulting linear program can be formulated as follows:

Dual program:

$$\max \quad \sum_{t=0}^{\infty} (\mathcal{K} - \eta_t) y_t \tag{3.6}$$

$$\text{s.t.} \quad \sum_{t=0}^{\infty} y_t \leq C_T \tag{3.7}$$

$$0 \leq y_t \leq 1 \text{ for all } t \tag{3.8}$$

In the following subsection, we use the framework developed by [33] to construct a randomized algorithm for the problem at hand. The algorithm is based on the primal-dual formulation developed in this section.

3.4 Fractional algorithm

3.4.1 Algorithm description

The off-line algorithm requires the prior knowledge of $\{A_t^{(2)}\}$. This implies that the algorithm knows at time 0 the values of η_t for all $t \geq 0$. In an on-line setting, at each time $t = 0, 1, \cdots$, the algorithm only knows the value of $\eta_{t'}$ for $t' \leq t$. Thus, the on-line algorithm does not know in advance all the constraints specified by (3.4), they are revealed to the algorithm one constraint per time slot.

The purpose of **Algorithm 1** is to compute a feasible fractional solution to the primary problem in an on-line fashion. At each slot t, the algorithm obtains the value of $A_t^{(2)}$ which indicates whether an adversary decided to send a new packet to q_2. The algorithm computes the value of z_t, as well as updates the current value of x such that constraint that corresponds to slot t in the primal program is satisfied. The output of the algorithm is used by **Algorithm 2** (described below) which computes a randomized integral policy at each time interval.

Since the design of both algorithms follow the primal-dual framework presented in [33] we present only a sketch of the algorithms.

Algorithm 1 uses additional variables $\tilde{x}$, $\{x_i\}_{i=1}^{\mathcal{K}}$, and $\tilde{z}_t$ for each time t. The constant a in Line 14 is chosen to make the dual solution feasible. For each new time slot t and the associated η_t, only $\{x_i\}_{i=\eta_t+1}^{\mathcal{K}}$ are updated while $\{x_i\}_{i=1}^{\eta_t}$ remain the same. The extra vector of the variables $\{\chi_t, t = 0, 1, \cdots\}$ records the value of x at slot t, and it is used later on by **Algorithm 2**.

Algorithm 1: Fractional Primal-Dual algorithm

1 z_t, y_t, $\eta_t \leftarrow 0$ for all t ;
2 x, $\tilde{x} \leftarrow 0$;
3 $x_1, \cdots, x_{\mathcal{K}} \leftarrow 0$;
4 $\tilde{z}_t \leftarrow 0$ for all t;
5 $a \leftarrow (1 + \frac{1}{C_T})^{C_T} - 1$;
6 For each new time slot t, the parameters are updated as following.;
7 **if** $\tilde{x} < 1$ **then**
8 | **if** $A_t^{(2)} = 1$ **then**
9 | | $\eta_t \leftarrow \eta_{t-1} + 1$;
10 | **else**
11 | | $\eta_t \leftarrow \eta_{t-1}$;
12 | **end**
13 | $\tilde{z}_t \leftarrow 1 - \tilde{x}$;
14 | $\tilde{x} \leftarrow \tilde{x}(1 + \frac{1}{C_T}) + \frac{1}{aC_T}$;
15 | **for** $i = \eta_t + 1$ **to** $\mathcal{K}$ **do**
16 | | $x_i \leftarrow \tilde{x}$;
17 | **end**
18 | $x \leftarrow \sum_{i=1}^{\mathcal{K}} x_i$;
19 | $z_t \leftarrow (\mathcal{K} - \eta_t)\tilde{z}_t$;
20 | $y_t \leftarrow 1$;
21 | $\chi_t \leftarrow x$;
22 **else**
23 | Terminate;
24 **end**

3.4.2 Correctness proof

Lemma 30. ***Algorithm 1*** *produces a feasible fractional solution to primal program.*

Proof. Let t be an arbitrary time slot, we show that the constraint specified by (3.4) that corresponds to t in the primary program (i.e., $x + z_t \geq \mathcal{K} - \eta_t$) is satisfied. We consider two cases:

- Case 1: $\tilde{x} \geq 1$. Then,

$$x = \sum_{i=1}^{\mathcal{K}} x_i \geq (\mathcal{K} - \eta_t)\tilde{x} \geq \mathcal{K} - \eta_t.$$

- Case 2: $\tilde{x} < 1$. Then, let $\bar{x}$ be the value of $\tilde{x}$ at time $t-1$, and we get

$$\begin{aligned} x + z_t &= \sum_{i=1}^{\mathcal{K}} x_i + (\mathcal{K} - \eta_t)(1 - \bar{x}) && (3.9) \\ &\geq (\mathcal{K} - \eta_t)\tilde{x} + (\mathcal{K} - \eta_t)(1 - \bar{x}) && (3.10) \\ &\geq \mathcal{K} - \eta_t. && (3.11) \end{aligned}$$

Here, the first inequality is due to the "for" loop in Line 15. The second inequality is due to the fact that $\tilde{x}$ can only increase as the algorithm proceeds. □

Lemma 31. ***Algorithm 1*** *produces a feasible fractional solution to the dual program when* $a = (1 + \frac{1}{C_T})^{C_T} - 1$.

Proof. To satisfy the dual constraint (3.7), the iterative updating process starting from Line 7 should not be allowed to run more than C_T times since y_t is assigned to be 1 at time t. Thus, it suffices to show that $\tilde{x} \geq 1$ after at most C_T time slots. Note that the increment of $\tilde{x}$ forms a geometric sequence with the ratio $1 + 1/C_T$,

and hence at the $(C_T - 1)^{\text{th}}$ time slot it holds that

$$\tilde{x} = \frac{(1 + \frac{1}{C_T})^{C_T} - 1}{a}. \tag{3.12}$$

Therefore, the dual solutions are feasible when

$$a = (1 + \frac{1}{C_T})^{C_T} - 1.$$

□

Theorem 32. *Let OPT be the optimal off-line solution to primal program. Then, the cost of the solution computed by* ***Algorithm 1*** *is upper bounded by*

$$(1 + \frac{1}{(1 + 1/C_T)^{C_T} - 1})OPT.$$

Proof. By P and D, we denote the values of objective functions of the primal and dual problem respectively. Let ΔP and ΔD be the change of the primal and dual cost between the successive updates. Let Δx be the change of x between two successive time slot. Given time t with the associated η_t, we obtain

$$\begin{aligned} \Delta P &= C_T \Delta x + z_t & (3.13)\\ &= C_T(\mathcal{K} - \eta_t)(\frac{\tilde{x}}{C_T} + \frac{1}{aC_T}) + (\mathcal{K} - \eta_t)(1 - \tilde{x}) & (3.14)\\ &= (\mathcal{K} - \eta_t)(1 + 1/a). & (3.15) \end{aligned}$$

It is easy to verify that $\Delta D = (\mathcal{K} - \eta_t)y_t = \mathcal{K} - \eta_t$. Therefore, $P = (1 + 1/a)D$ and

then $P \leq (1 + 1/a)OPT$ by the weak duality property, where

$$a = (1 + \frac{1}{C_T})^{C_T} - 1.$$

□

3.5 Randomized algorithm

3.5.1 Algorithm description

We use the fractional solution produced by **Algorithm 1** to construct a randomized algorithm, **Algorithm 2** for the original problem. The expected cost of the randomized algorithm is less than or equal to the primal cost of **Algorithm 1**. In particular, **Algorithm 2** uses the values of χ_t (output of **Algorithm 1**) to derive the probability to transmit uncoded packets. At any time t, at least $\lfloor \chi_t \rfloor$ packets need to be transmitted without coding. The algorithm uses variable τ to keep the record of the number of the uncoded packets transmitted. Clearly, if there is one packet from the adversary (i.e., $A_t^{(2)} = 1$ in Line 4), the relay transmits a coded packet.

In Line 7, if $\lfloor \chi_t \rfloor - \tau \geq 1$, then $\lfloor \chi_t \rfloor - \tau$ packets are transmitted immediately. Lines 10 and 12 compute probability for the relay to transmit a packet in q_1 in two different cases. Once the relay decides to transmit a packet without coding, then τ is increased by one in Line 18.

3.5.2 Example

To show the operation of Algorithms 1 and 2 we give an example of their execution. Assume the transmission cost $C_T = 5$ and the initial number of packets in q_1 is $\mathcal{K} = 3$. Table 3.1 shows the values computed by Algorithms 1 and 2. The arrivals to q_2 are indicated by $A_t^{(2)}$. For each time slot, the values of $\tilde{x}$, x_1, x_2, x_3 and

Algorithm 2: Randomized algorithm

input : the value of χ_t from **Algorithm 1**

$p_t \leftarrow 0$ for all t;
$\tau \leftarrow 0$;
For each new time slot t, if there is a packet in q_1, do:;
if $A_t^{(2)} = 1$ **then**
    transmit one coded packet by combining packets from q_1 and q_2;
end
if $\lfloor \chi_t \rfloor - \tau \geq 1$ **then**
    transmit $\lfloor \chi_t \rfloor - \tau$ packets from q_1;
    $\tau \leftarrow \lfloor \chi_t \rfloor$;
    $p_t \leftarrow \chi_t - \tau$;
else if $\chi_t > \tau$ **then**
    $p_t \leftarrow (\chi_t - \chi_{t-1})/[1 - (\chi_t - \tau)]$;
end
if $p_t > 0$ **then**
    Choose α uniformly random in from $[0, 1]$;
    if $\alpha \leq p_t$ **then**
        transmit 1 packet in q_1 ;
        $\tau \leftarrow \tau + 1$;
    end
end

Table 3.1: A example of execution of Algorithms 1 and 2.

Time t	0	1	2	3
$A_t^{(2)}$	0	0	0	1
$\tilde{x}$	0.13	0.3	0.49	0.72
x_1	0.13	0.3	0.49	0.49
x_2	0.13	0.3	0.49	0.72
x_3	0.13	0.3	0.49	0.72
χ_t	0.39	0.9	1.47	1.93
p_t	0.39	0.84	0.47	0.87
Decision	no	yes	no	yes
τ	0	1	1	2

χ_t are computed by **Algorithm 1**. **Algorithm 2** uses the value of χ_t to calculate the probability p_t of transmission. At time 0, $p_0 = 0.39$ means the relay transmits a packet with the probability 0.39, while "no" in Decision row shows that the final decision is to wait at time 0. At time 1, $\chi_0 = 0.9$ implies that the cumulative probability to transmit a packet without coding is 0.9, i.e., the relay decides to transmit at time 2 with the probability $(0.9 - 0.39)/(1 - 0.39) = 0.84$. At that time slot, the relay decides to transmit, therefore τ is updated to 1. At time 2, $\chi_2 = 1.47$ suggests that one packet is supposed to have been transmitted while another packet can be transmitted with the probability $p_2 = 0.47$. At time 3, a coded packet would be transmitted due to an arrival from the adversary. Moreover, only x_2 and x_3 are updated and another packet in q_1 would be transmitted with the probability 0.87.

3.5.3 Correctness proof

Theorem 33. ***Algorithm 2*** *achieves a competitive ratio of*

$$1 + \frac{1}{(1 + 1/C_T)^{C_T} - 1}.$$

The algorithm is asymptotically $e/(e-1)$-competitive as $C_T \to \infty$.

Proof. It suffices to show that the expected primal cost of the solution computed by **Algorithm 2** is smaller than the fractional solution computed by **Algorithm 1**. First notice that at slot t the expected number of transmissions without coding is equal to the number χ_t computed by **Algorithm 1**.

Given the time slot t and the associated η_t, the expected number of packets waiting in q_1 is $\mathcal{K} - \eta_t - \chi_t$, which is less than z_t obtained from **Algorithm 1** since

$$\begin{aligned} \mathcal{K} - \eta_t - \chi_t &= \mathcal{K} - \eta_t - \sum_{i=1}^{\mathcal{K}} x_i && (3.16) \\ &\leq \mathcal{K} - \eta_t - (\mathcal{K} - \eta_t)\tilde{x} && (3.17) \\ &\leq \mathcal{K} - \eta_t - (\mathcal{K} - \eta_t)\bar{x} && (3.18) \\ &= (\mathcal{K} - \eta_t)\tilde{z}_t, && (3.19) \end{aligned}$$

where $\bar{x}$ is equal to the value of $\tilde{x}$ at the previous iteration.

We conclude that the expected primal cost from **Algorithm 2** is less than that from **Algorithm 1**. □

The next lemma shows that in **Algorithm 2** the relay transmits at most four packets in its mini-slot.

Lemma 34. *In **Algorithm 2**, the maximum numbers of packets transmitted in one time slot is 4.*

Proof. It is sufficient to show that $\Delta x \leq 3$. Since the successive increments of $\tilde{x}$ are a geometric sequence, we have

$$\Delta x \leq \frac{\mathcal{K}}{a \cdot C_T}(1 + \frac{1}{C_T})^{C_T - 1}. \quad (3.20)$$

Notice that $(1 + 1/C_T)^{C_T - 1} \leq 3$ and $a \geq 1$. Then, $\Delta x \leq 3$ due to the assumption that $\mathcal{K} \leq C_T$. □

3.6 Two adversarial arrival processes

Thus far, we have examined the scenario where there is only one queue that keeps adversarial arrivals. In this subsection, we are considering a more general problem by allowing both arrivals to q_1 and q_2 controlled by adversary, where $\mathcal{K}$ packets will arrives at q_1, and so does q_2. Let $p_{i,j}$ be the j^{th} packet in q_i, i.e., there will be a set of packets $\{p_{1,1} \cdots, p_{1,\mathcal{K}}\}$ and $\{p_{2,1}, \cdots, p_{2,\mathcal{K}}\}$ added into q_1 and q_2 respectively by the adversary. We focus on the decision of the packets in q_1, while the packets in q_2 are always transmitted immediately. That is, in each time slot, we have to determine whether or not to transmit a packet in q_1. This assumption is inspired by an interesting and practical network scenario, where packets from n_2 is delay-sensitive (e.g., real time traffic), while packets from n_1 can tolerate delay; therefore, we can take advantage of the delay of packets in q_2 to increase the power efficiency by means of combining the transmission from both of the queues.

We start by introducing the notations. Let $x_{i,j}$ indicate if the packet $p_{i,j}$ from n_i is transmitted without coding. In particular, $x_{i,j} = 1$ if the corresponding packet is transmitted without coding; otherwise, $x_{i,j} = 0$. By z_t, we denote the number of packets waiting in q_1 at time t. Then the total cost at relay including transmission and waiting cost is $C_T \cdot \mathcal{K} + \sum_{i=1}^{\mathcal{K}} x_{1,i} + \sum_t z_t$, where the first term represents the total cost of transmitting packets in q_2, the second one means the additional cost for transmitting packets in q_1 that are transmitted without coding, and the last one indicates the delay. Due to the constant of $C_T \cdot \mathcal{K}$, the relay make decisions for packets in q_1 with the objective of minimizing $\sum_{i=1}^{\mathcal{K}} x_{1,i} + \sum_{t=0}^{\infty} z_t$.

In Subsection 3.6.1, we examine a formulation of the primal/dual program, how-

ever from which we find the difficulty to work out an on-line algorithm. Accordingly, Subsection 3.6.2 provides a genius interpretation of the primal/dual program, which makes Alg. 1 work in the setting of two adversarial arrivals.

3.6.1 Naive primal/dual program

We have known the objective function of the primal program, and we are going to propose the corresponding constraints. By $\eta_{i,t} = \sum_{\tau=0}^{t} A_t^{(i)}$, we define the number of packets that arrived at q_i by time t. Without loss of generality, we assume that $\eta_{2,t} \leq \eta_{1,t}$ since if $\eta_{2,t} > \eta_{1,t}$ for some t, then $\eta_{2,t} - \eta_{1,t}$ packets in q_1 cannot find the coding pair in q_2 and do not need any decision.

By time t, there are $\sum_{i=1}^{\eta_{2,t}} x_{2,i}$ packets in q_2 that are transmitted without coding, and hence $\eta_{2,t} - \sum_{i=1}^{\eta_{2,t}} x_{2,i}$ packets in q_1 are combined with the packets in q_2. For each time, there are three policies for the packets in q_1: (1) transmitted immediately without coding, (2) combined with the packet in q_2, or (3) do nothing. Putting together the packets corresponding the there decisions yells the constraint for each time t being $\sum_{i=1}^{\eta_{1,t}} x_{1,i} + (\eta_{2,t} - \sum_{i=1}^{\eta_{2,t}} x_{2,i}) + z_t = \eta_{1,t}$. Accordingly, the primal and dual program can be written as follows.

Primal program:

$$\min \quad C_T \sum_{i=1}^{\mathcal{K}} x_{1,i} + \sum_{t=0}^{\infty} z_t \tag{3.21}$$

$$\text{s.t.} \quad \sum_{i=1}^{\eta_{1,t}} x_{1,i} + z_t \geq \eta_{1,t} - \eta_{2,t} \text{ for all } t. \tag{3.22}$$

Dual program:

$$\begin{aligned} \max \quad & \sum_{t=0}^{\infty} (\eta_{1,t} - \eta_{2,t}) y_t \\ \text{s.t.} \quad & \sum_{t=0}^{\infty} y_t \leq C_T \\ & 0 \leq y_t \leq 1 \text{ for all } t. \end{aligned}$$

Based on the idea of Alg. 1, the fractional primal-dual algorithm corresponding to the above primal/dual program would update $x_{1,i}$ for at most C_T times, i.e., set $y_t \leftarrow 1$ for all t and $\sum_{t=0}^{\infty} y_t \leq C_T$. Consider that the $p_{1,1}$ arrives at q_1 at $t = 0$ and all other packets to q_1 come after $t = C_T - 1$, while all packets to q_2 arrive after $t = C_T$. Using Alg. 1, the variable $x_{1,1}$ is updated as $x_{1,1} \leftarrow x_{1,1}(1 + \frac{1}{C_T}) + \frac{1}{aC_T}$, and the variable y_t is updated to be 1 for time $0 \leq t \leq C_T - 1$. Then the constraint in the dual program is tight; as such, when $p_{1,2}$ arrives at q_1, the variable $x_{1,2}$ cannot be updated, i.e., packets $p_{1,j}$, for $j \geq 2$, are decided to be transmitted immediately after arriving. From this perspective, Alg. 1 cannot be directly applied for an on-line algorithm with the competitive ratio of $e/(e-1)$.

3.6.2 Alternative primal/dual formulation

We are facing the challenge because the primal constraint $\sum_{i=1}^{\eta_{1,t}} x_{1,i} + z_t = \eta_{1,t} - \eta_{2,t} + \sum_{i=1}^{\eta_{2,t}} x_{2,i}$ result in limited dual variables that can be updated. Consequently, we suggest another set of constraints that leads to the same optimal solution, as well as makes Alg. 1 applicable.

By $z_{i,t}$ we indicate if packet $p_{1,i}$ in q_1 waits at time t, where $z_{i,t} = 1$ if it waits at time t; otherwise, $z_{i,t} = 0$. For each time t, the value of $x_{1,i} + z_{i,t}$ would be either 0 or 1, i.e., $x_{1,i} + z_{i,t} = 0$ represents that the associated packet is transmitted with coding, while $x_{1,i} + z_{i,t} = 1$ means that the packet is either transmitted without coding or

wait at time t.

Given the value of $x_{2,i}$ for all i, we intend to separate the original constraint $\sum_{i=1}^{\eta_{1,t}}(x_{1,i}+z_{i,t}) = \eta_{1,t} - \eta_{2,t} + \sum_{i=1}^{\eta_{2,t}} x_{2,i}$ to $\eta_{1,t}$ constraints, i.e., to specify $x_{1,i}+z_{i,t}$ is 0 or 1 for all $1 \le i \le \eta_{1,t}$. At time t, there are $n_{1,t} - \eta_{2,t} + \sum_{i=1}^{\eta_{2,t}} x_{2,i}$ constraints such that $x_{1,i}+z_{i,t} = 1$ and $x_i + z_{i,t} = 0$ otherwise, where $1 \le i \le \eta_{1,t}$. Let $I_t = \{i : x_{1,i}+z_{i,t} = 1\}$ be the set of all index i such that $x_{1,i}+z_{i,t} = 1$ at time t. Intuitively, we are considering the constraints packet by packet; instead of the overall effect. At each time t, there are $\binom{\eta_{1,t}}{\eta_{1,t}-\eta_{2,t}+\sum_{i=1}^{\eta_{2,t}} x_{2,i}}$ possible choices of I_t. Let $\{I_t\}$ be the set of I_t for all t. Our objective is therefore to find the *correct* constraints, i.e. to determine $\{I_t\}$, such that the optimal solution in Eq. (3.21) is maintained.

Note that packet $p_{1,i}$ in q_1, with $i \in I_t$, is either transmitted without coding or wait at time t, while other packets in q_1 added before time t are transmitted with coding. Now we relate I_{t+1} with I_t as follows. First, we suppose no packet arrives at time $t+1$. Then I_{t+1} is equal to I_t, because no packet arrive at q_2 at time $t+1$ and packet $p_{1,i}$ in q_1, with $i \in I_t$, is either transmitted without coding or wait at time $t+1$. Second, if at time $t+1$ packet $p_{1,i}$ arrives at q_1, then the packet is either transmitted without coding or waits at time $t+1$ because there is no packet in q_2. Hence, in addition to the constraints by time t, one more constraint $x_{1,i}+z_{i,t+1} = 1$ should be considered, i.e., $I_{t+1} = I_t \cup \{i\}$. Finally, we examine the case that at time $t+1$ packet $p_{2,i}$ is added to q_2. If $x_{2,i} = 1$, then packet $p_{1,i}$, with $i \in I_t$, is still transmitted without coding or waits at time $t+1$; therefore $I_{t+1} = I_t$. If $x_{2,i} = 0$, then packet $p_{2,i}$ in q_2 is coded with $p_{1,j}$, for some $j \in I_t$, in q_1. Hence the corresponding constraint at time $t+1$ is $x_{1,j} + z_{j,t+1} = 0$ and $I_{t+1} = I_t \backslash \{j\}$.

However, we will observe in Ex. 35 that though every choice of $j \in I_t$ results in the same value of $\sum_{i=1}^{\eta_{1,t+1}}(x_{1,i}+z_{i,t})$ in Eq. (3.22), i.e., which is equal to $\eta_{1,t+1} - \eta_{2,t+1} + \sum_{i=1}^{\eta_{2,t+1}} x_{2,i}$, the different consideration of $j \in I_t$ gets the different optimal

value in Eq. (3.21). We associate $\{I_t\}$ with the constraints: for all i, $x_{1,i} + z_{i,t} = 1$ for $\alpha_i \leq t \leq \beta_i$, where α_i is the time when the packet $p_{1,i}$ arrives at q_1 and β_i is the largest t such that $i \in I_t$. Note that β_i can be infinite. Moreover, by γ_i, we denote the time when packet $p_{2,i}$ comes at q_2.

Example 35. We assume that $C_T = 5$ and $\mathcal{K} = 2$. Packets $p_{1,1}, p_{1,2}$ arrive q_1 at time 0 and 2 respectively, while packets $p_{2,1}, p_{2,2}$ are added to q_2 at time 4 and 5 respectively. We intend to minimize $C_T \cdot \sum_{i=1}^{2} x_{1,i} + \sum_{i=1}^{2} \sum_{t=0}^{\infty} z_{i,t}$ subject to the constraints $\sum_{i=1}^{\eta_{1,t}} (x_{1,i} + z_{i,t}) = \eta_{1,t} - \eta_{2,t} + \sum_{i=1}^{\eta_{2,t}} x_{2,i}$ for all t. We can calculate that the optimal solution is $x_{1,1} = 1$, $x_{1,2} = 0$, $x_{2,1} = 0$, $x_{2,2} = 1$, and the optimal value is 7.

Now, we figure out the value of $x_{1,i} + z_{i,t}$ for all i and time t, with the purpose of keeping the optimal value. From time $t = 0$ to $t = 1$, it is obvious that $x_{1,t} + z_{1,t} = 1$. At time $t = 2$, another packet is added to q_1, resulting in the constraint in Eq. (3.22) as $\sum_{i=1}^{2} (x_{1,i} + z_{i,t}) = 2$ for $t = 2, 3$. Therefore, at time $t = 2, 3$, we have $x_{1,1} + z_{1,t} = 1$ and $x_{1,2} + z_{2,t} = 1$. At time $t = 4$, packet $p_{2,1}$ arrives at q_2, and the constraint in Eq. (3.22) is $\sum_{i=1}^{2} (x_{1,i} + z_{i,4}) = 2 - 1 + x_{2,1}$. If $x_{2,1} = 1$, i.e., no packet in q_1 is combined with packet $p_{2,1}$, then we have $x_{1,1} + z_{1,4} = 1$ and $x_{1,2} + z_{2,4} = 1$. If $x_{2,1} = 1$, then one of packets in q_1 is coded and there will be two possible choices: (1) $x_{1,1} + z_{1,4} = 1$, $x_{1,2} + z_{2,4} = 0$, i.e., packet $p_{1,1}$ is determined to transmitted with coding, while packet $p_{1,2}$ is either transmitted without coding or wait at time $t = 4$; (2) $x_{1,1} + z_{1,4} = 0$, $x_{1,2} + z_{2,4} = 1$. At time $t = 5$, another packet is added to q_2 and we have the constraint $\sum_{i=1}^{2} (x_{1,i} + z_{i,5}) = x_{2,1} + x_{2,2}$. Similarly, we have a couple of choices to assign the value of $x_{1,i} + z_{i,t}$ for $t \geq 5$.

We assume that $x_{2,1} = 0$ and $x_{2,2} = 1$, and calculate the optimal value in Eq. (3.21) for each choice. Note that each case produces the same value of $\sum_{i=1}^{\eta_{1,t}} (x_{1,i} + z_{i,t})$

as $\eta_{1,t} - \eta_{2,t} + \sum_{i=1}^{\eta_{2,t}} x_{2,i}$ for all i and t.

Case (1) If the constraint is $x_{1,1} + z_{1,t} = 1$ for $0 \leq t \leq 3$, and $x_{1,2} + z_{2,t} = 1$ for $t \geq 2$: The optimal solution is $x_{1,1} = 0$ and $x_{1,2} = 1$, while the optimal value is 9.

Case (2) If the constraint is $x_{1,1} + z_{1,t} = 1$ for $t \geq 0$, and $x_{1,2} + z_{2,t} = 1$ for $2 \leq t \leq 3$: The optimal solution is $x_{1,1} = 1$ and $x_{1,2} = 0$, while the optimal value is 7.

We conclude that even though $x_{2,i}$ for all i is given, the optimal value for different choice to split the original constraint is different. ◀

Thus far, we understand that $\{I_t\}$ needs to be carefully considered to maintain the optimality. When the arrivals for each time are given, $\{I_t\}$ cannot be selected arbitrarily. Accordingly, we suggest the notion of the *feasible* $\{I_t\}$ as follows.

Definition 36. Given the arrivals $A_t^{(i)}$ for all i and t, we say $\{I_t\}$ is *feasible* if $\{I_t\}$ satisfies the following condition.

- When $A_t^{(1)} = A_t^{(2)} = 0$, $I_t = I_{t-1}$.
- When $A_t^{(1)} = 1$, say packet $p_{1,i}$ arrives at q_1, $I_t = I_t \cup \{i\}$.
- When $A_t^{(2)} = 1$, $I_t = I_{t-1}$ or $I_t = I_{t-1} \backslash \{j\}$ for some $j \in I_{t-1}$.

Definition 37. Given the arrivals $A_t^{(i)}$ for all i and t, we say a feasible $\{I_t\}$ is *optimal* if the optimal value in Eq. (3.21) is the same as the optimal value subject to the constraints associated with $\{I_t\}$.

Remark 38. Here, we give an intuition of the optimal $\{I_t\}$. Assume that $\mathcal{K} = 2$, and packets $p_{1,1}$ and $p_{1,2}$ are added prior to $p_{2,1}$, i.e., $\alpha_1 < \alpha_2 < \gamma_1$.

If $\gamma_1 - \alpha_2 > C_T$, then $p_{1,1}$ and $p_{1,2}$ would not wait for coding since the waiting cost cannot be compensated by the saving from the coding. Hence, $x_{1,1} + z_{1,\gamma_1} = 1$ and $x_{1,2} + z_{2,\gamma_1} = 1$.

If $\gamma_1 - \alpha_2 \leq C_T$, then one of $p_{1,1}$ and $p_{1,2}$ would be combined with $p_{2,1}$. Two cases are discussed as follows.

- Suppose that the optimal solution is $x_{1,1} = x_{2,1} = 0$, i.e. packets $p_{1,1}$ and $p_{1,2}$ would wait for coding. Both packets $p_{1,1}$ and $p_{1,2}$ are in q_1 at time $t = \gamma_1$, and therefore we can choose any of them to code with $p_{2,1}$. That is, we can choose $x_{1,1} + z_{1,\gamma_1} = 0$ or $x_{1,2} + z_{2,\gamma_1} = 0$.

- Suppose that the optimal solution is $x_{1,1} = 1$ or $x_{1,2} = 1$, i.e., one of packets $p_{1,1}, p_{1,2}$ would be transmitted without coding, while the other one would wait for coding. Then, it must be the case that $x_{1,1} = 1$ since packet $p_{1,1}$ wait longer than packet $p_{1,2}$ for the purpose of combining with $p_{2,1}$. Therefore, we have to choose $x_{1,2} + z_{1,\gamma} = 0$, but not $x_{1,2} + z_{1,\gamma_1} = 0$.

To conjecture for the general $\mathcal{K}$, when packet $p_{2,j}$ comes, if $\gamma_j - \alpha_i > C_T$ for all $i \in I_{\gamma_j - 1}$, then $I_{\gamma_j} = I_{\gamma_j - 1}$; otherwise, $I_{\gamma_j} = I_{\gamma_j - 1} \backslash \{k\}$ with $k = \arg\max_{i \in I_{\gamma_j - 1}} \alpha_i$, i.e., choose the packet in q_1 that is closest to time $t = \gamma_j$ to encode with packet $p_{2,j}$.

3.6.3 Compute the optimal solution when $\{I_t\}$ is given

Before presenting how to find the optimal $\{I_t\}$, we examine the optimal solution when I_t for all t is given. We consider the optimization problem as follows.

$$\begin{aligned} \min \quad & C_T \sum_{i=1}^{\mathcal{K}} x_{1,i} + \sum_{t=0}^{\infty} \sum_{i=1}^{\mathcal{K}} z_{i,t} \\ \text{s.t.} \quad & x_{1,i} + z_{i,t} \geq 1 \text{ for all } i \text{ and } \alpha_i \leq t \leq \beta_i \end{aligned}$$

Lemma 39. *The optimal value to the above problem is* $\sum_{i=1}^{\mathcal{K}} \min\{\beta_i - \alpha_i + 1, C_T\}$.

Proof. We compute the optimal solution via the associated dual program as follows.

$$\begin{aligned} \max \quad & \sum_{i=1}^{\mathcal{K}} \sum_{t=\alpha_i}^{\beta_i} y_{i,t} \\ \text{s.t.} \quad & \sum_{t=\alpha_i}^{\beta_i} y_{i,t} \leq C_T \\ & 0 \leq y_{i,t} \leq 1 \text{ for all } i \text{ and } t \end{aligned} \tag{3.23}$$

Since $0 \leq y_{i,t} \leq 1$, we have $\sum_{t=\alpha_i}^{\beta_i} y_{i,t} \leq \min\{\beta_i - \alpha_i + 1, C_T\}$, and the objective function is upper bounded as follows.

$$\begin{aligned} & \sum_{i=1}^{\mathcal{K}} \sum_{t=\alpha_i}^{\beta_i} y_{i,t} \\ \leq \quad & \sum_{i=1}^{\mathcal{K}} \min\{\beta_i - \alpha_i + 1, C_T\} \end{aligned}$$

The equality in the above equation holds when $y_{i,t} = 1$ for all $\alpha_i \leq t \leq \min\{\beta_i, \alpha_i + C_T - 1\}$. Therefore, according to the duality theory, we conclude that the optimal value is $\sum_{i=1}^{\mathcal{K}} \min\{\beta_i - \alpha_i + 1, C_T\}$. □

3.6.4 Optimal $\{I_t\}$

We understand in Subsection 3.6.2 that when a packet arrives at q_2 at time t, it is not obvious which $j \in I_{t-1}$ should be selected for $x_{1,j} + z_{j,t} = 0$. We are looking for the optimal $\{I_t^*\}$. Now we are ready to present an algorithm in Alg. 3 to compute the optimal $\{I_t^*\}$.

Theorem 40. *Alg. 3 produces the optimal $\{I_t^*\}$.*

Proof. First, it is obvious that $\{I_t^*\}$ produced from Alg. is feasible. It suffices to show that when a packet is added to q_2 at time t, Alg. 3 produces the optimal I_t^*. We prove it by induction over the number of packets $\mathcal{K}$.

Algorithm 3: Optimal $\{I_t\}$ computation

input : Arrivals $A_t^{(i)}$ for all i and t
output: $\{I_t\}$ and $x_{2,i}$ for all i

$\eta_{i,t} \leftarrow 0$ for all t;
$t \leftarrow 0$
while $\eta_{2,t} \leq \mathcal{K}$ **do**
  if $A_t^{(1)} = 1$ **then**
    $\eta_{1,t} \leftarrow \eta_{1,t-1} + 1$;
    $I_t \leftarrow I_{t-1} \cup \{\eta_{1,t}\}$;
  else
    $\eta_{1,t} \leftarrow \eta_{1,t-1}$;
    $I_t \leftarrow I_{t-1}$;
  end
  if $A_t^{(2)} = 1$ **then**
    $\eta_{2,t} \leftarrow \eta_{2,t-1} + 1$;
    $j^* \leftarrow \max\{j \in I_{t-1}\}$;
    if $t - \alpha_{j^*} \geq C_T$ **then**
      $x_{2,\eta_{2,t}} \leftarrow 1$;
    else
      $x_{2,\eta_{2,t}} \leftarrow 0$;
      $I_t \leftarrow I_t \backslash \{j^*\}$;
    end
  else
    $\eta_{2,t} \leftarrow \eta_{2,t-1}$;
  end
  $t \leftarrow t + 1$;
end

First, we consider $\mathcal{K} = 1$. If $\gamma_1 \leq C_T - 1$, then the optimal solution to Eq. (3.21) is $x_{1,1} = x_{2,1} = 0$, and $z_{1,t} = 1$ for $0 \leq t \leq \gamma_1 - 1$; hence, $x_{1,1} + z_{1,t} = 1$ for $0 \leq t \leq r_1 - 1$. If $\gamma_1 > C_T - 1$, then $x_{1,1} = x_{2,1} = 1$, and $z_{1,t} = 0$ for all t; hence $x_{1,1} + z_{1,t} = 1$ for all t. It is easy to verify that Alg. 3 produces the optimal $\{I_t^*\}$.

Second, we suppose that when $\mathcal{K} = k$, Alg. 3 produces the optimal $\{I_t^*\}$. Let $\mathbf{A} = \{(A_t^{(1)}, A_t^{(2)})\}$. By $\mathbf{A}\backslash\{(t_1, t_2)\}$ we denote the the arrival process that is equal to $\mathbf{A}$ except for $A_{t_1}^{(1)} = 0$ and $A_{t_2}^{(2)} = 0$, i.e., to remove the packet arrivals $A_{t_1}^{(1)}$ and $A_{t_2}^{(2)}$ if any. By $\mathrm{OPT}_k(\mathbf{A})$, we define the optimal value when the arrival process is $\mathbf{A} = \{(A_t^{(1)}, A_t^{(2)})\}$.

Now we are considering the case when $\mathcal{K} = k + 1$ associated with the arrival process $\mathbf{A}$. We express OPT_{k+1} in terms of OPT_k by means of Bellman equation and Lemma 39.

$$\mathrm{OPT}_{k+1}(\mathbf{A}) \;=\; \min_{i:\alpha_i<\gamma_1} \big\{\min\{\gamma_1 - \alpha_i, C_T\} + \mathrm{OPT}_k(\mathbf{A}\backslash\{\alpha_i, \gamma_1\})\big\}$$

Let $\zeta = \max\{i : \alpha_i < \gamma_1\}$. Note that Alg. 3 produces $\mathrm{OPT}_k(\mathbf{A}\backslash\{(\alpha_i, \gamma_1)\})$ for all $1 \leq i \leq \zeta$. Given $\mathbf{A}\backslash\{(\alpha_\zeta, \gamma_1)\}$, we assume that from Alg. 3 the constraints for packets $p_{1,i}$, with $1 \leq i \leq \zeta - 1$, are $x_{1,i} + z_{i,t} = 1$ for $\alpha_i \leq t \leq \beta_i^*$. Then we compute $\mathrm{OPT}_k(\mathbf{A}\backslash\{\alpha_\zeta, \gamma_1\})$ as follows.

$$\mathrm{OPT}_k(\mathbf{A}\backslash\{\alpha_\zeta, \gamma_1\}) = \sum_{i=1}^{\zeta-1} \min\{\beta_i^* - \alpha_i + 1, C_T\} + R,$$

where R include the the cost incurred due to the packets that arrive after γ_1. More-

over, for $1 \leq j < \zeta$, we have

$$\begin{aligned}
&\mathrm{OPT}_k(\mathbf{A}\backslash\{\alpha_j, \gamma_1\}) \\
=&\sum_{i=1}^{j-1} \min\{\beta_j^* - \alpha_i + 1, C_T\} + \sum_{i=j+1}^{\zeta} \min\{\beta_{i-1}^* - \alpha_i + 1, C_T\} + R
\end{aligned}$$

We note that, for $a \leq \gamma_1 \leq b$,

$$\min\{b - a, C_T\} = \min\{\gamma_1 - a, C_T\} + [\min\{a + C_T, b\} - \gamma_1]^+$$

Therefore, for $1 \leq j < \zeta$, we can computer

$$\begin{aligned}
&\min\{\gamma_1 - \alpha_j, C_T\} + \mathrm{OPT}_k(\mathbf{A}\backslash\{\alpha_j, \gamma_1\}) - \min\{\gamma_1 - \alpha_\zeta, C_T\} + \mathrm{OPT}_k(\mathbf{A}\backslash\{\alpha_\zeta, \gamma_1\}) \\
=&\sum_{i=j}^{\zeta-1} [\min\{\alpha_{i+1} + C_T, \beta_i^*\} - \gamma_1]^+ - \sum_{i=j}^{\zeta-1} [\min\{\alpha_i + C_T, \beta_i^*\} - \gamma_1]^+ \geq 0,
\end{aligned}$$

where the last inequality comes from $\alpha_{i+1} > \alpha_i$ for all i. Therefore, Alg. 3 results in the optimal $\{I_t\}$ when $\mathcal{K} = k + 1$. We conclude that Alg. 3 produces the optimal $\{I_t\}$ by induction. □

3.7 Fractional algorithm for the case of two adversaries

Given the arrivals $A_t^{(i)}$ for all i and t, the off-line optimization problem can be formulated as follows primal and dual program, where β_i^* is computed using Alg. 3.

Primal program:

$$\begin{aligned}
\min \quad & C_T \sum_{i=1}^{\mathcal{K}} x_{1,i} + \sum_{t=0}^{\infty} \sum_{i=1}^{\mathcal{K}} z_{i,t} \\
\text{s.t.} \quad & x_{1,i} + z_{i,t} \geq 1 \text{ for } \alpha_i \leq t \leq \beta_i^*
\end{aligned} \tag{3.24}$$

Deal program:

$$\begin{aligned} \max \quad & \sum_{i=1}^{\mathcal{K}} \sum_{t=\alpha_i}^{\beta_i^*} y_{i,t} \\ \text{s.t.} \quad & \sum_{t=\alpha_i}^{\beta_i^*} y_{i,t} \leq C_T \\ & 0 \leq y_{i,t} \leq 1 \text{ for all } i \text{ and } t \end{aligned} \tag{3.25}$$

Now, we are ready to present the on-line fractional algorithm for case of the two adversaries.

Using the similar proofs in Subsection 3.5.3, we conclude the results as follows.

Lemma 41. ***Algorithm 4*** *produces a feasible fractional solution to primal program.*

Lemma 42. ***Algorithm 4*** *produces a feasible fractional solution to the dual program when* $a = (1 + \frac{1}{C_T})^{C_T} - 1$.

Theorem 43. *Let OPT be the optimal off-line solution to primal program. Then, the cost of the solution computed by* ***Algorithm 4*** *is upper bounded by*

$$(1 + \frac{1}{(1 + 1/C_T)^{C_T} - 1})OPT.$$

Algorithm 4: Fractional Primal-Dual algorithm

z_t, y_t, $\eta_t \leftarrow 0$ for all t ;
x_i, $\tilde{x}_i \leftarrow 0$ for all $1 \leq i \leq \mathcal{K}$;
$\tilde{z}_t \leftarrow 0$ for all t;
$a \leftarrow (1 + \frac{1}{C_T})^{C_T} - 1$;
For each new time slot t, the parameters are updated as following.;
while $n_{2,t} < \mathcal{K}$ **do**
    if $A_t^{(1)} = 1$ **then**
        $\eta_{1,t} \leftarrow \eta_{1,t-1} + 1$;
        $I_t \leftarrow I_{t-1} \cup \{\eta_{1,t}\}$;
    else
        $\eta_{1,t} \leftarrow \eta_{1,t-1}$;
        $I_t \leftarrow I_{t-1}$;
    end
    if $A_t^{(2)} = 1$ **then**
        $\eta_{2,t} \leftarrow \eta_{2,t-1} + 1$;
        $j^* \leftarrow \max\{j \in I_{t-1}\}$;
        if $t - \alpha_{j^*} \geq C_T$ **then**
            $I_t \leftarrow I_{t-1}$;
        else
            $I_t \leftarrow I_{t-1} \backslash \{j^*\}$;
        end
    else
        $\eta_{2,t} \leftarrow \eta_{2,t-1}$;
        $I_t \leftarrow I_{t-1}$;
    end
    forall the $i \in I_t$ **do**
        if $x_i < 1$ **then**
            $z_{i,t} \leftarrow 1 - x_i$;
            $x_i \leftarrow x_i(1 + \frac{1}{C_T}) + \frac{1}{aC_T}$;
            $y_t \leftarrow 1$;
        end
    end
end

4. THE INDEX CODING PROBLEM: A GAME-THEORETICAL PERSPECTIVE*

4.1 Introduction

The Index Coding problem is one of the basic problems in wireless network coding [1, 2]. An instance of the Index Coding problem includes a server, a set of wireless clients, and a set $P = \{p_1, ..., p_m\}$ of m packets that need to be delivered to clients. Each client is interested in a certain subset of packets available at the server and has a (different) subset of packets available to it as side information. Server can transmit the packets or encoding thereof to clients via a noiseless broadcast channel. The goal is to find a transmission scheme that requires the minimum number of transmissions to satisfy the requests of all clients.

Fig. 4.1 depicts an instance of the Index Coding problem, where a server needs to deliver four packets $P = \{p_1, \cdots, p_4\}$ to four clients. Each client requires a unique packet in P and has side information as shown in the picture. It can be verified that the demands of all clients can be satisfied by broadcasting three packets: $p_1 + p_2$, $p_2 + p_3$, and p_4 (all operations are over $GF(2)$). Note that the traditional approach (without coding) requires the transmissions of all four packets $p_1, \cdots, p_4$.

Recently, the Index Coding problem has attracted a significant interest from the research community. The prior works on the Index Coding problem have focused on developing algorithms, establishing rate bounds, and analyzing the computational complexity of the problem [7, 34–38].

This section focuses on the game-theoretic aspects of the Index Coding problem.

* Part of the data reported in this section is reprinted with permission from "The Index Coding Problem: A Game-Theoretical Perspective" by Yu-Pin Hsu, I-Hong Hou, and Alex Sprintson, 2013. In *IEEE International Symposium on Information Theory Proceedings (ISIT)*, 977-981, Copyright 2013 by IEEE.

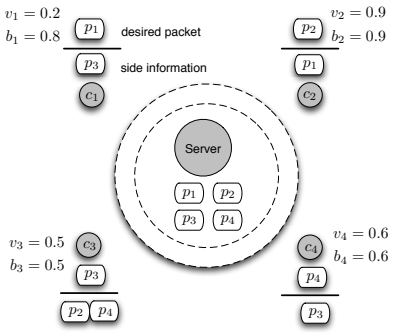

Figure 4.1: An instance of the Index Coding problem. Client $c_i, i = 1, \ldots, 4$ requests packet p_i. The side information sets of clients $c_1, \ldots, c_4$ are $\{p_3\}, \{p_1\}, \{p_2, p_4\}$, and $\{p_3\}$, respectively; v_i and b_i denote the value and bid of packet c_i, respectively.

In particular, we consider a setting in which each client is required to pay for the packets it obtains from server. The payment process is implemented through an auction. Each client submits a bid to the server for each packet it requests. The server decides whether to accept or reject each bid and determines the payment for all accepted bids. Clients are considered to be selfish, i.e., they consider the bids with the objective to maximize their utility while minimizing the amount of payment to the server. The utility of the clients is the difference between the true value of the packet to the client and the payment submitted to the server for this packet. Accordingly, our objective is to design a *truthful* mechanism, i.e., a mechanism that incentivizes every client to bid the true value of the packet.

In contrast to the original Index Coding problem where the goal of the server is to satisfy the demands of all clients, we focus on settings where the server's decisions are driven by the values of the transmitted packets to the clients. Intuitively, since each transmission at server incurs a certain cost, the server may decide to transmit a packet only when the packet is important enough for the clients. Thus, our goal is to maximize the *social welfare*, i.e., the total value of the packets the clients are able

to decode minus the total cost of transmitting the packets by the server. The social welfare reflects the positive externalities in terms of clients' valuations of packets and the negative externalities related by the transmission costs of server.

The auction mechanism includes two main components, namely the *coding function* and *payment function*. The coding function determines which packets are transmitted over the channel and how they are encoded. The payment function determines the amount of money the client will pay to the server for each packet it is able to decode.

4.1.1 Related work

The research on the Index Coding problem can be classified into two main directions. The first direction is focused on achievable rate bounds, as well as on the connections between the Index Coding and the Network Coding problems [34–36]. The second direction has on analyzing the computational complexity of the Index Coding problem as well as developing heuristic approaches to this problem [7,37,38]. In particular, the Index Coding problem has been shown to be $\mathcal{NP}$-hard in [38]. Chaudhry et al. [39] studied the related problem, referred to as the *Complementary Index Coding problem*. In this problem, the objective is to maximize the number of "saved" transmissions, i.e., $n - \mu$, where n is the number of packets that need to be delivered to the clients. The follow-up paper [40] focuses on finding sparse solutions to the problem, i.e., solutions in which each transmitted packet is a combination of at most two original packets.

4.1.2 Main results

First we design a truthful scheme, based on the *Vickrey-Clarke-Groves* (VCG) mechanism [41–43], that maximizes the social welfare of the system. It turns out that finding VCG based encoding functions for this scheme is an interesting optimization

problem. We refer to this problem as the *Price Collecting Index Coding* (PCIC) problem and show that this problem is $\mathcal{NP}$-hard. To mitigate the intractability of the PCIC problem, we present an approximation algorithm with provable performance guarantees. Unfortunately, as we show through a counter-example, a VCG based algorithm that relies on the approximation solution to the PCIC problem is no longer truthful. Accordingly, we present a non-VCG based payment scheme that achieves the truthfulness property in the presence of the approximation solution for the problem.

4.2 Model

We begin with the definition of the Index Coding problem. The input to the problem includes a server S, a set of n wireless clients $\Lambda = \{c_1, \cdots, c_n\}$, and a noiseless wireless broadcast channel. The server has a set of m packets $P = \{p_1, \cdots, p_m\}$ that need to be distributed to clients in Λ. We assume, without loss of generality, that each client requests a single packet in P and has access to a subset of packets in P as a side information. Indeed, a client that wants more than one packet can be substituted by multiple clients that share the same side information set. In particular, each client $c_i \in \Lambda$ is characterized by the ordered pair (w_i, H_i), where $w_i \in P$ is the packet requested by c_i and $H_i \subseteq P$ is the set of packets available to c_i as a side information. We assume that each packet p_i represents an element of the Galois field $GF(q)$.

We assume that each client $c_i \in \Lambda$ has the internal (private) value v_i for the packet w_i it requests. The transmission process includes an auction, in which each client submits a bid b_i for packet w_i. We denote by $V = \{v_1, \cdots, v_n\}$ and $B = \{b_1, \cdots, b_n\}$ the arrays that include the internal valuations and bids of the clients. Based on the bid vectors, the server identifies linear combinations that will be transmitted over the

channel. In this section, we focus on the *scalar-linear coding* schemes in which each packet $q_i = \sum_{j=1}^{n} g_{i,j} p_j$ transmitted by server at the iteration i, $1 \leq i \leq \eta$, is a linear combination of packets in P. Here, η is the total number of transmissions made by the server and $g_i = \{g_{i,1}, \cdots, g_{i,n}\} \in GF(q)^n$ is the *encoding vector* for the iteration i. We denote by $G = [g_i]$ the *encoding matrix*, rows of G correspond to encoding vectors of the packets transmitted over the channel. A *sparse code* is the general linear code in which each transmitted packet from server is a linear combination of *at most two* packets in P. The transmission of a packet by server incurs the cost C_T. Without lost of generality, we assume that $C_T = 1$, i.e., the total cost incurred by the server is equal to η.

To decode the packet w_i, client c_i uses a linear decoding function r_i, such that $w_i = r_i(q_1, \cdots, q_\eta, H_i)$. In our scheme, we do not require the server to satisfy the requests of all clients. Therefore, for some clients the function r_i might not exist. We say that a client can *immediately decode* a packet if there exists packet q_j and function f_i such that $w_i = q_j + f_i(H_i)$, where $f_j(H_i)$ is a linear combination of the packets in H_i. Note that with the immediate decoding function the client can only use one received packet. For example, in Fig. 4.1, client c_2 can immediately decode packet p_2 by using packet $p_1 + p_2$ transmitted over the channel. In contrast, client c_1 cannot decode packet p_1 by using packet $p_1 + p_2$ or packet $p_2 + p_3$ separately (but it can decode p_1 using a combination of packets $p_1 + p_2$ and $p_2 + p_3$). The immediate decoding scheme has a significant advantage in practical settings since a client only needs to receive one packet and can use a simple decoding algorithm to obtain the packet it needs.

The purpose of the server is to maximize the *social welfare*, defined as $\sum_{i=1}^{n} v_i \cdot \theta_i - \eta$, where θ_i is the indicator function that specifies if client c_i is able to decode the required packet w_i. In particular, $\theta_i = 1$ if there exists a linear decoding function r_i

such that $w_i = r_i(q_1, \cdots, q_\eta, H_i)$; otherwise, $\theta_i = 0$. We denote by $\Theta = \{\theta_1, \cdots, \theta_n\}$ the vector of indicator functions.

For example, in the setting depicted in Fig. 4.1, the social welfare of broadcasting the solution to the Index Coding problem, i.e. $\{p_1 + p_2; p_2 + p_3; p_4\}$, is $0.2 + 0.9 + 0.5 + 0.6 - 3 = -0.8$, while the higher social welfare of the sequence consisting of a single combination $\{p_3 + p_4\}$ is equal to $0.5 + 0.6 - 1 = 0.1$. Thus, transmitting a single combination $p_3 + p_4$ would be more desirable than transmitting three packets $\{p_1 + p_2; p_2 + p_3; p_4\}$ from the server's point of view.

The payment function is an important part of the auction mechanism. Server determines the amount of payment ϕ_i that each client needs to pay for the obtained packet. We denote by $\Phi = \{\phi_1, \cdots, \phi_n\}$ the payment vector for all clients. The value of ϕ_i is a function of B and Θ. The auction mechanism, including the algorithm for determining the encoding function and the payment function is known to all the parties.

We assume that every client c_i is selfish and chooses its best bidding policy b_i that maximizes its utility defined by $u_i(B) = v_i \cdot \theta_i - \phi_i$. We say that a mechanism (i.e., an encoding matrix associated with a payment function) is *truthful* if $u_i\big(v_i, B_{-i}\big) \geq u_i\big(\hat{b}_i, B_{-i}\big)$ for all $\hat{b}_i$ and B_{-i}, where $B_{-i} = B \setminus \{b_i\}$. That is, regardless of the bids submitted by other clients, the utility of client c_i is maximized when $b_i = v_i$.

In this section, we design the encoding matrix and the payment function such that satisfies the following two conditions: (i) every client is incentivized to report its true valuation of the desired packet (i.e., truthful property) and (ii) the social welfare is maximized. We will consider two scenarios, *multiple unicast* and *multiple multicast*. In multiple unicast environment, every client requires different packet (i.e., $w_i \neq w_j$ for $i \neq j$, and so $m = n$), while in the multiple multicast scenario, one packet could be requested by multiple packets (i.e., $m \leq n$).

4.3 VCG-based mechanism design

Our problem belongs to the general framework of *mechanism design* with *social choices* [44]. We leverage the celebrated Vickrey-Clarke-Groves (VCG) mechanism [41–43] to design a truthful mechanism. In particular, we propose a scheme, that include coding and payment functions, which is a variation of the VCG mechanism.

Note that the indicator functions Θ and the number of transmissions η are functions of the encoding matrix G. In particular, it is easy to determine Θ and η for a given matrix G. We define the *social welfare function* $w(B, G)$ as $w(B, G) = \sum_{i=1}^{n} b_i \cdot \theta_i - \eta$, where B replaces V in the definition of the social welfare. In addition, let $w_{-i}(B, G) = \sum_{j \neq i} b_j \cdot \theta_j - \eta$.

In our mechanism, we choose the encoding matrix that maximizes the the value of social welfare function for the given bids B.

VCG-coding scheme: The encoding matrix is chosen such that the function $w(B, G)$ is maximized, i.e. the optimal encoding matrix G^* is given by

$$G^* = \arg\max_{G} w(B, G). \tag{4.1}$$

If there are multiple encoding functions that maximize the social welfare functions, we choose one that satisfies the larger number of clients. Thus, without loss of generality, we can assume that $v_i \in [0, 1]$, since client c_i is assured to get the desired packet when submitting the bid $b_i = C_T = 1$.

The payment function ϕ_i is determined as follows.

VCG-payment function: The client c_i is charged as follows.

$$\phi_i = \left(\max_{G_{-i}} w(B_{-i}, G_{-i}) \right) - w_{-i}(B, G^*). \tag{4.2}$$

The first term in Equation (4.2) implies the optimal social welfare when client c_i is removed, while the second term represents the social welfare for all clients excluding c_i when the optimal encoding matrix G^* determined by (4.1) is employed. The VCG-payment function charges the "externality" of client c_i, i.e., the decrease in optimal social welfare when client c_i is included in the system.

Proposition 44. *The VCG-coding scheme associated with the VCG-payment function is a truthful mechanism, i.e. $B = V$. Moreover, the pricing function in (4.2) is non-negative, and the utilities of all clients are non-negative, $\phi_i \leq v_i$.*

Proof. Follows directly from the properties of the VCG mechanism [44]. □

Proposition 44 implies that each client c_i submits a bid $b_i = v_i$, i.e. $B = V$. Then, by (4.1) the server will choose encoding matrix G^* that maximizes the social welfare. We conclude in the following.

Proposition 45. *The VCG-coding scheme associated with the VCG-payment function maximize the social welfare.*

4.4 Hardness results

The problem of finding an optimal coding scheme in (4.1) is an interesting combinatorial problem by itself. We refer to this problem as the *Prize Collecting Index Coding* (PCIC) Problem. In the next lemma we show that this problem is $\mathcal{NP}$-hard for general linear encoding/decoding functions.

Proposition 46. *The PCIC problem is $\mathcal{NP}$-hard when considering general linear encoding/decoding scheme.*

Proof. We show reduction from the Index Coding problem which was shown to be $\mathcal{NP}$-hard. We start with an instance of the Index Coding problem and construct

Table 4.1: Algorithmic hardness of PCIC problem

	Immediate decoding	**General linear decoding**
Multiple unicast	Poly-time solvable	Truthful approximation algorithm: Greedy-VCG-coding scheme Greedy-VCG-payment function
Multiple multicast	$\mathcal{NP}$-hard and hard to approximate within the factor $n^{1-\epsilon}$	

an instance of the PCIC problem in which each client has a unit bid, i.e., $b_i = 1$ for each client. Since $b_i = C_T$, all clients will get the desired packet, hence $\theta_i = 1$ for all clients. This implies that $w(B, G) = n - \eta$. As the first term of $w(B, G)$ is constant, the problem of finding the code that maximizes $w(B, G)$ is equivalent to the problem of finding the encoding matrix G^* that minimizes the number of transmissions η. Thus, an optimal algorithm for the PCIC problem can be used for finding an optimal solution to the Index Coding problem. This implies that the PCIC problem is $\mathcal{NP}$-hard □

We understand that, in general, the PCIC problem is $\mathcal{NP}$-hard. In the following, we focus on the *sparse code*, where each packet is a combination of at most two packets in P. The reason to study the sparse solution is that, with sparse coding, both encoders and decoders can be implemented in a simpler manner which has a significant advantage for practical applications. Our results are summarized in Table 4.1.

The multiple unicast scenario is considered in Subsections 4.5 and 4.6. We show that there is a polynomial time algorithm for finding the optimal VCG-coding scheme with immediate decoding. For the general decoding function, we propose an approximation algorithm to the PCIC problem. Unfortunately, the VCG-payment function

in (4.2) can only be used with an optimal solution G^* from (4.1). An approximate solution to (4.1) is not sufficient to guarantee the truthfulness property. Accordingly, in Subsection 4.6, we propose a new pricing function that maintains the truthfulness property for the case in which the approximation solution is used.

In Subsections 4.7 and 4.8, we focus on the multiple multicast scenario. We show that in this scenario the PCIC problem is hard to approximate within a factor $n^{1-\epsilon}$ for any constant $\epsilon > 0$ even when the decoding is restricted to the immediate decoding.

4.5 Multiple unicast with immediate decoding

In this subsection we focus on the multiple unicast case with sparse coding and immediate decoding. We show that in this case the PCIC problem can be solved in polynomial time. We start by introducing the notion of the *weight dependency graph*.

Definition 47 (Weight dependency graph $G(V, A)$)**.** Given a multiple unicast instance of the PCIC problem, we define a directed graph $G(V, A)$ as follows:

- For each client $c_i \in C$, there is a corresponding vertex v_i in V.
- For any two clients c_i and c_j such that $w_i \in H_j$, there is a directed arc $(v_i, v_j) \in A$.
- The weight of arc (v_i, v_j) is $\gamma_{i,j} = b_i$.

Fig. 4.2 illustrates the weight dependency graph corresponding to the instance of the PCIC problem depicted in Fig. 4.1. For each vertex-disjoint cycle in the weight dependency graph, the server can save one transmission. The clients belonging to cycle C share the transmission cost $|C| - 1$, resulting in the higher social welfare. Indeed, in Fig. 4.2 we have a cycle (c_1, c_2, c_3) that involves three clients. It is easy

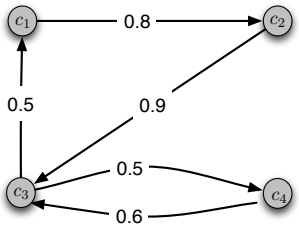

Figure 4.2: Weight dependency graph for the instance of the PCIC problem in Fig. 4.1

to verify that the three clients can be satisfied by two transmissions: $p_1 + p_2$ and $p_2 + p_3$. Note that if there is no cycle in $G(V, A)$ then sparse code can not improve the value of social welfare function. Given a cycle $C = (c_1, c_2, \cdots, c_k)$, server *encodes along cycle* C means that server would transmit the sequence of $k-1$ packets $\{p_1 + p_2; \cdots; p_{k-1} + p_k\}$.

Theorem 48. *In multiple unicast scenario with the restriction of sparse code and the immediate decoding, the PCIC problem can be solved in polynomial time.*

Proof. We prove by presenting a polynomial-time algorithm for this problem. Note that with immediate decoding we cannot use a cycle greater than two since a larger cycle will not result in decreasing the number of transmissions. For example, in Fig. 4.2 we cannot use the cycle (c_1, c_2, c_3) to generate an encoded packets $p_1 + p_2$ and $p_2 + p_3$ since in this case client c_1 will not be able to decode packet p_1. Thus, any optimal solution to the problem in this scenario corresponds to set of vertex-disjoint cycles of length two.

Our algorithm can be described as follows. First, given an instance of the PCIC problem, we construct the corresponding weight dependency graph $G(V, A)$. Then, we construct an undirected auxiliary graph $\tilde{G}(\tilde{V}, \tilde{E})$ through the following procedure.

- The vertex set $\tilde{V}$ of $\tilde{G}$ is the same as the vertex set V of G.
- For any two vertices $v_i, v_j \in V$ such that both arcs (v_i, v_j) and (v_j, v_i) exist, we create an edge $(\tilde{v}_i, \tilde{v}_j) \in \tilde{E}$.
- The weight of edge $(\tilde{v}_i, \tilde{v}_j) \in \tilde{E}$ is set to $\tilde{\gamma}_{i,j} = \gamma_{i,j} + \gamma_{j,i} - 1$.

Then, the algorithm performs the following steps:

1. Find the maximum weight matching M^* in $\tilde{G}(\tilde{V}, \tilde{E})$.
2. For every edge $(\tilde{v}_i, \tilde{v}_j) \in M^*$, transmit the packet $w_i + w_j$.
3. For the vertex v_i such that v_i is not matched and $b_i = 1$ transmit w_i.

The maximum weight matching can be solved in polynomial time (e.g., through Edmonds's algorithm). As discussed above, the only way to improve the social welfare is to use the vertex-disjoint cycles of length two in $G(V, A)$. Note that each such cycle corresponds to an edge in $\tilde{G}(\tilde{V}, \tilde{E})$. For each vertex-disjoint cycle in $G(V, A)$, we can save one transmission. Therefore, the weight $\tilde{\gamma}_{\tilde{e}}$ for $\tilde{e} = (i, j) \in \tilde{E}$ is defined to be $\gamma_{i,j} + \gamma_{j,i} - 1$, where the first two terms represent the contribution of the social welfare, while the last one is the shared transmission cost. Therefore, the optimal code is associated with the maximum weight matching in $\tilde{G}(\tilde{V}, \tilde{E})$. The third step in the algorithm is to make sure that the clients that submit the bid equal to the transmission cost will get the desired packets. □

4.6 Multiple unicast with general decoding

In this subsection, we assume that the clients can use a general linear decoding functions (i.e., beyond immediate decoding). We first investigate the hardness of the VCG-coding scheme in this setting.

Lemma 49. *In multiple unicast scenario with the restriction to the sparse code, the PCIC problem is $\mathcal{NP}$-hard.*

Proof. We show a reduction from the cycle packing problem in directed graphs. The objective of this problem is to find the maximum number of vertex-disjoin cycles in a directed graph $G(V, A)$. Let $G(V, A)$ be an instance of the cycle packing problem. We construct the weight dependency graph $G(V, A)$ by associating the weight $\gamma_a = 1$ to each arc $a \in A$. We then create an instance of the PCIC problem that corresponds to $G(V, A)$. In this instance, the bid of each client is set to 1. Since $b_i = C_T$ for all clients, each client is guaranteed to decode the wanted packet. Thus, $w(B, G) = n - \eta$, where the first term is constant. Thus, in this case, an optimal solution to the PCIC problem will have the minimum possible number of transmissions. With sparse coding, the number of transmissions can be reduced only when the server encodes along vertex-disjoin cycles in the weight dependency graph. For each vertex-disjoint cycle in $G(V, A)$, one transmission can be saved. Thus, finding the minimum number of transmissions is equivalent to find the maximum number of vertex-disjoint cycles in $G(V, A)$, which is $\mathcal{NP}$-hard. Thus, we conclude that finding a sparse code in the multiple unicast scenario is $\mathcal{NP}$-hard. □

We propose the greedy-VCG-coding scheme in Alg. 5 as an approximation algorithm for the PCIC problem in this setting. Next, we present a corresponding pricing scheme that motivates all clients to reveal the true value of the packets.

We define the weight of cycle C in $G(V, A)$ as $\gamma(C) = \sum_{a \in C} \gamma_a - (|C| - 1)$, where the first term is the sum of all bids along the cycle, and the second one is the total transmission cost of all packets that correspond to this cycle. Let $\mathbf{C}$ be a set of vertex-disjoint cycles in $G(V, A)$. Then $w(B, G) = \sum_{C \in \mathbf{C}} \gamma(C)$, where server encodes along the cycles in $\mathbf{C}$ as the encoding matrix G.

Algorithm 5: Greedy-VCG-coding scheme

input : Bids vector B and side information H_i for all clients
output: Encoding matrix G

1 Create the weight dependency graph $G(V, A)$;
2 Define the cost λ_a of arc $a \in A$ by $\lambda_a = 1 - \gamma_a$;
3 By $\lambda(C) = \sum_{a \in C} \lambda_a$, we define the cost of cycle C;
4 $\mathbf{S}_1, \mathbf{S}_2 \leftarrow \emptyset$; // $\mathbf{S}_1, \mathbf{S}_2$ will include the clients that are served
5 **while** *in $G(V, A)$, there is a cycle with the cost less than or equal to one* **do**
6 Find the cycle C in $G(V, A)$ with the smallest cost;
7 Server encodes along cycle C;
8 $\mathbf{S}_1 \leftarrow \mathbf{S}_1 \cup \{c_v : v \in V, v \in C\}$;
9 $G(V, A) \leftarrow G(V, A) \backslash \{$vertices along C, edges along or incidnet to $C\}$;
10 **end**
11 **for** $i \leftarrow 1$ **to** n **do**
12 **if** $b_i = 1$ *but* $c_i \notin \mathbf{S}_1$ **then**
13 Server will transmit w_i;
14 $\mathbf{S}_2 \leftarrow \mathbf{S}_2 \cup \{c_i\}$;
15 **end**
16 **end**

The idea of the greedy-VCG-coding scheme is iteratively to identify the cycle of the maximum weight and then to remove it. However, this scheme cannot be implemented directly since it is $\mathcal{NP}$-hard to find a maximum weight cycle. We note that $\gamma(C)$ can be revised as $\gamma(C) = 1 - \sum_{a \in C}(1 - \gamma_a)$. Thus, in this case, a maximum weight cycle with respect to weight $\gamma(C)$ corresponds to the minimum cost cycle with respect to cost $\sum_{a \in C}(1 - \gamma_a)$. Therefore, in lines 2 and 3 of Alg. 5, we define the cost of arc $a \in A$ by $\lambda_a = 1 - \gamma_a$ and the cost of cycle C by $\lambda(C) = \sum_{a \in C} \lambda_a$. Then, the algorithm finds the minimum cost cycle in Line 6. To this end, we can use well-known polynomial time algorithms such as Floyd-Warshall algorithm. After such a cycle is identified, it is removed from the graph in Line 9. The condition in Line 5 guarantees that the maximum weight cycle in the remaining graph $G(V, A)$

(i.e., the minimum cost cycle) has a non-negative weight. In Line 13, the clients with the bids equal to the transmission cost are served. Such clients are then included in set $\mathbf{S}_2$. In addition, set $\mathbf{S}_1$ contains the clients associated with the cycles chosen in Line 6. Note that all procedures in Alg. 5 require a polynomial number of steps, hence Alg. 5 can be executed in polynomial time.

Unfortunately, Alg. 5 cannot be used with the VCG-payment functions specified by (4.2) (i.e to replace G^* with the approximate solution). In particular, the following example shows that such combination might not have the truthfulness property.

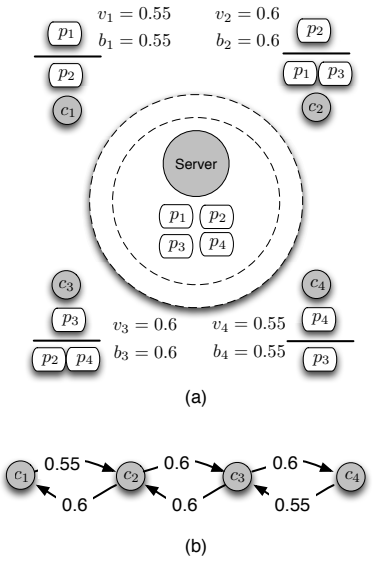

Figure 4.3: (a) Counter-example of the truthful property for the greedy-VCG-coding scheme accompanied with the VCG-payment function (b) The weight dependency graph

Example 50. Consider the instance of the problem depicted in Fig. 4.3. Given B_{-1} fixed, we will show that client c_1 has the potential to lie. Suppose that client c_1

submits the bid $b_1 = 0.55$ equal to its internal value of packet p_1. In this case, Alg. 5 will identify the cycle (c_2, c_3) and output the corresponding solution $p_2 + p_3$. In this case, the utility u_1 of c_i is equal to zero.

Now, suppose that c_1 submits bid $b_1 = 0.7$. In this case, Alg. 5 will transmit the sequence of packets $\{p_1 + p_2; p_3 + p_4\}$ that corresponds to cycles (c_1, c_2) and (c_3, c_4), respectively. According to (4.2), the payment of client c_1 is equal to $\phi_1 = (0.6 + 0.6 - 1) - (0.6 + 0.6 + 0.55 - 2) = 0.45$. In this case, the utility of client c_1 is equal to one, which is bigger than the case in which the client is truthful. ◀

Accordingly, we present the greedy-VCG-payment scheme implemented by Alg. 6 to charge the clients c_i. This scheme results in a truthful auction.

It can be easily verified that Alg. 6 has polynomial complexity. Note that in Line 4 of Alg. 6, the cost function of $G(V, A)$ is different from that in Alg. 5. To compute the payment for client $c_i \in \mathbf{S}_1$, we assign the unit cost for every outgoing arc (i, j) of vertex i (i.e. set $\gamma_{i,j} = 0$ and $b_i = 0$). For each iteration of Line 8, we calculate the difference of the cost between the cycles C_1 and C_2 in $G(V, A)$, where C_1 has the minimum global cost while C_2 is the local optimal cycle that traverses vertex i. We note that when the cycle C_{last} containing vertex i is selected in the last iteration of Alg. 5, the "while" loop in Alg. 6 might not capture C_{last} because the cost of arc (i, j) is set to be one. Line 20 deals with this case. When client c_i submits $b_i = \lambda^\dagger - 1$, the weight of cycle C_{last} in $G(V, A)$ is $(1 - \lambda^\dagger) + (\lambda^\dagger - 1) = 0$, where the first term is the weight when $b_i = 0$ while the second term is the increment when $b_i = \lambda^\dagger - 1$. Therefore, the cost of C_{last} is one, which satisfies the condition of Line 5 of Alg. 5.

The set $\mathcal{P}$ includes the result of each iteration. If the difference is zero, client c_i will not be charged; otherwise, it will be charged the value equal to the minimum of

Algorithm 6: Greedy-VCG-payment function

input : Bids vector B and side information H_i for all clients
output: Payment ϕ_i for client c_i

Clients that are not in $\mathbf{S}_1$ and $\mathbf{S}_2$ do not be charged, **return** $\phi_i = 0$;
The payment for the clients belonging to the $\mathbf{S}_2$ is 1, **return** $\phi_i = 1$;
// Following deals with the clients in $\mathbf{S}_1$
Create the weight dependency graph $G(V, A)$;
The cost is defined as follows;

- for arc $(i, j) \in A$, $\lambda_{(i,j)} = 1$
- for arc $a \neq (i, j)$, which is not outgoing from vertex i, $\lambda_a = 1 - \gamma_a$

By $\lambda(C) = \sum_{a \in C} \lambda_a$, we define the cost of cycle C;
$\mathcal{P} \leftarrow \emptyset$;
// set $\mathcal{P}$ will include the possible payments
while *in $G(V, A)$, there is a cycle of the cost less than or equal to one* **do**
    Find the cycle C_1 in $G(V, A)$ with the smallest cost, say the cost λ^* ;
    Find the cycle C_2 in $G(V, A)$ that traverses vertex i and has the smallest cost among these that go through vertex i, say the cost $\lambda^\dagger$;
    if $\lambda^\dagger - \lambda^* = 0$ **then**
        return $\phi_i = 0$
    end
    else
        $\mathcal{P} \leftarrow \mathcal{P} \cup (\lambda^\dagger - \lambda^*)$;
        $G(V, A) \leftarrow G(V, A) \backslash \{$vertices along C_1, edges along or incident to $C_1\}$;
    end
end
Find the cycle C_2 in $G(V, A)$ (if any) that traverses vertex i and has the smallest cost among these that go through vertex i, say the cost $\lambda^\dagger$;
$\mathcal{P} \leftarrow \mathcal{P} \cup (\lambda^\dagger - 1)$;
return $\phi_i = \min \mathcal{P}$

an element in set $\mathcal{P}$.

Given the bids of all clients, the coding algorithm determines the encoding matrix G, which, in turn, determines the indicator vector Θ. We say that a coding algorithm is *monotone* if, for any B_{-i}, there exists a single *critical value* $\overline{b}_i$ such that c_i gets the desired packet (i.e. $\theta_i = 1$) when $b_i \geq \overline{b}_i$; otherwise, c_i can not.

Theorem 51. *The mechanism is truthful if and only if the coding algorithm is monotone and the payment scheme is based on the critical value.*

Proof. Follows immediately from Theorem 1 in [45]. □

Theorem 52. *The Alg.'s 5 and 6 result in a truthful mechanism.*

Proof. According to Theorem 51, it is sufficient to show that Alg. 5 is monotone, as well as Alg. 6 is based on the critical value.

Suppose client c_i is able to recover the desired packet. Then, a cycle containing vertex v_i will be chosen by Alg. 5. Alg. 5 prefers the cycle of the maximum weight (i.e., minimum cost) in weight dependency graph $G(V, A)$. When c_i increases the bid, so does the weight of the cycles that pass through vertex v_i, and therefore a cycle containing vertex v_i will be selected by Alg. 5. Hence, the greedy-VCG-coding scheme is monotone.

Given B_{-i}, for each iteration in Alg. 6, we calculate the lowest bid b_i for client c_i such that a cycle containing vertex v_i would be selected by Alg. 5. Since the minimum of these values are adopted as the payment for client $c_i \in \mathbf{S}_1$, client $c_i \in \mathbf{S}_1$ is charged by the critical value. Other cases, i.e. client $c_i \notin \mathbf{S}_1$, follow immediately. □

Let OPT be the optimal solution of the VCG-coding scheme, i.e. $OPT = w(B, G^*)$. Assume G_{greedy} is the encoding matrix from the greedy-VCG-coding scheme. We define the *approximation ratio* of Alg. 5 by OPT/APX, where $APX = w(B, G_{\text{greedy}})$.

Theorem 53. *The approximation ratio of Alg. 5 scheme is equal to the maximum length of the cycle in the weight dependency graph.*

Proof. The optimal solution G^* in (4.1) corresponds to the maximum weight vertex-disjoint cycle packing, say $\mathbf{C}^*$, in the weight dependency graph $G(V, A)$. The total weight of these cycles in $\mathbf{C}^*$ is OPT, i.e. $w(B, G^*) = \sum_{C \in \mathbf{C}^*} \gamma(C)$

Given a cycle C in the weight dependency graph, let $C \cap \mathbf{C}^*$ be the set of the cycles in $\mathbf{C}^*$ that have a shared vertex with C. For each iteration k of Alg. 5, we encode along the cycle C_k of the minimum cost in $G_k(V, A)$, where $G_k(V, A)$ is the remaining graph in iteration k.

Let $\mathbf{C}_k^*$ be the set of cycles in $\mathbf{C}^*$ that belongs to $G_k(V, A)$. We define $APX_k = \gamma(C_k)$ and $OPT_k = \sum_{C \in (C_k \cap \mathbf{C}_k^*)} \gamma(C)$. Because C_k is the maximum weight cycle in this iteration, the weight of the cycles in $C_k \cap \mathbf{C}_k^*$ is smaller than APX_k. Moreover, since there are $|C_k|$ vertices in cycle C_k, there are at most $|C_k|$ cycles in $C_k \cap \mathbf{C}_k^*$. Hence, $OPT_k \leq APX_k \cdot |C_k|$, which results in the approximation ratio of the maximum cycle length in the weight dependency graph. □

In the next example, we show that the approximation ratio proven in Theorem 53 is tight.

Example 54 (Tight example of Alg. 5)**.** Consider the setting depicted in Fig. 4.4(a). For this setting, Alg. 5 results in the following transmissions $p_1 + p_2$, $p_2 + p_3$, and $p_3 + p_4$. Note that these transmissions correspond to cycle (c_1, c_2, c_3, c_4) in Fig. 4.4(b). In contrast, the optimal solutions is $p_1 + p_5$, $p_2 + p_6$, $p_3 + p_7$, and $p_4 + p_8$. Therefore, the approximation ratio is $4(1 - \epsilon)/1$ for any $\epsilon > 0$. ◀

4.7 Multiple multicast with immediate decoding

In this subsection, we focus on the spare solution and the immediate decoding in multiple multicast scenario. We will show that the PCIC problem is not only

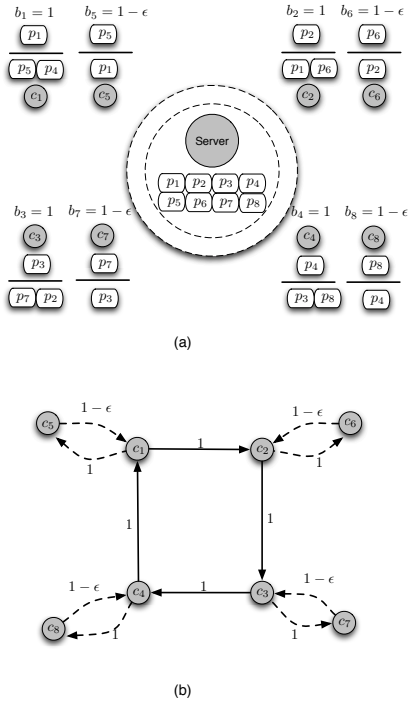

Figure 4.4: (a) Tight example of greedy-VCG-coding scheme (b) The weight dependency graph

$\mathcal{NP}$-hard, but hard to approximate.

We prove the results by presenting a reduction from the Independent Set (IS) problem into the PCIC problem. Given an instance, i.e. a graph $G(V, E)$, of the IS problem, we construct an instance of the PCIC problem as follows. For each vertex $v \in V$ and edge $e \in E$, we create packet p_v and p_e. The packet set P is consist of p_v and p_e for all $v \in V$ and $e \in E$. For each edge $e = (u, v) \in E$, three clients $c_{e,1}, c_{e,2}, c_{e,3}$ are defined such that:

- $w_{e,1} = p_e$, $H_{e,1} = \{p_u, p_v\}$, $b_{e,1} = 1$
- $w_{e,2} = p_u$, $H_{e,2} = \{p_e\}$, $b_{e,2} = 1/\deg(u)$
- $w_{e,3} = p_v$, $H_{e,3} = \{p_e\}$, $b_{e,3} = 1/\deg(v)$

Three technical results are provided before the main proof.

Lemma 55. *In the resulted instance of the PCIC problem, every client will get the request packet by using the optimal encoding matrix G^*.*

Proof. It is clear that, for any $e \in E$, client $c_{e,1}$ is guaranteed to obtain the desired packet owing to $b_{e,1} = 1$.

Suppose that, for some $e = (u, v) \in E$, client $c_{e,2}$ could not get the desired packet when G^* is applied. Let $C_u = \{c_{e,2} : e \in E \text{ incident to } u\}$. Two cases are considered.

First, if all clients in C_u do not obtain the desired packet, another packet p_u can be added to G^* since the additional contribution of the bids is one, which is equal to the transmission cost of p_u.

Then, we study the case that some client in C_u (say client $c_{\tilde{e},2}$) gets the desired packet. Since client $c_{e,2}$ can not recover the the desired packet p_u, server will encode $p_u + p_{\tilde{e}}$ for client $c_{\tilde{e},2}$. However, we can replace $p_u + p_{\tilde{e}}$ by p_u, and $w(B, G^*)$ will be

increased by at least $1/\deg(u)$ because of the contribution of $b_{e,2}$. Then we construct the contradiction to the optimality of G^*, and conclude that every client will get the desired packets by applying the optimal encoding matrix. □

Lemma 56. *In the resulted instance of the PCIC problem, it holds that $\eta^* \leq |E| + OPT_{vc}$, where η^* is the minimum number of transmissions such that every client obtains the request packet, and OPT_{vc} is the optimal solution to the Vertex Cover problem.*

Proof. Let $V^* \subseteq V$ be the optimal solution to the Vertex Cover problem. Server encodes as following rule.

- For each vertex $v \in V^*$, server transmits packet p_v;
- For each edge $e = (u, v) \in E$,
  - If $u, v \in V^*$, server transmits packet p_e;
  - Otherwise (assume $v \in V^*$), server transmits packet $p_u + p_e$.

Transmitting packets p_e or $p_u + p_e$ for all $e \in E$ get client $c_{e,1}$ satisfied. If vertex $\xi \in V^*$, client $c_{e,2}$ or $c_{e,3}$ with e incident to ξ is satisfied due to the transmission of p_ξ; otherwise, they can obtain the request packets by combining the packet $p_\xi + p_e$ and the side information p_e. By transmitting $|E| + OPT_{\text{vc}}$ packets from server, all clients are assured to get the request packets. Hence, $\eta^* \leq |E| + OPT_{\text{vc}}$. □

Lemma 57. *In the resulted instance of the PCIC problem, it holds that $\eta^* \geq |E| + OPT_{vc}$, where η^* is the minimum number of transmissions such that every client obtains the request packet, and OPT_{vc} is the optimal solution to the Vertex Cover problem.*

Proof. We note that in order to satisfy client $c_{e,1}$ for edge $e = (u, v) \in E$, server must send at least one (say q_e) of the packets p_e, $p_e + p_u$, and $p_e + p_v$ due to the immediate decoding. To satisfy client $c_{e,1}$ for all $e \in E$, server needs at least $|E|$ transmissions.

Let $\tilde{V}_e$ include vertex u (and v) if $c_{e,2}$ (and $c_{e,3}$) can not recover the wanted packet p_u (and p_v) from q_e. Let $\tilde{V} = \cup_{e \in E} \tilde{V}_e$. For each vertex $\tilde{v} \in \tilde{V}$, there exists some edge $\tilde{e}$ incident to $\tilde{v}$ such that $c_{\tilde{e},2}$ or $c_{\tilde{e},3}$ can not recover the wanted packet from $q_{\tilde{e}}$. To satisfy it, at least one of packets $p_{\tilde{v}}$ and $p_{\tilde{v}} + p_{\tilde{e}}$ (different from $q_{\tilde{e}}$) is required because of the immediate decoding. Hence, to satisfy these clients that can not successful decode from q_e for all $e \in E$, server has to send at least another $|\tilde{V}|$ packets.

Since q_e can only help at most one of clients $c_{e,2}$ and $c_{e,3}$ recover the wanted packet, then $\tilde{V}$ is a vertex cover. To satisfy all clients, $\eta^* \geq |E| + |\tilde{V}| \geq |E| + OPT_{\text{vc}}$. □

Theorem 58. *In the multiple multicast scenario with the restriction of the sparse code and the immediate decoding, the PCIC problem is $\mathcal{NP}$ hard and hard to approximate within the factor $n^{1-\epsilon}$ for any constant $\epsilon > 0$.*

Proof. From Lemma 55, every client will successful recover the request packet by using the optimal encoding matrix G^*, and hence

$$\begin{aligned} w(B, G^*) &= \sum_{e \in E} b_{e,1} + \sum_{e \in E} (b_{e,2} + b_{e,3}) - \eta^* && (4.3) \\ &= |E| + |V| - \eta^*, && (4.4) \end{aligned}$$

where, due to the constant of $|E| + |V|$, G^* agrees with the minimum number of transmissions η^* to satisfy all clients. From Lemma 56 and 57, $\eta^* = |E| + OPT_{\text{vc}}$, and OPT_{vc} is the optimal solution to the Vertex Cover problem in $G(V, E)$. Let $OPT = w(B, G^*)$, and then $OPT = |V| - OPT_{\text{vc}}$.

Let OPT_{is} be the optimal solution to the IS problem. Since $OPT_{\text{is}} + OPT_{\text{vc}} = |V|$,

we obtain that $OPT = OPT_{\text{is}}$. Because of the hardness of the IS problem [46], the result follows. □

4.8 Multiple multicast with general linear decoding

Finally, we claim that, by using the general linear decoding function, the PCIC problem in multiple multicast scenario is also $\mathcal{NP}$-hard and hard to approximate. The proof is similar to Theorem 58, but need more explains to Lemma 57 as follows.

Lemma 59. *Consider the sparse code and the general linear decoding function. Based on the reduction in the proof of Theorem 58, $\eta^* \geq |E| + OPT_{vc}$, where η^* is the minimum number of transmissions such that every client obtains the request packet, and OPT_{vc} is the optimal solution to the Vertex Cover problem.*

Proof. To satisfy client $c_{e,1}$ for edge $e \in E$, server must send at least one (say q_e) of the packets $p_e + p$ for some $p \in P$. Note that if $q_{e_1} = q_{e_2} = p_{e_1} + p_{e_2}$ for some $e_1, e_2 \in E$, at least two more transmissions are required for c_{e_1} and c_{e_2} respectively, because they do not have p_{e_2} and p_{e_1} respectively as the side information. Without loss of generality, we then assume q_e is different for all $e \in E$. Therefore, to satisfy client $c_{e,1}$ for all $e \in E$, server needs at least $|E|$ transmissions.

We use the same definition of the set $\tilde{V}$ in Lemma 57. For each vertex $\tilde{v} \in \tilde{V}$, there exists some edge $\tilde{e}$ incident to $\tilde{v}$ such that $c_{\tilde{e},2}$ or $c_{\tilde{e},3}$ can not recover the wanted packet from $q_{\tilde{e}}$. To satisfy it, at least one (say $q_{\tilde{v}}$) of packets $p_{\tilde{v}} + \tilde{p}$ (different from $q_{\tilde{e}}$) for some $\tilde{p} \in P$ is required. Let edges $\tilde{e}_1, \tilde{e}_2$ incident to $\tilde{v}_1, \tilde{v}_2 \in \tilde{V}$ respectively, such that $c_{\tilde{e}_1,2}$ (or $c_{\tilde{e}_1,3}$) and $c_{\tilde{e}_2,2}$ (or $c_{\tilde{e}_2,3}$) are not satisfied from $q_{\tilde{e}_1}$ and $q_{\tilde{e}_2}$. If $q_{\tilde{v}_1} = q_{\tilde{v}_2} = p_{\tilde{v}_1} + p_{\tilde{v}_2}$, two more transmissions are needed because $c_{\tilde{e}_1,2}$ (or $c_{\tilde{e}_1,3}$) and $c_{\tilde{e}_2,2}$ (or $c_{\tilde{e}_2,3}$) do not have $p_{\tilde{v}_2}$ and $p_{\tilde{v}_1}$ as the side information. Hence, to satisfy these clients that can not successful decode from q_e for all $e \in E$, server has to send at least another $|\tilde{V}|$ packets.

Since q_e can only help at most one of clients $c_{e,2}$ and $c_{e,3}$ recover the wanted packet, then $\tilde{V}$ is a vertex cover. To satisfy all clients, $\eta^* \geq |E| + |\tilde{V}| \geq |E| + OPT_{\text{vc}}$. $\square$

4.9 Numerical results

In this subsection, we numerically study the performance of the proposed greedy-VCG-coding scheme in multiple unicast scenario. The greedy-VCG-coding scheme is designed to approach the optimal social welfare when clients apply the general linear decoding function. First, we evaluate the social welfare $\sum_{i=1}^{n} v_i \cdot \theta_i - \eta$ of the greedy-VCG-coding scheme and compare it with the optimal social welfare when clients are restricted to the immediate decoding, which optimal coding scheme is tractable in polynomial time (Theorem 48).

Fig. 4.5 shows the results, and the experiment setting is as follows. We consider n clients (x-axle), where client c_i requires packet p_i. The value v_i of packet p_i is uniformly distributed between 0 and 1. Let bid $b_i = v_i$ due to the truthfulness. The fixed number of side packets are considered, i.e. $|H_i| = 3$ or $|H_i| = 6$. The side information of client c_i is randomly selected from $P \setminus \{p_i\}$, e.g. when $|H_i|$=3, three packets are chosen randomly for client c_i. Throughout this subsection, each value in the simulation plots represents the average over 500 runs. In Fig. 4.5, though the greedy-VCG-coding is the approximation algorithm, for most cases the greedy-VCG-coding associated with the general linear decoding outperforms the optimal encoding matrix associated with the immediate decoding. For immediate decoding, the optimal encoding matrix only focuses on the vertex-disjoint cycle of length two in the weight dependency graph. When the number of clients is relatively fewer, more cycles would be of length two. Accordingly, the performance of the greedy-VCG-coding is slightly worse when $|H_i| = 6$ and $n = 10, 16$.

In multiple unicast scenario, the social welfare is zero when server transmits

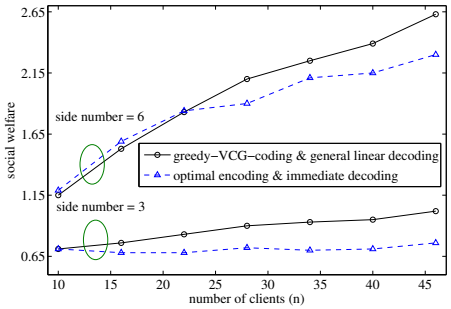

Figure 4.5: Social welfare: greedy-VCG-coding & general linear decoding vs. optimal encoding & immediate decoding

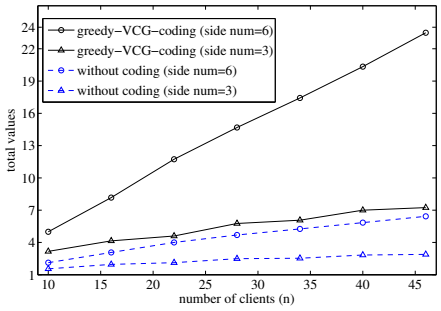

Figure 4.6: Total values: greedy-VCG-coding vs. without coding

packets without coding. To display the advantage of coding, we show in Fig. 4.6 the total values $\sum_{i=1}^{n} v_i \cdot \theta_i$ of the packets. Given an instance of the PCIC problem, we first run the greedy-VCG-coding algorithm and calculate the total values as well as the required number of transmissions η. Then, we explore the maximum total values of packets when server does not apply coding and is restricted to η transmission. It can be observed that the greedy-VCG-coding is not only improve the social welfare, but also the total values of packets.

5. INCENTIVE-COMPATIBLE AND NON-MONETARY DIRECT DATA EXCHANGE WITH NETWORK CODING*

5.1 Introduction

There is a growing interest in wireless device-to-device (D2D) communication between mobile clients. In a typical scenario, the clients form a small ad-hoc network to exchange the data directly with each other. For example, a small number of mobile devices can exchange the data over a local network (such as Wi-Fi or Bluetooth) to reduce delays and minimize the load on more expensive long-range cellular networks. Communication over a local network has many advantages, such as reduced power consumption and lower delays. The advantages of wireless D2D communications have been demonstrated in several recent studies [47–50].

We focus on data exchange between a group of *selfish* wireless clients. Each client initially holds a subset of files and is interested in some of the files held by other clients. The clients use the network coding technique to increase the efficiency of data exchange. For example, Fig. 5.1 shows three clients c_1, c_2, and c_3 that need files p_1, p_2, and p_3 and have file sets $\{p_2, p_3\}, \{p_1, p_3\}, \{p_1, p_2\}$ available to them as a side information, respectively, i.e., client c_1 holds files p_2 and p_3, client c_2 holds files p_1 and p_3, and client c_3 holds files p_1 and p_2. Suppose that each of the files can be delivered to all the clients with one transmission. Without network coding, at least three transmissions are necessary to satisfy the demands of all clients. By using the network coding technique, all clients can be satisfied by just two transmissions. Indeed, it is easy to verify that if client c_1 transmits a linear combination of p_2 and

* Part of the data reported in this section is reprinted with permission from "Truthful and Non-Monetary Mechanism for Direct Data Exchange" by I-Hong Hou, Yu-Pin Hsu, and Alex Sprintson, 2013. In *Allerton Conference on Communication, Control, and Computing (Allerton)*, 406-412, Copyright 2013 by IEEE.

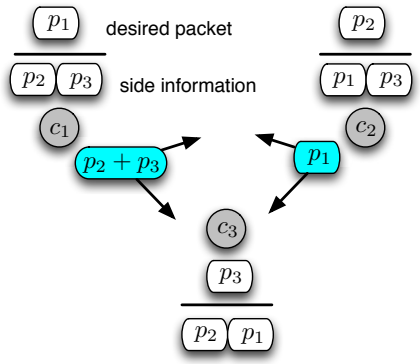

Figure 5.1: Coded information exchange between clients c_1, c_2, c_3.

p_3, and client c_2 transmit p_1, all three clients will be able to decode required packets.

While the problem of minimizing the number of transmissions with network coding has been considered in the previous studies [3–5], the problem of mechanism design for the settings with selfish clients remained open. Indeed, a selfish client might choose to become a "free rider" by choosing not to transmit or make as few transmissions as possible. Indeed, in some cases such clients will be able to decode all packets they need without making a single transmission (e.g., client c_3 in Fig. 5.1). This, in turn, could affect the data exchange process with negative consequences for all clients. Furthermore, different clients might have different internal values for the packets they request and the clients with low valuations are less likely to participate in the data exchange process. Since the clients are selfish, they may not reveal their internal valuation to other clients if this does not benefit them. Accordingly, in this section, we focus on design and analysis of a *incentive-compatible* mechanism under which each client optimizes its utility by reporting the true valuations of the required packets.

We focus on a setting in which data exchange is managed by a distinguished client (referred to as a *broker*). In the beginning of data transfer the broker ob-

tains a *bid* from each client which specifies its maximum *transmission rate* (i.e., the ratio of transmitted and received packets). After receiving the bids, the broker determines the transmission rate of each client as well as the network coding scheme. In a incentive-compatible mechanism, the clients have incentive to bid the maximum transmission rate that corresponds to their valuations of the packets. A popular incentive-compatible mechanism is the Vickery-Clarke-Groves (VCG) mechanism [41–43]. However, VCG mechanism and its variations involve monetary transactions between clients and the broker, hence they cannot be applied to the problem at hand.

5.1.1 Main results

In this section, we propose a incentive-compatible non-monetary mechanism for the broker. We focus on the case in which each client needs a unique file and each transmission can be received by all clients in the system (i.e., all the clients are in close proximity of each other). We then show that our mechanism can be extended for a setting where a transmission of a client can only be received by some of clients in the group (i.e., some clients are too far away to receive the transmissions from each other) and several clients require the same file. Since our mechanism is non-monetary and does not require additional infrastructure, it can be easily implemented in practical settings. We further demonstrate that this mechanism can be implemented in a fully distributed fashion without the presence of the broker.

We further study the performance of our mechanism in terms of the *social welfare*, i.e., the total value of the decoded files minus the total cost of all transmissions. We establish an upper bound of $\mathcal{O}(\log N)$ on the difference between the optimal social welfare and the social welfare achieved by our algorithm, where N is the number of clients in the group. Through simulations, we also demonstrate that this difference

is actually very small for practical settings.

5.2 System overview

Consider a wireless network that consists of N clients $\Lambda = \{c_1, \cdots, c_N\}$ that need to exchange a set P of files. We assume that client c_i needs file $p_i \in P$, and has side information $H_i \subseteq P$ available to it. We can assume without loss of generality that each client requires a single file since a client that requires more than one file can be substituted by multiple clients that share the same side information set. Each client c_i has a private value v_i, where $0 \leq v_i \leq 1$, for file p_i that captures its benefit of receiving p_i.

Clients are able to obtain the needed files via data exchanges over a wireless broadcast channel. The transmission process between clients is mediated by a *broker*. The broker is only conceptual since we will show in Subsection 5.5 that the transmission process can be determined in a fully distributed fashion. We assume that files are of the same size and each contains Z packets. We will address both cases where Z is a large number and $Z = 1$. When Z is large, clients exchange large files. On the other hand, $Z = 1$ corresponds to the scenario where each client only needs a small number of packets. One application is that a base station broadcasts a video stream to a number of clients. Since wireless transmissions may be unreliable, each client may miss a small number of packets. Clients then use D2D communications to recover these missing packets.

We consider that clients need to pay some *transmission cost* for the packets they transmit. Specifically, we define the *upload ratio* r_i of client c_i as

$$\frac{\text{number of packets transmitted by } c_i}{Z}$$

. The transmission cost of client c_i is then assumed to be $C(r_i)$, where $C(\cdot)$ is a

non-decreasing and convex function with $C(0) = 0$ and $C(1) = 1$. The transmission cost of a client can be interpreted as, for example, the cost of battery power or the cost of channel access for transmitting packets. We denote by $R = \{r_1, \cdots, r_N\}$ the array that contains the upload ratios of all clients.

In the beginning of the data exchange, each client is required to submit a bid b_i to the broker. The bid specifies the maximum transmission cost the client is willing to incur in exchange for file p_i. We denote by $B = \{b_1, \cdots, b_N\}$ the array that contains the bids of the clients. Based on the clients' bids B, the broker decides for each client c_i a *transmission schedule* that consists of the upload ratio r_i, as well as the combination of the packets client c_i needs to transmit. When Z is large, r_i can be chosen to be any value between $[0, 1]$. On the other hand, when $Z = 1$, r_i can only be 0 or 1. In the case of $Z = 1$, the broker may determine transmission schedules randomly.

Given the transmission schedules, we can determine if client c_i is able to decode the required file p_i. We require that a client c_i with $r_i > 0$ must be able to decode p_i. We defined by χ_i the indicator function to specify if client c_i can decode p_i. The *net utility* $u_i(B)$ of c_i can then be written as $u_i(B) = v_i \cdot \chi_i - C(r_i)$. Note that the net utility depends on the bids of other clients as well as the broker's mechanism for determining transmission schedules.

The algorithm for determining the transmission schedules is known to all the parties. Each client c_i is considered to be selfish and chooses its best bidding policy b_i that maximizes its utility $u_i(B)$, or $E[u_i(B)]$ when the broker makes decisions randomly. We say that a mechanism is *incentive-compatible* if each client maximizes its own net utility by choosing $b_i = v_i$.

Definition 60. Let $B_{-i} = B \backslash \{b_i\}$. A mechanism is incentive-compatible if, for all b_i

and B_{-i}, $u_i(v_i, B_{-i}) \geq u_i(b_i, B_{-i})$, or $E[u_i(v_i, B_{-i})] \geq E[u_i(b_i, B_{-i})]$ when the broker makes decisions randomly, under it.

In this section, our goal is to design an incentive-compatible broker mechanism that does not involve in any monetary transactions. The performance of the proposed mechanism is further evaluated in terms of the *total social welfare*, defined as $\sum_{i=1}^{N} v_i \cdot \chi_i - \sum_{i=1}^{N} C(r_i)$, which is the sum of net utilities over all clients.

5.3 Feasibility of the transmission schedule

The transmission schedule decided by the broker consists of two parts: the upload ratio r_i for each client c_i, and the combinations of the packets that c_i has to transmit. When the vector R of the upload ratios is given, one important question is whether there exists a transmission schedule under which a subset $S \subseteq \Lambda$ of clients can successfully decode their needed files. We establish a sufficient condition for the existence of such a schedule.

Lemma 61. *Assume that $p_i \neq p_j$, for all $i \neq j$. Given a vector R of upload ratios and a subset $S \subseteq \Lambda$ with $r_i = 0$ for all $c_i \notin S$. Suppose that there exists a partition of $S = S_1 \cup S_2 \cup \cdots \cup S_k$ with $S_m \cap S_n = \phi$, for all $m \neq n$, such that, for each $S_m \in S$ and for each $c_i \in S_m$, we have*

1. *$\{p_i\} \in H_j$, for all $c_j \in S_m$, and $j \neq i$, and*
2. *$\sum_{c_j \in S_m \setminus \{c_i\}} r_j \geq 1$.*

Then there exists a transmission schedule such that

i) The upload ratio of each client c_i is r_i.

ii) All clients in S are able to decode their respective needed files.

iii) None of the clients outside S can decode their needed files.

Proof. We prove this lemma by constructing a transmission schedule that satisfies the conditions of the lemma. Let client $c_i \in S_m$ transmit Zr_i coded packets, each containing a linear combination of one packet from each of the files $\{p_j : c_j \in S_m \backslash \{c_i\}\}$. This can be achieved because $\{p_j : c_j \in S_m \backslash \{c_i\}\} \subseteq H_i$ by property (1).

Now, for a client $c_j \in S_m$, it receives Zr_i coded packets from c_i, for each other $c_i \in S_m$. Each of these coded packets contains a linear combination of one packet from each of $\{p_k : c_k \in S_m \backslash \{c_i\}\}$. Since c_j have the files $\{p_k : c_k \in S_m \backslash \{c_i, c_j\}\}$ as its side information, by property (1), it can employ Gaussian elimination to obtain Zr_i packets of the file p_j, for each $c_i \in S_m$. Thus, c_j obtains a total number of $\sum_{i \in S_m, i \neq j} Zr_i \geq Z$ packets of p_j, and successfully receives the whole file p_j.

On the other hand, in the above transmission schedule, no clients transmit any packets that involve $\{p_j : c_j \notin S\}$. Therefore, none of the clients outside S can receive their needed files. □

Given a vector R of upload ratios and a subset $S \subseteq \Lambda$ with $r_i = 0$, for all $c_i \notin S$, if the condition in Lemma 61 is satisfied, we say (R, S) is *feasible*. The notion of Lemma 61 is that if (R, S) is feasible, then each client $c_i \in S$ can receive enough amount of packets to recover its needed file p_i (i.e. $\chi_i = 1$).

For the special case where $H_i = P \backslash \{p_i\}$, for all i, Lemma 61 can be greatly simplified and strengthened as follows:

Corollary 62. *Suppose $H_i = P \backslash \{p_i\}$, for all i. Consider a vector R of upload ratios, and a subset $S \subseteq \Lambda$ with $r_i = 0$, for all $c_i \notin S$. Then (R, S) is feasible if and only if*

$$\sum_{c_j \in S \backslash \{c_i\}} r_j \geq 1,$$

for all $c_i \in S$.

Proof. By Lemma 61, it is straightforward to show that the condition is sufficient. To show that it is necessary, assume that there exists $c_i \in S$ with $\sum_{c_j \in S \setminus \{c_i\}} r_j < 1$, then the total number of packets that c_i receives is no larger than $(\sum_{c_j \in S \setminus \{c_i\}} Z r_j) < Z$. Hence, it is impossible for c_i to obtain all packets in p_i. □

5.4 Infeasibility of VCG-based mechanism

A classical solution for the incentive-compatible mechanism design is to apply the Vickery-Clarke-Groves (VCG) mechanism. However, VCG-based mechanisms are not suitable for solving our problem. For clarity, we briefly discuss the design of the VCG mechanism below.

Consider a system where a centralized server decides which set of clients to serve. When it serves a set S of clients, it needs to pay some cost, defined as $cost(S)$. Each client c_i has a secret valuation $\hat{v}_i$ for being served. The *social welfare* of serving S is defined as $\sum_{c_i \in S} \hat{v}_i - cost(S)$. The VCG mechanism then works as follows: First, each client submits its valuations $\hat{b}_i$ to the server. Note that we do not require $\hat{b}_i = \hat{v}_i$, as clients may lie about their valuations. Then, the server does the following steps:

1. It chooses the set S that maximizes $\sum_{i \in S} \hat{b}_i - cost(S)$.

2. It charges each scheduled client c_i an amount of money as follows.

$$[(\max_{S': c_i \notin S'} \sum_{c_j \in S'} \hat{b}_j - (cost(S'))) - (\sum_{c_j \in S, j \neq i} \hat{b}_j - (cost(S)))]^+,$$

   where $x^+ := \max\{x, 0\}$. The expression for the charge is called the *critical price*, since it is the minimum value that client c_i needs to bid in order to be scheduled in the previous step.

It is well-known that the VCG mechanism is incentive-compatible [44]. However, the VCG mechanism involves monetary exchanges, which requires additional

infrastructure. Hence, it is not suitable for direct data exchange in wireless ad hoc networks.

One may wonder whether there exists simple non-monetary adaptations of the VCG mechanism to our problem. We can consider one with the following setting: We treat the server as a broker, $cost(S)$ to be the minimum total upload ratios required to make (R, S) feasible. To be more specific, $cost(S) \equiv \min_{R:(R,S) \text{ is feasible}} \sum_i r_i$. With this interpretation, the definitions of social welfare in the VCG mechanism and in our setting are equivalent. In the second step of the VCG mechanism, the sever charges each client some money. One naive adaptation is to treat the charges of clients as the upload ratios in our system, i.e., $r_i \equiv [(\max_{S':c_i \notin S'} \sum_{c_j \in S'} \hat{b}_j - (cost(S'))) - (\sum_{c_j \in S, j \neq i} \hat{b}_j - (cost(S)))]^+$, for all $c_i \in S$. However, with this adaptation, we fix the vector R of upload ratios, and there is no guarantee that (R, S) is still feasible. A numerical example that demonstrates this intuition is shown below.

Example 63. Consider three clients with the identical valuation $v_i = 0.6$, and the side information $H_i = P \backslash \{p_i\}$. Let the cost function be $C(x) = x$. In the first step, the broker decides that the optimal decision is to schedule all three clients. Indeed, by Corollary 62, there exists a transmission schedule with $r_i = 0.5$, for all i, under which all three clients get their needed files, and the social welfare is $(0.6 - 0.5) \times 3 = 0.3$. However, the second step of the VCG-based algorithm results in $r_i = 0.3$ for all i, and, by Corollary 62, the set of all clients with such upload ratios are no longer feasible. ■

5.5 Incentive-compatible mechanisms

We first focus on the setting that each client c_i has all files other than p_i, i.e. $H_i = P \backslash \{p_i\}$, while the more general model is discussed in Subsection 5.7. We propose mechanisms for both cases where the number of packets in a file, Z, is large,

and where $Z = 1$.

5.5.1 Mechanism for exchanging large files

The broker mechanism for large Z is designed as follows: First, we sort bids in descending order such that $b_1 \geq b_2 \geq \ldots$. We then find the maximum number $i^* \in \{1, \cdots, N\}$ with $b_{i^*} \geq C(\frac{1}{i^*-1})$. A client whose bid is greater than or equal to b_{i^*} is included in the set S^*, and is scheduled to transmit with the upload ratio $r_i = \frac{1}{|S^*|-1}$. All other clients neither transmit nor receive their needed files. We notice that no monetary transaction occurs during the process. This mechanism is formally stated in Alg. 7.

Algorithm 7: Broker mechanism for large files

input : Bids vector B, and side information H_i for all clients
output: Transmission schedules

1 Sorting bids: $b_1 \geq b_2 \geq \cdots \geq b_N$;
2 Find $i^* = \max\{i : b_i \geq C(\frac{1}{i-1})\}$;
3 $S^* \leftarrow \{c_1, \cdots, c_{i^*}\}$;
4 $r_i \leftarrow \frac{1}{|S^*|-1}$ for $c_i \in S^*$;
5 $r_i \leftarrow 0$ for $c_i \notin S^*$;
6 Client $c_i \in S^*$ transmits r_i proportion of the coded file $\sum_{p \in \{p_1, \cdots, p_{i^*}\} \setminus \{p_i\}} p$;

The following lemma is a direct result from Corollary 62.

Lemma 64. (R, S^*) *produced from Alg. 7 is feasible.*

Next, we will show that Alg. 7 is also incentive-compatible.

Theorem 65. *Alg. 7 is incentive-compatible.*

Proof. Consider a client $c_{\hat{i}}$, where the index of $\hat{i}$ is set to be the position of $c_{\hat{i}}$ in Line 1 of Alg. 7 when it bids $b_{\hat{i}} = v_{\hat{i}}$. That is, we have $b_1 \geq b_2 \geq \ldots b_{\hat{i}-1} \geq v_{\hat{i}} \geq b_{\hat{i}+1} \ldots$.

Let $S^*(b_{\hat{i}})$ be the set S^* under Alg. 7 when $c_{\hat{i}}$ bids $b_{\hat{i}}$. We consider the following two cases:

Case (1) $c_{\hat{i}} \notin S^*(v_{\hat{i}})$, i.e., $c_{\hat{i}}$ is not scheduled by bidding its true value: By the design of Alg. 7, we have that $v_{\hat{i}} < C(\frac{1}{\hat{i}-1})$, and $b_i < C(\frac{1}{i-1})$ for all $i > \hat{i}$. Also, the net utility of $c_{\hat{i}}$ is 0 when it bids its true value.

Suppose $c_{\hat{i}}$ bids $b_{\hat{i}} \neq v_{\hat{i}}$, we consider three possibilities based on the position of $b_{\hat{i}}$ in Line 1 of Alg. 7: $b_{\hat{i}-1} \geq b_{\hat{i}} \geq b_{\hat{i}+1}$, i.e., the position is the same as that under the true value; $b_k \geq b_{\hat{i}} \geq b_{k+1}$, for some $k < \hat{i} - 1$, i.e., the position is higher than that under the true value; and $b_k \geq b_{\hat{i}} \geq b_{k+1}$, for some $k > \hat{i}$.

In the first and second cases, since the positions of all clients c_i with $i > \hat{i}$ are the same as those when $c_{\hat{i}}$ bids its true value, they are still not scheduled by Alg. 7. Hence, we have $|S^*(b_{\hat{i}})| \leq \hat{i}$, and $r_i \geq \frac{1}{\hat{i}-1}$, for all $i \in S^*(b_{\hat{i}})$. The net utility of $\hat{i}$ is then no larger than $\max\{0, v_{\hat{i}} - C(\frac{1}{\hat{i}-1}\}) \leq 0$, as $v_{\hat{i}} < C(\frac{1}{\hat{i}-1})$.

In the third case, client $c_{\hat{i}}$ is now placed on the k^{th} position with $k > \hat{i}$. We have $b_{\hat{i}} \leq b_k < C(\frac{1}{k-1})$. Hence, $c_{\hat{i}}$ is still not scheduled and have net utility 0. In sum, when $c_{\hat{i}} \notin S^*(v_{\hat{i}})$, $c_{\hat{i}}$ cannot improve its own net utility by lying about its $v_{\hat{i}}$.

Case (2) $c_{\hat{i}} \in S^*(v_{\hat{i}})$, i.e., $c_{\hat{i}}$ is scheduled by bidding its true value: Let $i^* = |S^*(v_{\hat{i}})|$. We have $v_{\hat{i}} \geq b_{i^*} \geq C(\frac{1}{i^*-1})$, and the net utility of $c_{\hat{i}}$ is greater or equal to 0. Now, by the design of Alg. 7, we have $b_i < C(\frac{1}{i-1})$ for all $i > i^*$, and each client c_i with $i > i^*$ will not be scheduled regardless of $b_{\hat{i}}$. Therefore, $|S^*(b_{\hat{i}})| \leq i^*$, and $r_i \geq C(\frac{1}{i^*-1})$, for $i \in S^*(b_{\hat{i}})$, regardless of $b_{\hat{i}}$. The net utility of $c_{\hat{i}}$ is then upper-bounded by $v_{\hat{i}} - C(\frac{1}{i^*-1}) \geq 0$, which is its net utility when bidding $b_{\hat{i}} = v_{\hat{i}}$.

Fully consider cases 1 and 2, we conclude that Alg. 7 is an incentive-compatible mechanism. □

In addition to being incentive-compatible, we note that Alg. 7 can also be easily

implemented in a fully distributed fashion. Instead of relying on the broker, each client simply broadcasts its bid b_i. After receiving the values of all bids, each client determines how many, and what, packets it should transmit by running Alg. 7. Thus, our mechanism can still be implemented without the presence of the broker.

5.5.2 Mechanism for $Z = 1$

When $Z = 1$, we require that $r_i \in \{0, 1\}$, for all i. The broker mechanism for $Z = 1$ is designed as follows: We also sort bids such that $b_1 \geq b_2 \geq \ldots$. We then find the maximum number i^* with $b_{i^*} \geq \frac{2}{i^*} C(1) = \frac{2}{i^*}$, and select the set $S^* = \{c_1, c_2, \ldots, c_{i^*}\}$. We select two clients from S^* uniformly at random and make each of them transmit a linear combination of files needed by all other clients in S^*. Therefore, all clients in S^* obtain their needed files, while none of the clients outside S^* obtain any files. Further, as each client in S^* has $\frac{2}{i^*}$ chance of being selected to transmit, the expected net utility of a client $c_i \in S^*$ is $v_i - \frac{2}{i^*}$. Alg. 8 formally specifies the mechanism.

Algorithm 8: Broker mechanism for $Z = 1$

input : Bids vector B, and side information H_i for all clients
output: Transmission schedules

1 Sorting bids: $b_1 \geq b_2 \geq \cdots \geq b_N$;
2 Find $i^* = \max\{i : b_i \geq \frac{2}{i^*}\}$;
3 $S^* \leftarrow \{c_1, \cdots, c_{i^*}\}$;
4 Randomly select $c_{i_1}, c_{i_2} \in S^*$;
5 c_{i_1} transmits a coded packet $\sum_{p \in \{p_1, \cdots, p_{i^*}\} \setminus \{p_{i_1}\}} p$;
6 c_{i_2} transmits a coded packet $\sum_{p \in \{p_1, \cdots, p_{i^*}\} \setminus \{p_{i_2}\}} p$;

Theorem 66. *Alg. 8 is incentive-compatible.*

Proof. The proof is similar to that of Theorem 65. Consider a client $c_{\hat{i}}$, where the index of $\hat{i}$ is set to be the position of $c_{\hat{i}}$ in Line 1 of Alg. 8 when it bids $b_{\hat{i}} = v_{\hat{i}}$. That

is, we have $b_1 \geq b_2 \geq \ldots b_{\hat{i}-1} \geq v_{\hat{i}} \geq b_{\hat{i}+1} \ldots$. Let $S^*(b_{\hat{i}})$ be the set S^* under Alg. 8 when $c_{\hat{i}}$ bids $b_{\hat{i}}$. We consider the following two cases:

Case (1) $c_{\hat{i}} \notin S^*(v_{\hat{i}})$, i.e., $c_{\hat{i}}$ is not scheduled by bidding its true value: By the design of Alg. 8, we have that $v_{\hat{i}} < 2/\hat{i}$, and $b_i < 2/i$ for all $i > \hat{i}$. Also, the net utility of $c_{\hat{i}}$ is 0 when it bids its true value.

Suppose $c_{\hat{i}}$ bids $b_{\hat{i}} \neq v_{\hat{i}}$, we consider three possibilities based on the position of $b_{\hat{i}}$ in Line 1 of Alg. 8: $b_{\hat{i}-1} \geq b_{\hat{i}} \geq b_{\hat{i}+1}$, i.e., the position is the same as that under the true value; $b_k \geq b_{\hat{i}} \geq b_{k+1}$, for some $k < \hat{i} - 1$, i.e., the position is higher than that under the true value; and $b_k \geq b_{\hat{i}} \geq b_{k+1}$, for some $k > \hat{i}$.

In the first and second cases, since the positions of all clients c_i with $i > \hat{i}$ are the same as those when $c_{\hat{i}}$ bids its true value, they are still not scheduled by Alg. 8. Hence, we have $|S^*(b_{\hat{i}})| \leq \hat{i}$, and the probability that a client is selected to transmit is greater or equal to $2/\hat{i}$. The expected net utility of $\hat{i}$ is then no larger than $\max\{0, v_{\hat{i}} - 2/\hat{i}\}) = 0$, as $v_{\hat{i}} < 2/\hat{i}$.

In the third case, client $c_{\hat{i}}$ is now placed on the k^{th} position with $k > \hat{i}$. We have $b_{\hat{i}} \leq b_k < 2/k$. Hence, $c_{\hat{i}}$ is still not scheduled and have net utility 0. In sum, when $c_{\hat{i}} \notin S^*(v_{\hat{i}})$, $c_{\hat{i}}$ cannot improve its own net utility by lying about its $v_{\hat{i}}$.

Case (2) $c_{\hat{i}} \in S^*(v_{\hat{i}})$, i.e., $c_{\hat{i}}$ is scheduled by bidding its true value: Let $i^* = |S^*(v_{\hat{i}})|$. We have $v_{\hat{i}} \geq b_{i^*} \geq 2/i^*$, and the net utility of $c_{\hat{i}}$ is greater or equal to 0. Now, by the design of Alg. 8, we have $b_i < 2/i$ for all $i > i^*$, and each client c_i with $i > i^*$ will not be scheduled regardless of $b_{\hat{i}}$. Therefore, $|S^*(b_{\hat{i}})| \leq i^*$, regardless of $b_{\hat{i}}$. The net utility of $c_{\hat{i}}$ is then upper-bounded by $v_{\hat{i}} - 2/i^* \geq 0$, which is its net utility when bidding $b_{\hat{i}} = v_{\hat{i}}$.

Fully consider cases 1 and 2, we conclude that Alg. 8 is incentive-compatible. $\square$

5.6 Performance analysis

We have developed two incentive-compatible mechanisms for the cases of large Z and of $Z = 1$, respectively. In this subsection, we study the social welfare, defined as the sum of net utility of all clients, of these two mechanisms by comparing it against theoretical upper-bounds. We call the difference between the social welfare achieved by our mechanisms and their respective performance upper-bounds the *loss of optimality*.

5.6.1 Performance of algorithm 7

We first study the theoretical upper-bound of social welfare when Z is large.

Lemma 67. *Maximum social welfare for large Z is achieved when all clients receive their needed files, or none of the clients receive any files.*

Proof. If, for any subset $S \subseteq \Lambda$, every feasible (R, S) gives rise to a negative social welfare, i.e. $\sum_{c_i \in S} v_i - \sum_{c_i \in S} C(r_i) < 0$, then the maximum social welfare is achieved by $r_i = 0$ for all i, under which case none of the clients receive any files.

Suppose that there is a feasible $(R, \hat{S})$, where $\hat{S} \subseteq \Lambda$, such that $\sum_{c_i \in \hat{S}} v_i - \sum_{c_i \in \hat{S}} C(r_i) \geq 0$. By Corollary 62, we have, for each $c_i \in \Lambda$,

$$\sum_{c_j \in \Lambda \setminus \{c_i\}} r_j = \sum_{c_j \in S \setminus \{c_i\}} r_j \geq 1.$$

Hence, (R, Λ) is also feasible. Further, the social welfare under (R, Λ) is greater or equal to that under $(R, \hat{S})$, since $\sum_{i=1}^{N} v_i - \sum_{c_i \in \hat{S}} C(r_i) \geq \sum_{c_i \in \hat{S}} v_i - \sum_{c_i \in \hat{S}} C(r_i)$. That is, we can construct another transmission schedule that makes all clients receive their needed files, and the social welfare of this transmission schedule is at least as large as that under $(R, \hat{S})$. This concludes the proof. □

Corollary 68. *The maximum social welfare for large Z is* $\max\{0, \sum_{i=1}^{N} v_i - N \cdot C(\frac{1}{N-1})\}$.

Proof. We study the maximum social welfare when all clients receive their needed files. We have

$$\begin{aligned} & \sum_{i=1}^{N} v_i - \sum_{i=1}^{N} C(r_i) \\ \leq & \sum_{i=1}^{N} v_i - N \cdot C\left(\frac{1}{N} \cdot \sum_{i=1}^{N} r_i\right) \quad (5.1) \\ \leq & \sum_{i=1}^{N} v_i - N \cdot C\left(\frac{1}{N-1}\right). \quad (5.2) \end{aligned}$$

We note that (5.1) is based on the convex cost function $C(.)$. On the other hand, (5.2) holds because the feasibility condition $\sum_{c_j \in \Lambda \setminus \{c_i\}} r_j \geq 1$, for all $c_i \in \Lambda$, implies that $\sum_{i=1}^{N} r_i \geq \frac{N}{N-1}$. The equalities (5.1) and (5.2) are achievable when $r_i = 1/(N-1)$ for all $c_i \in \Lambda$. By Lemma 67, the maximum social welfare is $\max\{0, \sum_{i=1}^{N} v_i - N \cdot C(1/(N-1))\}$. □

We are ready to study the loss of optimality of Alg. 7.

Theorem 69. *The loss of optimality of Alg.7 is bounded by $\mathcal{O}(logN)$, when $b_i = v_i$, for all i.*

Proof. If the maximum social welfare is 0, then there is no loss of optimality, as the social welfare under Alg. 7 is always non-negative.

Next consider the case when the maximum social welfare is $\sum_{i=1}^{N} v_i - NC(1/(N-1))$. Sort all clients so that $v_1 \geq v_2 \geq \ldots$. Assume that Alg. 7 schedules a set $S^* = \{1, 2, \ldots, i^*\}$ of clients. We then have $r_i = 1/(i^* - 1)$, for all $i \leq i^*$, and $v_i < C(1/(i-1))$, for all $i > i^*$. The social welfare under Alg. 7 is $\sum_{i=1}^{i^*} v_i - i^* \cdot C(1/(i^*-1))$,

and the loss of optimality is

$$\begin{aligned}
&\sum_{i=1}^{N} v_i - NC(\frac{1}{N-1}) - (\sum_{i=1}^{i^*} v_i - i^* \cdot C(\frac{1}{i^*-1})) \\
\leq &\sum_{i=1}^{N} v_i - \sum_{i=1}^{i^*} v_i + i^* \cdot C(\frac{1}{i^*-1}) \\
= &\sum_{i=i^*+1}^{N} v_i + i^* \cdot C(\frac{1}{i^*-1}) \\
\leq &\sum_{i=i^*+1}^{N} C(\frac{1}{i-1}) + \sum_{i=2}^{i^*} C(\frac{1}{i-1}) + C(1) \\
= &\sum_{i=1}^{N-1} C(\frac{1}{i}) + C(1).
\end{aligned}$$

□

We note that the largest loss of optimality can only be achieved when the values of v_i are carefully selected. In Subsection 5.8, we will demonstrate that the loss of optimality is actually very small when the values of v_i are randomly generated.

5.6.2 Performance of algorithm 8

We now study the theoretical upper-bound of social welfare when $Z = 1$. In this case, we require that $r_i \in \{0, 1\}$, that is, a client either transmits a packet or transmits nothing.

Lemma 70. *The maximum social welfare for $Z = 1$ is at most* $\max\{0, \sum_{i=1}^{N} v_i - 2\}$

Proof. Since we require that a client with $r_i > 0$ must receive its needed file from other clients, we have $\sum_j r_j \geq 2$, if $r_i = 1$, for some i. Therefore, if at least one of the clients transmits, the social welfare is at most $\sum_{i=1}^{N} v_i - 2C(1) = \sum_{i=1}^{N} v_i - 2$. □

Theorem 71. *The loss of optimality of Alg. 8 is bounded by* $1 + \sum_{i=2}^{N} \frac{2}{i}$.

Proof. If the maximum social welfare is 0, then the loss of optimality of Alg. 8 is also 0. Therefore, we only consider the case when the maximum social welfare is larger than 0.

Sort all clients so that $v_1 \geq v_2 \geq \ldots$. Assume that Alg. 8 schedules a set $S^* = \{1, 2, \ldots, i^*\}$. The social welfare achieved by Alg. 8 is then $\sum_{i=1}^{i^*} v_i - 2$. Further, we have $v_i < \frac{2}{i}$, for all $i > i^*$. The loss of optimality is then at most

$$(\sum_{i=1}^{N} v_i - 2) - (\sum_{i=1}^{i^*} v_i - 2) = \sum_{i=i^*+1}^{N} v_i < 1 + \sum_{i=2}^{N} \frac{2}{i}.$$

□

5.7 Extensions

Thus far, we focused on the setting that every client c_i misses a unique file and has all other files in P. We also assume that each client can receive the transmission from each other. In this subsection, we discuss some extensions that can be applied to more general scenarios.

5.7.1 Some clients cannot exchange files with each other

So far, we have assumed that each client can transmit a file that is useful for another client. This may not be true in some realistic settings. For example, some clients may be too far away from each other, and hence can not receive the transmissions by each other. Also, it is possible that client c_i does not possess the file that another client c_j needs.

To model this scenario, we say that $p_j \in H_i$ if and only if c_i has the file p_j, and c_j can receive transmissions from c_i. It is easy to check that Lemma 61 still holds with this slight modification. We then define a *dependency graph* as follows:

Definition 72. The dependency graph is an undirected graph G defined as follows:

- For each client $c_i \in \Lambda$, there is a corresponding vertex in G.
- For any two clients c_i and c_j such that $p_i \in H_j$ and $p_j \in H_i$, there is an edge between the corresponding vertices.

Based on the dependency graph, we propose a broker mechanism in Alg. 9 for the case that Z is large.

Algorithm 9: Broker mechanism for the dependency graph with large Z

input : Bids vector B, and side information H_i for all clients
output: Transmission schedules

```
1  Create the dependency graph G;
2  S* ← φ;
3  r_i ← 0, ∀i;
4  for k ← N to 1 do
5  |  while there exists a k-clique in G such that b_i ≥ C(1/(k−1)) for all
   |  corresponding client c_i in the clique do
6  |  |  S* ← S* ∪ { all corresponding c_i in the clique};
7  |  |  r_i ← 1/(k−1) for all corresponding c_i in the clique ;
8  |  |  Remove the clique from G ;
9  |  end
10 end
```

In Line 5 of the above algorithm, if there are multiple cliques that satisfy the condition, ties are broken by a predetermined order that is independent of the bids of clients.

Theorem 73. *Alg. 9 produces a feasible (R, S^*) and is incentive-compatible.*

Proof. Using Lemma 61, we can establish that Alg. 9 produces a feasible (R, S^*) by treating the partitions $S_1, S_2, \ldots$ in Lemma 61 as the cliques found in Line 5.

Consider a client $c_{\hat{i}}$, and the (R, S^*) produced by Alg. 9 when it bids $b_{\hat{i}} = \infty$. If $c_{\hat{i}}$ is not scheduled by bidding ∞, it will not be schedule regardless of its bid. Hence, its net utility is always 0. On the other hand, suppose that $c_{\hat{i}}$ is scheduled by bidding ∞, and let k be the size of the clique when $c_{\hat{i}}$ is included in S^* in Lines 5 – 9 of Alg. 9. Since ties are broken by a predetermined order when multiple cliques satisfy the condition in Line 5, $c_{\hat{i}}$ will be included in S^* with a clique of size k as long as $b_{\hat{i}} \geq C(\frac{1}{k-1})$, and, if $b_{\hat{i}} < C(\frac{1}{k-1})$, it will not be included in S^*. Therefore, if $c_{\hat{i}}$ is scheduled, its upload ratio is $\frac{1}{k-1}$, regardless of its actual bid. The net utility of $c_{\hat{i}}$ is then upper-bounded by $\max\{0, v_{\hat{i}} - C(\frac{1}{k-1})\}$, which can be attained by bidding $b_{\hat{i}} = v_{\hat{i}}$. In sum, client $c_{\hat{i}}$ cannot improve its net utility by lying about its true value. □

A similar mechanism can be developed for the case that $Z = 1$.

Algorithm 10: Broker mechanism for the dependency graph with $Z = 1$

input : Bids vector B, and side information H_i for all clients
output: Transmission schedules

```
1  Create the dependency graph G;
2  S* ← φ;
3  r_i ← 0, ∀i;
4  for k ← N to 1 do
5  |  while there exists a k-clique in G such that b_i ≥ 2/k for all corresponding
   |  client c_i in the clique do
6  |  |  S* ← S* ∪ { all corresponding c_i in the clique};
7  |  |  Randomly select two clients in the clique and set r_i = 1 for these two
   |  |  clients ;
8  |  |  Remove the clique from G ;
9  |  end
10 end
```

Theorem 74. *Alg. 10 produces a feasible* (R, S^*) *and is incentive-compatible.*

Proof. It is obvious that Alg. 10 produces a feasible schedule. We show that it is also incentive-compatible.

Consider a client $c_{\hat{i}}$, and the (R, S^*) produced by Alg. 10 when it bids $b_{\hat{i}} = \infty$. If $c_{\hat{i}}$ is not scheduled by bidding ∞, it will not be schedule regardless of its bid. Hence, its net utility is always 0. On the other hand, suppose that $c_{\hat{i}}$ is scheduled by bidding ∞, and let k be the size of the clique when $c_{\hat{i}}$ is included in S^*. Since ties are broken by a predetermined order when multiple cliques satisfy the condition in Line 5, $c_{\hat{i}}$ will be included in S^* with a clique of size k as long as $b_{\hat{i}} \geq \frac{2}{k}$, and, if $b_{\hat{i}} < frac2k$, it will not be included in S^*. Therefore, if $c_{\hat{i}}$ is scheduled, its expected transmission cost is $\frac{2}{k}$, regardless of its actual bid. The net utility of $c_{\hat{i}}$ is then upper-bounded by $\max\{0, v_{\hat{i}} - \frac{2}{k})\}$, which can be attained by bidding $b_{\hat{i}} = v_{\hat{i}}$. In sum, client $c_{\hat{i}}$ cannot improve its net utility by lying about its true value. □

5.7.2 Multiple clients miss the same file

We consider the case that multiple clients require the same file. Assume that we have a set of clients $\{c_{1,1}, \cdots, c_{1,\kappa_1}, c_{2,1}, \cdots, c_{2,\kappa_2}, \cdots, c_{N,1}, \cdots, c_{N,\kappa_N}\}$, where clients $\{c_{i,1} \cdots, c_{i,\kappa_i}\}$ miss the same file p_i with the identical side information $P \backslash \{p_i\}$. We propose Alg. 11 and Alg. 12 for the cases when Z is large, and when $Z = 1$, respectively. These two algorithms are variation1 of Alg. 7 and Alg. 8. Intuitively, Alg. 11 combines the clients $\{c_{i,1}, \cdots, c_{i,\kappa_i}\}$ to be a super-client c_i and then uses the same argument of Alg 7 and Alg. 8. Each client in the same super-client has the same expected transmission cost if the corresponding super-client is scheduled. Using Theorems 65 and 66, it is straightforward to show that both algorithms are incentive-compatible.

Algorithm 11: Broker mechanism when multiple clients need the same file with large Z

input : Bids vector B, and side information H_i for all clients
output: Transmission schedules

$b_i \leftarrow \kappa_i \times \min b_{i,j}$ for all i ;
Sorting bids: $b_1 \geq b_2 \geq \cdots \geq b_N$;
Find $i^* = \max\{i : b_i \geq C(\frac{1}{i-1})\}$;
S^* includes $c_{i,j}$ for $i = 1, \cdots, i^*$ and $j = 1, \cdots, \kappa_i$;
$r_{i,j} \leftarrow \frac{1}{\kappa_i} \cdot \frac{1}{i^*-1}$ for $c_{i,j} \in S^*$;
$r_{i,j} \leftarrow 0$ for $c_{i,j} \notin S^*$;
Client $c_{i,j} \in S^*$ transmits $r_{i,j}$ proportion of the coded file $\sum_{p \in \{p_1, \cdots, p_{i^*}\} \setminus \{p_i\}} p$;

Algorithm 12: Broker mechanism when multiple clients need the same file with $Z = 1$

input : Bids vector B, and side information H_i for all clients
output: Transmission schedules

$b_i \leftarrow \kappa_i \times \min b_{i,j}$ for all i ;
Sorting bids: $b_1 \geq b_2 \geq \cdots \geq b_N$;
Find $i^* = \max\{i : b_i \geq \frac{2}{i}\}$;
S^* includes $c_{i,j}$ for $i = 1, \cdots, i^*$ and $j = 1, \cdots, \kappa_i$;
Randomly select b_{i_1} and b_{i_2} from $\{b_1, b_2, \ldots, b_{i^*}\}$;
Randomly pick one client in $\{b_{i_1,1}, b_{i_1,2}, \ldots\}$, and one client in $\{b_{i_2,1}, b_{i_2,2}, \ldots\}$ to transmit;

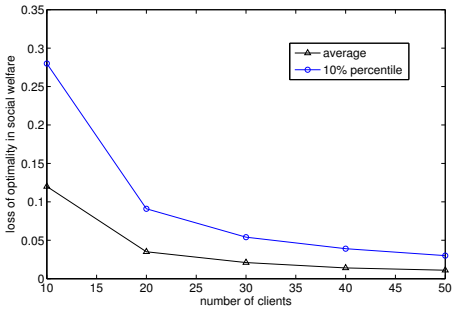

Figure 5.2: Loss of optimality in terms of social welfare when the cost function is $C(x) = x$.

Theorem 75. *Alg. 11 and Alg. 12 produce feasible* (R, S^*) *and are incentive-compatible.*

5.8 Numerical results

In this subsection, we numerically study the performance of Alg. 7 of Subsection 5.5.

Fig. 5.2 and 5.3 show the results for $C(x) = x$ and $C(x) = x^2$, respectively, where the triangle symbol represents the average result over 100000 runs, and the circle one is the 10000^{th} largest value of loss of optimality in the 100000 runs.

The experiment setting is as follows. We consider N clients (x-axle), where client c_i requires file p_i and has the side information $H_i = P \backslash \{p_i\}$. The value v_i of file p_i is uniformly distributed between 0 and 1. Let bid $b_i = v_i$ due to the incentive-compatibleness. We can observe that the proposed mechanism is close to optimal one as the number of clients gets larger. Note that if $v_N \geq C(\frac{1}{N-1})$, there will be no loss of optimality. Moreover, if we consider $C(x) = x$, then the probability $\mathbb{P}(v_N \geq \frac{1}{N-1}) = (1 - \frac{1}{N-1})^N$ approaches to 1 as $N \to \infty$. We can then conclude that Alg. 7 performs well in average case when the number of clients is large enough.

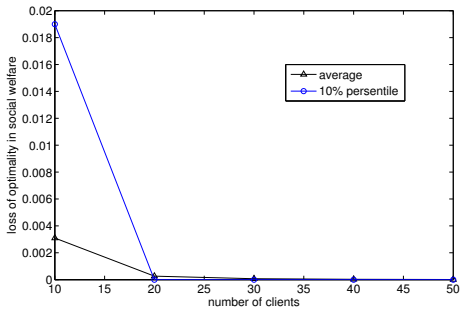

Figure 5.3: Loss of optimality in terms of social welfare when the cost function is $C(x) = x^2$.

6. JOINT CODING AND SCHEDULING FOR WIRELESS BROADCAST NETWORKS WITH SIDE INFORMATION

6.1 Introduction

The Index Coding problem is one of the basic problems in wireless network coding [1, 2]. An instance of the Index Coding problem includes a server, a set of wireless clients, and a set of packets that need to be delivered by the server to the clients. Each client is interested in a certain subset of packets and has a different subset of packets available to it as side information, e.g., through overhearing other transmissions. The server can transmit the original packets or their combinations via a noiseless broadcast channel. The goal is to find a coding scheme that requires the minimum number of transmissions to satisfy the requests of all clients.

The prior works on the Index Coding problem have focused on developing algorithms, establishing rate bounds, and analyzing the computational complexity of the problem [7, 34–38]. In contrast to the traditional Index Coding problem in which all packets are available in advance, we assume that the packets arrive to the server according to a certain random process. We also assume communications are performed over a lossy and time-varying channel.

More specifically, the server has several *data flows*, such that each data flow has it own arrival rate. Each of the clients is interested in some of the data flows and has prior side information about packets that belong to other flows. We assume that the server has an input queue for each flow that stores the packets of that flow that have not been delivered to the clients.

The key components of the system are the *coding oracle* and the *scheduler*. We will start with the *instantaneous coding*, in which the coding oracle identifies how

to encode for any set of *new* arrivals. The scheduler determines which of the coded packets will be broadcast for any given transmission opportunity. The design of the coding oracle is closely related to the solution of the Index Coding problem and it is outside of the scope of this section. Rather, our goal is to design the scheduler that will work with any given coding oracle.

In traditional wireless broadcast networks the *capacity region* describes the largest arrival rate that can be handled by networks [51]. In particular, for each rate within the capacity region there exists a scheduler that allows all packets to reach their destinations while maintaining the stability of the queues. We introduce a similar concept, referred to as a *decodable capacity region*, which captures the arrival rates that can be handled by using the network coding technique. For each rate within the decodable capacity region there exists a coding scheme and a scheduler that ensure that all destinations can decode the packets they need while maintaining the stability of the queues. Clearly, the decodable capacity region depends on the coding oracle employed by the system and our goal is to design a scheduler that achieves the maximal decodable capacity region for any given coding oracle.

We then extend our result by making the input to the coding oracle more flexible and introducing the *coding controller*, which decides which packets need to be given to the coding oracle for encoding. The joint coding and scheduling is proposed accordingly.

6.1.1 Related work

Several works, e.g., [10–15,52–54], investigate the scheduling and/or routing problems in the network coding enabled networks. Reference [10] studies the network coding in the tandem networks and formulates several related cross-layer optimization problems, while [11] devises a joint coding-scheduling-rate controller for settings

in which pairwise intersession network coding is allowed. References [13, 14, 52, 53] present scheduling algorithms for inter-session network coding. Reference [15] proposes a distributed algorithm that minimizes the transmission cost of a multicast session. The work in [54] is the most relevant to this section; however, this work focuses on the dynamic coding algorithm for a perfect channel, while we consider a lossy time-varying channel.

6.1.2 Main results

We propose the idea of decodable capacity region. To cope with the fully dynamic problem including coding and scheduling, we suggest to separate the scheduler and coding controller from the coding oracle. While coding oracle is given and fixed, our objective is to provide a framework of the scheduler and coding controller, that will work with any coding oracle. We start with the optimal scheduler design for the instantaneous coding. Then we propose the joint coding and scheduling design.

6.2 System model

For clarity, we present our results in the context of a simple broadcast topology with a single source and two destinations as depicted in Fig. 6.1.

6.2.1 Broadcast networks

We consider a network system in Fig. 6.1 that includes a server S and two wireless clients c_1, c_2. The server has a set of data flows $F = \{f_1, f_2\}$ that need to be broadcasted to clients, where each flow f_i is composed of packets $\{p_{i,1}, p_{i,2}, \cdots\}$. We assume that each packet is a symbol of some alphabet Σ. Packets that belong to f_i are stored in queue q_i at the server. Moreover, each client c_i wants flow f_i and has the side information about all the packets in $F \backslash \{f_i\}$ at beginning.

We consider the discrete time system, in which the server can transmit at most

one packet during each time slot. By $A_i(t)$ we indicate if at time t a packet $p_{i,j}$ for some j arrives at queue q_i, i.e., $A_i(t) = 1$ if a packet $p_{i,j}$ is added to q_i at time t; otherwise, $A_i(t) = 0$. Each client is associated with an ON/OFF channel. By $S(t) = (s_1(t), s_2(t))$ we describe the channel states at time t, where $s_i(t) = 1$ if the server can successfully transmit a packet to client c_i at time t; otherwise, $s_i(t) = 0$.

The packet arrivals and the channel states are assumed to be independent and identically distributed (i.i.d.) over time. By $\lambda_i = \lim_{t\to\infty} \frac{1}{\tau} \sum_{\tau=1}^{t} A_i(\tau)$, we define the arrival rate to q_i. The rate vector $\lambda = (\lambda_1, \lambda_2)$ and the probability of the ON channels are unknown to the server.

6.2.2 Coding oracle

The coding oracle is defined by the coding function $\gamma(.) : \Sigma^h \to \Sigma$, which specifies the combination for the h packets. In particular, we use a simple coding oracle as shown in Fig. 6.1:

- $\gamma(p_{i,j}) = p_{i,j}$ for $i \in 1, 2$ and some j: if only one packet $p_{i,j}$ arrives at q_i, the coding oracle specifies that an original packet $p_{i,j}$ needs to be sent.

- $\gamma(p_{1,j}, p_{2,k}) = p_{1,j} + p_{2,k}$ for some j, k: if both packets $p_{1,j}$ and $p_{2,k}$ arrives at the same time, the coding oracle specifies that the combination of $p_{1,j}$ and $p_{2,k}$ needs to be delivered over the channel.

The coding oracle determines how to encode for the *new* arrived packets. We refer to this coding scheme as the *instantaneous coding*. For example in Fig. 6.1, at time 1, packets $p_{1,1}$ and $p_{2,1}$ that arrive at the server at time 1 are forwarded into the coding oracle; therefore, a coded packet $p_{1,1} + p_{2,1}$ combined from $p_{1,1}$ in q_1 and $p_{2,1}$ in q_2 needs to be transmitted. At time 2, an original packet $p_{1,2}$ needs to be scheduled to be transmitted at present or later.

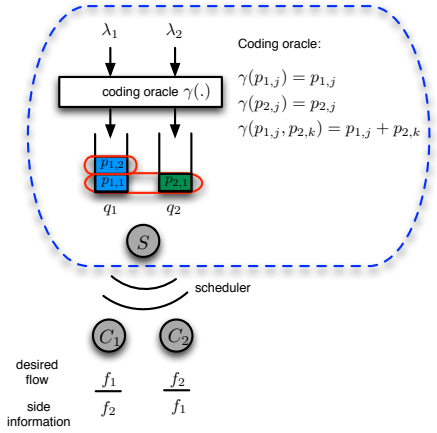

Figure 6.1: Server S delivers flows f_1, f_2 to clients c_1, c_2, associated with the coding oracle $\gamma(.)$. Packets $p_{1,1}, p_{2,1}, p_{1,2}$ arrive at the server at time $1, 1, 2$ respectively.

Note that the coding oracle only results in *how* to encode, but not combining the packets immediately and putting in another queue. In other words, via the coding oracle, the server know how these packets in the queue match to each other as shown in Fig. 6.1, and the server will combine the packets only when the associated coded packet is scheduled to be transmitted.

The coding function $\gamma(.)$ implies how to encode the packets of different flows such that these packets can be decoded by the corresponding clients; therefore, the clients can recover the desired packets by combining the received coded packets and the side information, e.g. via $\gamma(p_{1,1}, p_{2,1}) = p_{1,1} + p_{2,1}$ the server knows that $p_{1,1} + p_{2,1}$ is required to be successfully delivered to both c_1, c_2 such that the clients can decode what they want.

6.2.3 Optimal scheduler

Given a coding oracle, we are focusing on the scheduler design. For each time t, following the result of coding oracle the server schedules one coded packets $\psi(t) \in \Sigma$ to transmit, e.g., $\psi(1) = p_{1,1} + p_{2,1}$. We assume that the server is prohibited to

re-transmit the same coded packet.

The packets are removed from the queues only when their corresponding coded packets are successfully received by the clients, e.g., in Fig. 1, if $\psi(1) = p_{1,1} + p_{2,1}$ and $S(1) = (1, 0)$, then both packets $p_{1,1}$ and $p_{2,1}$ are kept in q_1 and q_2 at time 1; however, if $S(1) = (1, 1)$, then both packets are removed.

Let $Q_i(t)$ be the length of q_i at time t, and $\mu_i(t)$ be the number of packets removed from q_i at time t. Then the queueing dynamics can be described as

$$Q_i(t+1) = [Q_i(t) - \mu_i(t)]^+ + A_i(t),$$

where $[x]^+ = \max(x, 0)$, as well as $\mu_i(t)$ depends on the scheduler and channel states of the history, i.e, $\{\psi(\tau)\}_{\tau=1}^t$, and $\{S(\tau)\}_{\tau=1}^t$. We say q_i is *stable* if

$$\limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=1}^{t} \mathbb{E}[Q_i(\tau)] < \infty.$$

The *decodable capacity region* Λ is the set of all rate vectors λ for which there exists a scheduler that makes every client recover the desired packet and all queues stable. A scheduler is *optimal* if it can support all rate vectors $\lambda \in \Lambda$.

6.2.4 Remark

In Subsection 6.3, we develop the decodable capacity region and the optimal scheduler for the network system described in this subsection. Our results can be generalized as follows.

- In Subsection 6.4, re-transmitting the same coded packet is allowed.
- In Subsection 6.5, the proposed scheduler is extended by a joint coding and scheduling design.

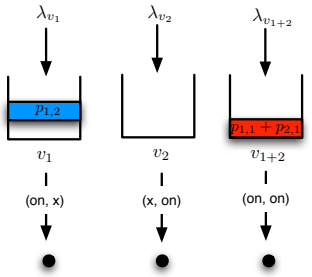

Figure 6.2: Virtual network in Subsection 6.3 including there virtual queues needed to be served, where "•" means an empty queue. The condition when the virtual channel is ON is denoted in each link, where the character "x" is a "don't care" term, e.g., $s_{v_1}(t) = 1$ when $s_1(t) = 1$, as well as $s_{v_{1+2}} = 1$ when $s_1(t) = 1$ and $s_2(t) = 1$.

Moreover, our results can be generalized to arbitrary broadcast topologies and coding oracles.

6.3 Optimal scheduler design

In this subsection, we present the optimal scheduler for the network system in Subsection 6.2. We note that there are two types of packets in q_i, for $i = 1, 2$, which includes the original packets and the packets that will be encoded. We hence simplify the problem by introducing the virtual network in Fig. 6.2. The scheduling in the virtual network are correlated to the scheduling in the real queueing network.

The virtual network in Fig. 6.2 consists of the virtual queues v_1, v_2, v_{1+2}, with the queue sizes at time t being $V_1(t)$, $V_2(t)$, $V_{1+2}(t)$, respectively. For each time, if in the real network only one packet $p_{i,j}$ for some j arrives at q_i, then $p_{i,j}$ is added to v_i of the virtual network as well. Moreover, if both packets $p_{1,j}$ and $p_{2,k}$ for some j, k arrive at q_1 and q_2 at the same time, packet $p_{1,j} + p_{2,k}$ is added to the virtual queue v_{1+2}. Let λ_{v_x}, where $x \in \{1, 2, 1+2\}$, be the virtual arrival rate to v_x. By $f(\lambda) \triangleq (\lambda_{v_1}, \lambda_{v_2}, \lambda_{v_{1+2}})$, we describe the virtual arrival rate as the function $f(.)$ of λ

in the real network. Then $f(\lambda) = (\lambda_1(1-\lambda_2), (1-\lambda_1)\lambda_2, \lambda_1\lambda_2)$.

In the virtual network, we associate each virtual queue with a virtual ON/OFF channel, and let the virtual channel states at time t be $S_v(t) = (s_{v_1}(t), s_{v_2}(t), s_{v_{1+2}}(t))$. Then $S_v(t) = (s_1(t), s_2(t), s_1(t)s_2(t))$. Moreover, in the virtual network, at most one queue can be served for each time. Let $\psi_v(t)$ be the virtual queues that is selected to be served at time t.

We are ready to make the connection between $\psi_v(t)$ and the scheduler in the real network. At time t, if in the virtual network packet $p_{i,j}$ for some j in v_i is served, i.e., $\psi_v(t) = v_i$, then in the real network the server will schedule packet $p_{i,j}$ to be transmitted over the channel and remove it from q_i. Moreover, if packet $p_{1,j} + p_{2,k}$ for some j, k in v_{1+2} of the virtual network is served, i.e, $\psi_v(t) = v_{1+2}$, then in the real network the server will deliver packet $p_{1,j} + p_{2,k}$ as well as remove both packets $p_{1,j}$ and $p_{2,k}$ from q_1 and q_2. Then we obtain $Q_1(t) = V_1(t) + V_{1+2}(t)$ and $Q_2(t) = V_2(t) + V_{1+2}(t)$. Therefore, if there exists a scheduling scheme that makes the virtual queues stable for $f(\lambda)$, the real queues are stable for λ. We conclude the general result in Proposition 76.

Proposition 76. *There exists a scheduling algorithm to stabilize the real queues for λ if and only if there exists a scheduling algorithm to stabilize the virtual queues for $f(\lambda)$. Moreover, Λ is the decodable capacity region of the real queues if and only if $f(\Lambda)$ is the capacity region of the virtual network.*

We first discuss the decodable capacity region, which implies the advantage of the coding, and then suggest the dynamic scheduling algorithm.

6.3.1 Decodable capacity region

We can characterize the capacity region of the virtual queues as follows.

$$\begin{aligned}
\lambda_{v_1} &\leq \mathbb{P}(s_1(t)=1);\\
\lambda_{v_2} &\leq \mathbb{P}(s_2(t)=1);\\
\lambda_{v_{1+2}} &\leq \mathbb{P}(s_1(t)=1 \text{ and } s_2(t)=1);\\
\lambda_{v_1}+\lambda_{v_2} &\leq \mathbb{P}(s_1(t)=1 \text{ or } s_2(t)=1);\\
\lambda_{v_1}+\lambda_{v_{1+2}} &\leq \mathbb{P}(s_1(t)=1);\\
\lambda_{v_2}+\lambda_{v_{1+2}} &\leq \mathbb{P}(s_2(t)=1);\\
\lambda_{v_1}+\lambda_{v_2}+\lambda_{v_{1+2}} &\leq \mathbb{P}(s_1(t)=1 \text{ or } s_2(t)=1).
\end{aligned}$$

By substituting $f(\lambda)=(\lambda_1(1-\lambda_2),(1-\lambda_1)\lambda_2,\lambda_1\lambda_2)$, the decodable capacity region of the real queues is calculated as follows.

$$\lambda_1 \leq \mathbb{P}(s_1(t)=1); \quad \lambda_2 \leq \mathbb{P}(s_2(t)=1).$$

Fig. 6.3 compares the decodable capacity region and the capacity region without network coding, where we can see the potential of the network coding.

6.3.2 Optimal scheduler

Let $X=\{1,2,1+2\}$ be the set of all indices of the virtual queues. We describe the queueing dynamics in the virtual network as follows.

$$V_x(t+1)=[V_x(t)-\mu_{v_x}(t)]^+ + A_{v_x}(t),$$

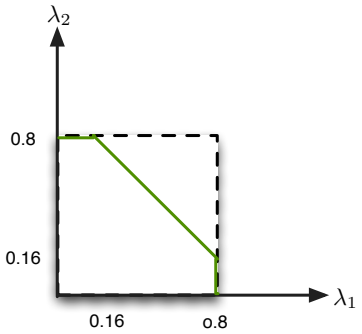

Figure 6.3: Decodable capacity region (dash line) versus capacity region without coding (solid line) when $\mathbb{P}(s_1(t) = 1) = \mathbb{P}(s_2(t) = 1) = 0.8$.

where $x \in X$, $\mu_{v_x}(t)$ is the number of packets in v_x served at time t, and $A_{v_x}(t)$ indicates if a packet arrives at v_x. In particular,

$$\mu_{v_i}(t) = \begin{cases} s_i(t) & \text{if } \psi_v(t) = v_i \\ 0 & \text{otherwise} \end{cases} \quad \text{for } i \in \{1, 2\};$$

$$\mu_{v_{1+2}} = \begin{cases} s_1(t)s_2(t) & \text{if } \psi_v(t) = v_{1+2} \\ 0 & \text{otherwise} \end{cases} .$$

To achieve the capacity region of the virtual network, we apply the MAX-Weight-type algorithm [51] to choose the optimal $\psi_v^*(t)$:

$$\psi_v^*(t) = \arg\max_{\psi_v(t)} \underbrace{\sum_{x \in X} V_x(t)\mu_{v_x}(t)}_{\text{weighted sum}} .$$

To evaluate the weighted sum in the above equation, we get

- if $\psi_v(t) = \emptyset$: the weighted sum is 0.

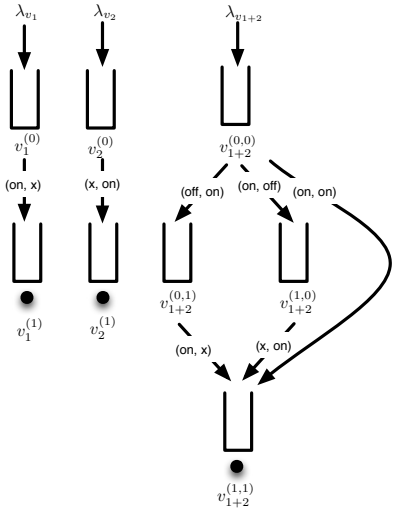

Figure 6.4: Virtual network in Subsection 6.4.

- if $\psi_v(t) = v_1$: $V_1(t)s_1(t)$.
- if $\psi_v(t) = v_2$: $V_2(t)s_2(t)$.
- if $\psi_v(t) = v_{1+2}$: $V_{1+2}(t)s_1(t)s_2(t)$.

We remark that in real network the server does not always choose a coded packet to transmit even when both channels $s_1(t)$ and $s_2(t)$ are ON; however, the decision depends on $V_i(t)$ (where $s_i(t) = 1$) and $V_{1+2}(t)$.

Putting together Proposition 76 and the throughput-optimality [51] of the MAX-Weight-type scheduling in the virtual network yields the optimal scheduler in the real network, as stated in the following theorem.

Theorem 77. *In the real network, the scheduler that follows the MAX-Weight-type scheduling in the virtual network is optimal.*

6.4 Extension: re-transmission allowed

We have proposed the optimal scheduler for the setting where re-transmitting the same coded packet is not allowed. In this subsection, the result is extended by allowing the server to make multiple transmissions of the same coded packet. To that end, we introduce another virtual network with the virtual queues as shown in Fig. 6.4. The notion of the superscripts is to indicate if in the real network the packet has been received by the corresponding clients, e.g., if in the virtual network there is a packet $p_{1,j} + p_{2,k}$ for some j, k in $v_{1+2}^{(1,0)}$, then in the real network the coded packet $p_{1,j} + p_{2,k}$ has reached c_1, but not c_2 yet.

Now we are describing the external arrivals to the virtual queues $v_1^{(0)}$, $v_2^{(0)}$, $v_{1+2}^{(0,0)}$. For each time, if in the real network only one packet $p_{i,j}$ for some j arrives at q_i, it will also be put in $v_i^{(0)}$ of the virtual network; however, if in the real network both $p_{1,j}$ and $p_{2,k}$ for some j, k come, packet $p_{1,j} + p_{2,k}$ is added to the virtual queue $v_{1+2}^{(0,0)}$.

Similar to Subsection 6.3, let $X = \{1, 2, 1+2\}$ be the set including all subscripts of the virtual queues. Moreover, by Y_x, with $x \in X$, we denote the set of all superscripts associated with the subscript x. The links in Fig. 6.4 shows the packet flows in the virtual network. We associate an ON/OFF channel to each link $v_x^m \to v_x^n$, where $x \in X$ and $m, n \in Y_x$:

- $v_1^{(0)} \to v_1^{(1)}$: the virtual channel state is $s_1(t)$.
- $v_2^{(0)} \to v_2^{(1)}$: $s_2(t)$.
- $v_{1+2}^{(0,0)} \to v_{1+2}^{(0,1)}$: $(1 - s_1(t))s_2(t)$.
- $v_{1+2}^{(0,0)} \to v_{1+2}^{(1,0)}$: $s_1(t)(1 - s_2(t))$.
- $v_{1+2}^{(0,0)} \to v_{1+2}^{(1,1)}$: $s_1(t)s_2(t)$.

- $v_{1+2}^{(0,1)} \to v_{1+2}^{(1,1)}$: $s_1(t)$.
- $v_{1+2}^{(1,0)} \to v_{1+2}^{(1,1)}$: $s_2(t)$.

Moreover, all links in the virtual network interfere to each other, i.e., for each time, at most one virtual queue can be served, which idea comes from the fact that at most one packet can be transmitted in real network. Let $\psi_v(t)$ be the virtual queue served at time t.

Let $V_x^y(t)$, with $x \in X$ and $y \in Y_x$, be the size of v_x^y at time t. The virtual queues $v_1^{(1)}, v_2^{(1)}, v_{1+2}^{(1,1)}$ are empty queues, which implies that in the real network the coded packet has been successfully delivered. Let $\mathbf{1}_x = (1, \cdots, 1) \in Y_x$,with $x \in X$, be the superscript in which all elements are one's. Now we describe the queueing dynamics in the virtual network:

$$\begin{aligned} V_x^y(t+1) = \quad & [V_x^y(t) - \sum_{m \in Y_x} \mu_{v_x^y, v_x^m}(t)]^+ + \\ & \sum_{n \in Y_x} \mu_{v_x^n, v_x^y}(t) + A_{v_x^y}, \end{aligned}$$

where $x \in X$, $y \in Y_x \backslash \mathbf{1}_x$, $A_{v_x^y}(t)$ indicates if an external packet arrives to v_x^y at time t, and $\mu_{v_x^y, v_x^m}$ is the number of packets moved from v_x^y to v_x^n. In particular,

$$\mu_{v_i^{(0)}, v_i^{(1)}} = \begin{cases} s_i(t) & \text{if } \psi_v(t) = v_i^{(0)} \\ 0 & \text{otherwise} \end{cases} \quad \text{for } i \in \{1, 2\};$$

$$\mu_{v_{1+2}^{(m,n)}, v_{1+2}^{(r,s)}} = \begin{cases} 1 & \text{if } \psi_v(t) = v_{1+2}^{(m,n)}, \\ & \text{and the link } v_{1+2}^{(m,n)} \to v_{1+2}^{(r,s)} \text{ is ON} \\ 0 & \text{otherwise} \end{cases} .$$

We define the weight $W_{v_x^m,v_x^n}(t)$ of the link $v_x^m \to v_x^n$ at time t as

$$W_{v_x^m,v_x^n}(t) = [V_x^m(t) - V_x^n(t)]^+.$$

Then we propose the Back-Pressure-type scheduling [51] to decide the optimal $\psi_v^*(t)$:

$$\psi_v^*(t) = \arg\max_{\psi_v(t)} W_{v_x^m,v_x^n} \mu_{v_x^m,v_x^n}.$$

Now we are ready to connect $\psi_v^*(t)$ in the virtual network with the scheduler in the real network. If in the virtual network $p_{i,j}$ for some j in $v_i^{(0)}$ is served, then in the real network the server will schedule packet $p_{i,j}$ to be delivered and remove it from q_i. Moreover, if in the virtual network packet $p_{1,j} + p_{2,k}$ for some j, k is delivered from $v_{1+2}^{(m,n)}$ to $v_{1+2}^{(r,s)}$, with $(r, s) \neq (1, 1)$, then the server will deliver packet $p_{1,j} + p_{2,k}$ in the real network, but both $p_{1,j}$ and $p_{2,k}$ are still kept in q_1 and q_2. When packet $p_{1,j} + p_{2,k}$ arrives at $v_{1+2}^{(1,1)}$ in the virtual network, both packets $p_{1,j}$ and $p_{2,k}$ are removed from q_1 and q_2 of the real network. Intuitively, coded packet $p_{1,j} + p_{2,k}$ will be sent in the real network until it achieves $v_{1+2}^{(1,1)}$ in the virtual network. Then we obtain $Q_i(t) = V_i^{(0)}(t) + V_{1+2}^{(0,0)}(t) + V_{1+2}^{(0,1)}(t) + V_{1+2}^{(1,0)}(t)$ for $i = 1, 2$; as such, the real queues are stable for λ if $f(\lambda)$ can be supported in the virtual network. We therefore have the following theorem because of the throughput-optimality [51] of Back-Pressure-type scheduling in the virtual network.

Theorem 78. *In the real network, the scheduler that follows the Back-Pressure-type scheduling in the virtual network is optimal.*

6.5 Extension: joint coding and scheduling design

Thus far, we was focusing on the scheduler. We will present the joint coding and scheduling in this subsection.

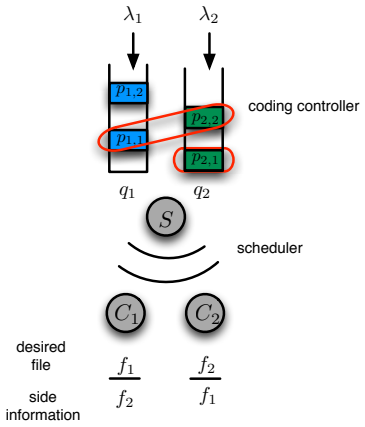

Figure 6.5: Include coding controller to the system. Packets $p_{2,1}$, $p_{1,1}$, $p_{2,2}$, $p_{1,2}$ arrive at time $1, 2, 3, 4$, respectively.

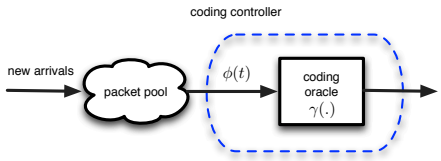

Figure 6.6: Coding controller versus coding oracle.

Instead of giving all new arrivals to the coding oracle, the system includes the *coding controller*, which for each time t determines a set $\phi(t)$ of packets in the queue that will be forwarded to the coding oracle. In contrast to instantaneous coding, we refer this coding system as the *dynamic coding*. In Fig. 6.5, if the server uses the instantaneous coding as before, original packets $p_{1,1}, p_{1,2}, p_{2,1}, p_{2,2}$ need to be scheduled to be transmitted; however, using dynamic coding the server is allowed to apply the coding oracle to packets over different time. For example in Fig. 6.5, $\phi(1) = \{p_{2,1}\}$, $\phi(2) = \emptyset$, $\phi(3) = \{p_{1,1}, p_{2,2}\}$, and $\phi(4) = \emptyset$. Note that $\phi(2) = \emptyset$ and $\phi(3) = \{p_{1,1}, p_{2,2}\}$ means that there is no coding decision at time 2, and the coding decision for $p_{1,1}$ is delayed to time 3.

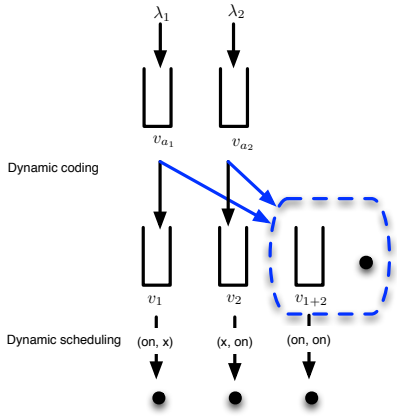

Figure 6.7: Virtual network in Subsection 6.5, where "•" means an empty queue. The flow $\{v_{a_1}, v_{a_2}\} \rightarrow \{v_{1+2}, \emptyset\}$ is depicted in blue color.

Fig. 6.6 illustrates the notion of the coding controller, where all new arrivals are put in the packet pool. For each time t, the coding controller chooses the packets $\phi(t)$ from the packet pool and outputs the coding decision by passing the packets $\phi(t)$ into the coding oracle $\gamma(.)$. In this section, the coding decisions that have been decided for $\phi(t)$ cannot be changed later, i.e., $\phi(t)$ needs to be removed from the packet pool. Moreover, without loss of generality, we assume that $\phi(t)$ contains at most one packets for each data flows.

We focus on the scenario where the re-transmission is not allowed. We propose the virtual network in Fig. 6.7, where the lower part is for the scheduler similar to Subsection 6.3. In this virtual network, we introduce more virtual queues v_{a_i}, with $i \in \{1, 2\}$, that store the virtual arrivals, i.e, when packet $p_{i,j}$ for some j is added to q_i, it is added to v_{a_i} as well.

In this virtual network, the additional links are introduced, i.e., $v_{a_1} \rightarrow v_1$, $v_{a_2} \rightarrow v_2$, and $\{v_{a_1}, v_{a_2}\} \rightarrow \{v_{1+2}, \emptyset\}$, where the last one means that one of packets in v_{a_1} and v_{a_2} will go to v_{1+2} and the other one will be just removed from its virtual queue

if both v_{a_1} and v_{a_2} are not empty.

Let $\Phi = \{\emptyset, 1, 2, 1+2\}$ be the set including all the actions of the coding controller, i.e., do nothing, original packet $p_{i,j}$ for some i, j, and the combination of packets $p_{1,j}$ and $p_{2,k}$ for some j, k. Let $\phi_v(t) \in \Phi$ be the coding action selected in the virtual network at time t. In additional to $\psi_v(t)$ in Subsection 6.3, $\phi_v(t)$ needs to be determined for each time t. If $\phi_v(t) = i$, with $i \in \{1, 2\}$, then a packet $p_{i,j}$ for some j in v_{a_i} will be delivered to v_i. If $\phi_v(t) = \{1 + 2\}$, one of $p_{1,j}$ and $p_{2,k}$ for some j, k will be sent to v_{1+2} and be stored in another type $p_{1,j} + p_{2,k}$ of the packet, while the other is sent to the empty queue.

In this virtual network, the upper part is used for the coding controller in the real network while the lower part is for the scheduler in the real network. At time t, if in the virtual network a packet $p_{i,j}$ for some j is sent from v_{a_i} to v_x with $x \in \{1, 2\}$, then the server in the real network decides an original packet $\phi(t) = p_{i,j}$. If in the virtual network, two packets $p_{1,j}$, $p_{2,k}$ for some j, k are delivered to v_{1+2} and empty queue, then in the real network the server selects $\phi(t) = \{p_{1,j}, p_{2,k}\}$ for encoding.

Since the Back-Pressure-type scheduling in the virtual network is throughput-optimal, the corresponding coding controller and scheduler in the real network are optimal, as stated in the following.

Theorem 79. *In the real network, the coding controller $\phi(t)$ and scheduler $\psi(t)$ that follow $\phi_v(t)$ and $\psi_v(t)$ of Back-Pressure-type scheduling in the virtual network are optimal.*

6.6 Numerical results

In this subsection, we numerically study the performance of the proposed jointly coding and scheduling algorithm. The average system backlog is defined as the summation of each time-averaged queue length. In Fig. 6.8 and 6.9, we show the

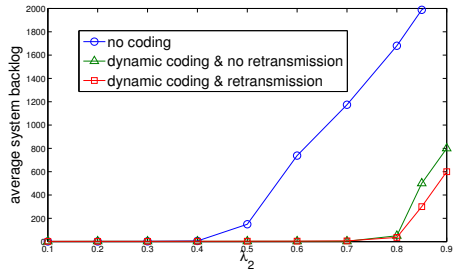

Figure 6.8: Average system backlog versus λ_2 when $\lambda_1 = 0.5$.

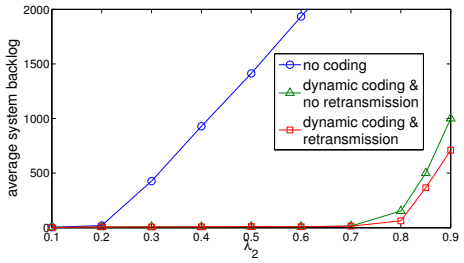

Figure 6.9: Average system backlog versus λ_2 when $\lambda_1 = 0.75$.

average system backlog versus λ_2 when $\lambda_1 = 0.5$ and 0.75 respectively. Moreover, $\mathbb{P}(s_1(t) = 1) = \mathbb{P}(s_2(t) = 1) = 0.8$. The $-\bigcirc-$ line represents applying Max-Weight-type scheduling algorithm to the real network when the server does not perform any coding operation. The $-\triangle-$ and $-\square-$ lines stand for using the proposed dynamic coding and scheduling under no retransmission policy and retransmission allowance policy respectively.

6.7 Dynamic coding and scheduling for general coding oracle

In Subsection 6.5, we propose the jointly design of the coding controller and scheduling controller for the simple topology. In this subsection, we discuss how to extend the results to the more general settings, where there can be more than two clients and any coding oracle.

We will focus on the case when the retransmission is allowed. We build up the virtual network as follows.

- Create the virtual queues v_{a_i} for $i \in \{1, \cdots, N\}$ that stores the arrivals.
- For each coding function, say $(g_1, \cdots, g_m) = \gamma(f_1, \cdots, f_n)$ with m coded packets and n arrival packets, create the m virtual queues $v_{g_i}^{(y_1, \cdots, y_n)}$, where $i \in \{1, \cdots, m\}$ indicates how to encode for $f_1, \cdots, f_n$, and $(y_1, \cdots, y_n)$ includes the indicators that implies if client c_i, with $i \in \{1, \cdots, n\}$, has received the coded packet.
- The links in the upper part of the virtual network are connected according to the coding oracle. For each coding function, say $(g_1, \cdots, g_m) = \gamma(f_1, \cdots, f_n)$, create the flows $\{v_{a_1}, \cdots, v_{a_n}\} \to \{v_{g_1}^{(0, \cdots, 0)}, \cdots, v_{g_m}^{(0, \cdots, 0)}, \emptyset\}$.
- The links in the lower part of the virtual network are connected according to the channel condition. For each $v_{g_i}^{(y_1, \cdots, y_n)}$, create the links $v_{g_i}^{(y_1, \cdots, y_n)} \to v_{g_i}^{(w_1, \cdots, w_n)}$ with $w_j > y_j$ for some j's. The link $v_{g_i}^{(y_1, \cdots, y_n)} \to v_{g_i}^{(w_1, \cdots, w_n)}$ is ON if for all $1 \leq j \leq n$: (1) $s_j = 1$ when $w_j = 1$ and $y_j = 0$ and (2) $s_j = 0$ when $w_j = 0$ and $y_j = 0$.
- Each time one coding action is selected for the upper part.
- Each time only one queue in the lower part can be served.

The Back-Pressure-type algorithm is applied to the virtual network. According to the result of the Back-Pressure in the virtual network, the upper part of the virtual network implies how and when to assign the coding rule in the real network, while the lower one indicates which coded packet to transmit at present. Therefore, we can conclude as follows for the more general setting.

Theorem 80. *For any broadcast network associated any coding oracle, the controller that follows the Back-Pressure-type scheduling in the upper and lower parts of the virtual network is optimal.*

7. CONCLUSION

We have identified several cutting-edge issues and challenges at the intersections of network coding, control theory, and game theory. By developing general methodologies, we offer new insights into network coding, controller design, and incentive design. In general, these problems would require interdisciplinary components, e.g., network coding theory, control theory, game theory, network optimization, queueing theory, complexity theory, approximation algorithm, and distributed computing.

7.1 Extension of opportunistic network coding problem

The theoretical results in this dissertation are focused on the three-hops network. How to extend the results to a general network with noisy channels is critical. In general, Markov decision process leads to a notorious "curse of dimensionality" problem, and therefore is hard to be analyzed when it contains many random variables. The Lyapunov theory plays an important role to investigate multi-hops networks. Using the Lyapunov techniques, we expect to design an on-line algorithm and build up the performance bound. Moreover, Markov decision process and Lyapunov theory focus on the average behavior. How to interplay between Markov decision process, Lyapunov theory, and competitive analysis (worst-case analysis) would be challenging.

7.2 Extension of network coding associated with game and control theory

We consider the selfishness and network dynamics separately in this dissertation. The following question is still open: how to design an incentive mechanism that works with any coding oracle and controller? To that end, we first have to answer the feasibility question: what properties should a coding oracle and controller have such that there exists an incentive mechanism that is tractable? Then, we can focus

on the incentive mechanism design for these feasible coding oracles and controllers.

7.3 Future work

Our future plans will involve continuing the research discussed above. In parallel, we note interesting topics at network robustness in power networks, since a large-scale network leads to an increasing concern about the robustness of protocols. The ultimate goal of our research is to address the problems posed in the design and analysis of communication systems and networks. We believe that our work will benefit both theoretical and practical developments.

7.3.1 Network robustness

Cascading failures in a smaller sub-system might be lead to a catastrophic failure in a large-scale network. Therefore, how to design and analyze a robust protocol against an unexpected event is significant. Markov decision process, Lyapunov techniques, and large deviation theory can be applied to evaluate the probabilities of an unexpected event in a stochastic system; meanwhile, a mitigation scheme can be developed to improve the system performance. Without knowing the distribution of an uncertainty, reachability analysis in control theory can be used for a worst-case analysis, i.e., to compute the set that encloses all possible trajectories of a system under an uncertainty. An application of this tool is to verify if a system can exclude the undesired events, and ensure that the desired performance is met.

7.3.2 Set theory in a large-scale power network

The traditional reachability analysis is developed for a small system. We suggest extending the concept of reachability analysis to a large-scale power network by combining techniques from matrix decomposition and reachable set concept. Once a system can be described in a matrix, we have candidate tools such as LU, SVD, and

Schur decomposition. We expect to incorporate the decomposition and reachability analysis to reduce the computational complexity in a large-scale power network.

REFERENCES

[1] Y. Birk and T. Kol, "Coding on Demand by an Informed Source (ISCOD) for Efficient Broadcast of Different Supplemental Data to Caching Clients," *IEEE Trans. Inf. Theory*, vol. 52, pp. 2825–2830, 2006.

[2] Z. Bar-Yossef, Y. Birk, T. S. Jayram, and T. Kol, "Index Coding with Side Information," *IEEE Trans. Inf. Theory*, vol. 57, pp. 1947–1494, 2011.

[3] A. Sprintson, P. Sadeghi, G. Booker, and S. E. Rouayheb, "Deterministic Algorithm for Coded Cooperative Data Exchange," *Quality, Reliability, Security and Robustness in Heterogeneous Networks* Springer, pp. 282–289, 2012.

[4] A. Sprintson, P. Sadeghi, G. Booker, and S. E. Rouayheb, "Randomized Algorithm and Performance Bounds for Coded Cooperative Data Exchange," *in Proc. of IEEE International Symposium on. Information Theory (ISIT)*, pp. 1888–1892, 2010.

[5] D. Ozgul and A. Sprintson, "An Algorithm for Cooperative Data Exchange with Cost Criterion," *in Proc. of Information Theory and Applications Workshop (ITA)*, pp. 1–4, 2011.

[6] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network Information Flow," *IEEE Trans. Inf. Theory*, vol. 46, pp. 1204–1216, 2000.

[7] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "XORs in the Air: Practical Wireless Network Coding," *IEEE/ACM Trans. Netw.*, vol. 16, pp. 497–510, 2008.

[8] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading Structure for Randomness in Wireless Opportunistic Routing," *in Proc. of ACM Special In-*

terest Group on Data Communication (SIGCOMM), pp. 169–180, 2007.

[9] M. Effros, T. Ho, and S. Kim, "A Tiling Approach to Network Code Design for Wireless Networks," *in Proc of IEEE Information Theory Workshop (ITW)*, pp. 62–66, 2006.

[10] Y. E. Sagduyu and A. Ephremides, "Cross-Layer Optimization of MAC and Network Coding in Wireless Queueing Tandem Networks," *IEEE Trans. Inf. Theory*, vol. 54, pp. 554–571, 2008.

[11] A. Khreishah, C. C. Wang, and N. Shroff, "Cross-Layer Optimization for Wireless Multihop Networks with Pairwise Intersession Network Coding," *IEEE J. Sel. Areas Commun.*, vol. 27, pp. 606–621, 2009.

[12] V. Reddy, S. Shakkottai, A. Sprintson, and N. Gautam, "Multipath Wireless Network Coding: A Population Game Perspective," *in Proc of IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1–9, 2010.

[13] A. Eryilmaz, D. S. Lun, and B. T. Swapna, "Control of Multi-Hop Communication Networks for Inter-Session Network Coding," *IEEE Trans. Inf. Theory*, vol. 57, pp. 1092–1110, 2011.

[14] T. Ho and H. Viswanathan, "Dynamic Algorithms for Multicast with Intra-Session Network Coding," *IEEE Trans. Inf. Theory*, vol. 55, pp. 797–815, 2009.

[15] Y. Xi and E. M. Yeh, "Distributed Algorithms for Minimum Cost Multicast with Network Coding," *IEEE/ACM Trans. Netw.*, vol. 18, pp. 379–392, 2010.

[16] E. C. Ciftcioglu, Y. E. Sagduyu, R. A. Berry, and A. Yener, "Cost-Delay Tradeoffs for Two-Way Relay Networks," *IEEE Trans. Wireless Commun.*, vol. 10, pp. 4100–4109, 2011.

[17] D. N. T. Nguyen and X. Yang, "Multimedia Wireless Transmission with Network Coding," *in Proc. of IEEE Packet Video*, pp. 326–335, 2007.

[18] Y. Huang, M. Ghaderi, D. Towsley, and W. Gong, "TCP Performance in Coded Wireless Mesh Networks," *in Proc. of IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 179–187, 2008.

[19] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: Wiley-Interscience, 1994.

[20] S. M. Ross, *Introduction to Stochastic Dynamic Programming*. New York, NY, USA: Academic Press, 1994.

[21] L. I. Sennott, *Stochastic Dynamic Programming and the Control of Queueing Systems*. New York, NY, USA: Wiley-Interscience, 1998.

[22] D. P. Bertsekas, *Dynamic Programming: Deterministic and Stochastic Models*. Upper Saddle River, NJ, USA: Prentice Hall, 1987.

[23] V. S. Borkar, "Control of Markov Chains with Long-Run Average Cost Criterion: The Dynamic Programming Equations," *SIAM Journal on Control and Optimization*, vol. 10, pp. 642–657, 1989.

[24] R. Cavazos-Cadena and L. I. Sennott, "Comparing Recent Assumptions for the Existence of Average Optimal Stationary Policies," *Operations Research Letters*, vol. 11, pp. 33–37, 1992.

[25] M. Schal, "Average Optimality in Dynamic Programming with General State Space," *Mathematics of Operations Research*, vol. 18, pp. 163–172, 1993.

[26] L. I. Sennott, "Average Cost Optimal Stationary Policies in Infinite State Markov Decision Processes with Unbounded Costs," *Operations Research*, vol. 37, pp. 626–633, 1989.

[27] G. Koole, *Monotonicity in Markov Reward and Decision Chains: Theory and Applications*. Hanover, MA, USA: Now Publishers Inc, 2007.

[28] S. I. Resnick, *Adventures in Stochastic Processes*. Boston, MA, USA: Birkhauser, 1992.

[29] V. S. Borkar, "Convex Analytic Methods in Markov Decision Processes," *Handbook of Markov Decision Processes* Springer, pp. 347–375, 2002.

[30] S. Meyn, *Control Techniques for Complex Networks*. New York, NY, USA: Cambridge University Press, 2007.

[31] W. N. Kang, F. P. Kelly, N. H. Lee, and R. J. Williams, "Product Form Stationary Distributions for Diffusion Approximations to a Flow-Level Model Operating under a Proportional Fair Sharing Policy," *ACM SIGMETRICS Performance Evaluation Review*, vol. 35, pp. 36–38, 2007.

[32] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*. New York, NY, USA: Cambridge University Press, 1998.

[33] N. Buchbinder and J. Naor, *The Design of Competitive Online Algorithms via a Primal-Dual Approach*. Hanover, MA, USA: Now Publishers Inc, 2009.

[34] A. Blasiak, R. Kleinberg, and E. Lubetzky, "Index coding via linear programming." [Online]. Available: http://arxiv.org/abs/1004.1379

[35] N. Alon, E. Lubetzky, U. Stav, A. Weinstein, and A. Hassidim, "Broadcasting with Side Information," *in Prof. of IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 823–832, 2008.

[36] S. E. Rouayheb, A. Sprintson, and C. Georghiades, "On the Index Coding Problem and Its Relation to Network Coding and Matroid Theory," *IEEE Trans. Inf. Theory*, vol. 56, pp. 3187–3195, 2010.

[37] M. A. R. Chaudhry and A. Sprintson, "Efficient Algorithms for Index Cdoing," *in Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1–4, 2008.

[38] M. Langberg and A. Sprintson, "On the Hardness of Approximating the Network Coding Capacity," *IEEE Trans. Inf. Theory*, vol. 57, pp. 1008–1014, 2011.

[39] M. A. R. Chaudhry, Z. Asad, A. Sprintson, and M. Langberg, "On the complementary Index Coding problem," *in Proc. of IEEE International Symposium on Information Theory (ISIT)*, pp. 244–248, 2011.

[40] M. A. R. Chaudhry, Z. Asad, A. Sprintson, and M. Langberg, "Finding Sparse Solutions for the Index Coding Problem," *in Proc. of IEEE Global Communications Conference (GLOBECOM)*, pp. 1–5, 2011.

[41] W. Vickrey, "Counterspeculation, Auctions, and Competitive Sealed Tenders," *The Journal of Finance*, vol. 16, pp. 8–37, 1961.

[42] E. Clarke, "Multipart Pricing of Public Goods," *Public Choice*, vol. 11, pp. 17–33, 1971.

[43] T. Groves, "Incentives in Teams," *Econometrica*, vol. 41, pp. 617–631, 1973.

[44] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*. New York, NY, USA: Cambridge University Press, 2007.

[45] A. Mualem and N. Nisan, "Truthful Approximation Mechanisms for Restricted Combinatorial Auctions," *Games and Economic Behavior*, vol. 64, pp. 612–631, 2008.

[46] J. Hastad, "Clique is Hard to Approximate within $n^{1-\epsilon}$," *Acta Mathematica*, vol. 182, pp. 105–142, 1999.

[47] H. Y. Hsieh and R. Sivakumar, "On Using Peer-to-Peer Communication in Cellular Wireless Data Networks," *IEEE Trans. Mobile Comput.*, vol. 3, pp. 57–72, 2004.

[48] M. J. Neely and L. Golubchik, "Utility Optimization for Dynamic Peer-to-Peer Networks with Tit-for-Tat Constraints," *in Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1458–1466, 2011.

[49] J. Zhao, P. Zhang, and G. Cao, "On Cooperative Caching in Wireless P2P Networks," *in Proc. of IEEE International Conference on Distributed Computing Systems (ICDCS)*, pp. 731–739, 2008.

[50] M. P. Rinne, C. S. Wijting, C. B. Ribeiro, and K. Hugl, "Device-to-Device Communication as an Underlay to LTE-Advanced Networks," *IEEE Commun. Mag.*, vol. 47, pp. 42–49.

[51] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems.* San Rafael, CA, USA: Morgan & Claypool Publishers, 2010.

[52] N. M. Jones, B. Shrader, and E. Modiano, "Optimal Routing and Scheduling for a Simple Network Coding Scheme," *in Proc of IEEE International Conference on Computer Communications (INFOCOM)*, pp. 352–360, 2012.

[53] N. M. Jones, B. Shrader, and E. Modiano, "Distributed CSMA with Pairwise Coding," *in Proc of IEEE International Conference on Computer Communications (INFOCOM)*, pp. 2094–2012, 2013.

[54] M. J. Neely, A. S. Tehrani, and Z. Zhang, "Dynamic Index Coding for Wireless Broadcast Networks," *IEEE Trans. Inf. Theory*, vol. 59, pp. 7525 – 7540, 2013.