DELAY-AWARE SCHEDULING IN WIRELESS CODING NETWORKS: TO WAIT OR NOT TO WAIT

A Thesis

by

SOLAIRAJA RAMASAMY

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

MASTER OF SCIENCE

December 2010

Major Subject: Computer Engineering

DELAY-AWARE SCHEDULING IN WIRELESS CODING NETWORKS: TO WAIT OR NOT TO WAIT

A Thesis

by

SOLAIRAJA RAMASAMY

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

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Alex Sprintson
Committee Members,	A.L.Narasimha Reddy
	Radu Stoleru
Head of Department,	Costas N. Georghiades

December 2010

Major Subject: Computer Engineering

ABSTRACT

Delay-aware Scheduling in Wireless Coding Networks:

To Wait or Not to Wait. (December 2010)

Solairaja Ramasamy, B.Tech., College of Engineering, Guindy,

Anna University, India

Chair of Advisory Committee: Dr.Alex Sprintson

Wireless technology has become an increasingly popular way to gain network access. Wireless networks are expected to provide efficient and reliable service and support a broad range of emerging applications, such as multimedia streaming and video conferencing. However, limited wireless spectrum together with interference and fading pose significant challenges for network designers. 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 wireless medium.

Reverse carpooling is one of the main techniques used to realize the benefits of network coding in wireless networks. With reverse carpooling, two flows are traveling in opposite directions, sharing a common path. The network coding is performed in the intermediate (relay) nodes, which saves up to 50% of transmissions.

In this thesis, we focus on the scheduling at the relay nodes in wireless networks with reverse carpooling. When two packets traveling in opposite directions are available at the relay node, the relay node combines them and broadcasts the resulting packet. This event is referred to as a *coding opportunity.* When only one packet is available, the relay node needs to decide whether to wait for future coding opportunities, or to transmit them without coding. Though the choice of holding packets exploits the positive aspects of network coding, without a proper policy in place that controls how long the packets should wait, it will have an adverse impact on delays and thus the overall network performance. Accordingly, our goal is to find an optimal control strategy that delicately balances the tradeoff between the number of transmissions and delays incurred by the packets.

We also address the fundamental question of what local information we should keep track of and use in making the decision of *of whether to transmit uncoded packet or wait for the next coding opportunity.* The available information consists of queue length and time stamps indicating the arrival time of packets in the queue. We could also store history of all previous states and actions. However, using all this information makes the control very complex and so we try to find if the overhead in collecting waiting times and historical information is worth it.

A major contribution of this thesis is a stochastic control framework that uses state information based on what can be observed and prescribes an optimal action. For that, we formulate and solve a stochastic dynamic program with the objective of minimizing the long run average cost per unit time incurred due to transmissions and delays. Subsequently, we show that a stationary policy based on queue lengths is optimal, and the optimal policy is of threshold-type. Then, we describe a non-linear optimization procedure to obtain the optimal thresholds.

Further, we substantiate our analytical findings by performing numerical experiments under varied settings. We compare systems that use only queue length with those where more information is available, and we show that optimal control that uses only the queue length is as good as any optimal control that relies on knowing the entire history.

To My parents, teachers and God

ACKNOWLEDGMENTS

First and foremost, I express my deepest gratitude to my advisor, Dr.Alex Sprintson, for his invaluable guidance and assistance throughout my masters programme. I thank him for the confidence he had in me and for giving me the opportunity to work on this research project. I also sincerely thank Dr.Gautam Natarajan and Dr.Srinivas Shakkottai who along with Dr.Alex Sprintson initiated this project and also actively participated throughout. Their feedback and opinions were of immense help. It is their expertise and hard work that helped us crossing all barriers and taking this work up to this point. I also thank Navid Abedini, my friend and a Ph.D student with Dr.Srinivas Shakkottai, for all his hard work and contributions to this work particularly our Infocomm submission.

I would also like to extend my gratitude to Dr.Narasimha Reddy and Dr.Radu Stoleru for readily agreeing to be on my committee. It is really an honor to have such esteemed professors on my committee.

I also thank Vinith Reddy, a former M.S student, in our group for mentoring me during the early stages of research. The valuable knowledge he shared about OPNET and his research experience helped me a lot in the course of time. I also thank my friends Somasundaram Esakkiappan, Karthik Raviprakash and Ramprasad Diwakaran who made my transition back into student life easier and for making the stay in College Station a pleasant experience.

Finally, I thank my closest friend, Pragatheeswaran Angu, my parents and the rest of my family members for their love and support in all my endeavors. *All praise to God!!*

This material is based upon work supported by the National Science Foundation under Grant No. 0954153

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

Wireless networks are becoming increasingly popular in recent years. With the advent of cheaper smart phones and other hand held devices, wireless networks have become ubiquitous. In order to provide support to the wide range of requirements arising due to emerging applications such as multimedia streaming and video conferencing, significant amount of work has been done recently focusing on reducing the negative impacts due to interference, fading and energy constraints in wireless networks.

The novel technique of Network coding introduced by Ahlswede *et al.* in their seminal work [1] has been proved to significantly improve the throughput benefits and energy efficiency in wireless networks. In contrast to the traditional routing techniques where packets from different flows are treated as distinct entities, network coding enables cooperation among different network flows and allows intermediate forwarding nodes to mix packets from multiple flows and transmit them together as a single packet. In wireless networks, network coding can exploit the broadcast nature of the medium and offers great benefits in terms of reduced number of transmissions and thus more effective bandwidth and energy usage.

For example, consider a wireless network coding scheme depicted in Figure 1(a). Here, wireless nodes 1 and 2 need to exchange packets x_1 and x_2. It is also assumed that the end nodes 1 and 2 cannot communicate directly due to power constraints. Hence all communication has to be relayed through node 3 which lies in between. In this scenario, the traditional simple *store-and-forward* approach needs four transmissions totally. However, the network coding solution uses a *store-code-and-forward*

The journal model is *IEEE Transactions on Automatic Control.*

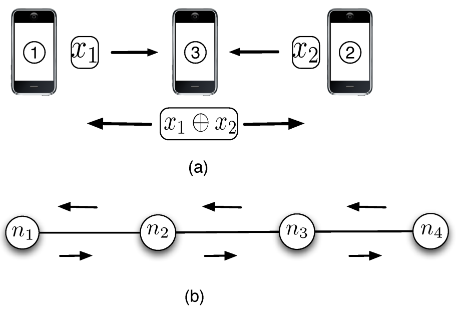

Fig. 1. Reverse carpooling in wireless coding network

approach in which the two packets x_1 and x_2 are combined by means of a bitwise XOR operation at the relay and broadcast to nodes 1 and 2 simultaneously. Nodes 1 and 2 can then decode this coded packet to obtain the packets they are interested in. In larger networks, the gain due to network coding will be even higher.

In order to better realize the benefits of network coding, several techniques like coding aware routing (e.g., DCAR [2], [3]) have been proposed. Effros *et al.* [4] introduced the strategy of *reverse carpooling* that allows choosing routes intelligently such that information flows traveling in opposite directions share a common path. Figure 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 intermediate nodes n_2 and n_3 perform coding and this coding approach results in a significant reduction (up to 50%) in the number of transmissions for the 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.

A. Motivation and Goals

In the simple reverse carpooling based network coding approach discussed previously, we refer to the scenario where intermediate nodes combine packets from different flows as *opportunistic coding*. We call it opportunistic as an intermediate node performs coding only when it finds packets from all compatible flows during the period when it is scheduled to transmit. If there is no opportunity for coding i.e. if packets are not available from one of the flows, then the intermediate nodes simply forward the packets without any coding. Here, by missing on future potential coding opportunities the network fails to fully realize the benefits of network coding. An alternative which is worth considering is to wait for a future opportunity and perform coding. Though this approach exploits the positive aspects of network coding and results in reduced number of transmissions, the additional delay introduced may have severe impact on overall performance of the system. Hence there is a need for establishing control mechanisms to delicately manage the trade-off between the number of transmissions and delay in the network. 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. Thus, the network must schedule packets considering both delays as well as coding gains.

We look into the design of distributed controllers to schedule packets at intermediate nodes. We also need to find appropriate strategies to be used in deciding the action to be taken when there is no coding opportunity. Essentially, the controller has to decide whether to transmit without coding or wait for future coding opportunities. Since both transmissions and holding packets involve costs, the objective is to design a controller that minimizes the average cost incurred due to transmissions and delays. In the process, we also look into the question of what local information need to be collected at each intermediate node that will be used by the controllers in making decisions.

B. Basic Model

Consider a relay node that transmits packets between two of its adjacent nodes that has flows in opposite directions, as depicted in Figure 2. We call the broadcast link at relay node R, *hyperlink*. 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 nodes 2 and 1, respectively. Every time when the relay node gets an opportunity to transmit, if both queues are not empty then it can transmit two packets (one from each queue) simultaneously by performing an XOR operation. However, what should the hyperlink do if one of the queues has packets to transmit, while the other queue is empty? Should the relay wait for a coding opportunity in future or just transmit a packet from a non-empty queue without coding? This fundamental question is the main motivation for this work.

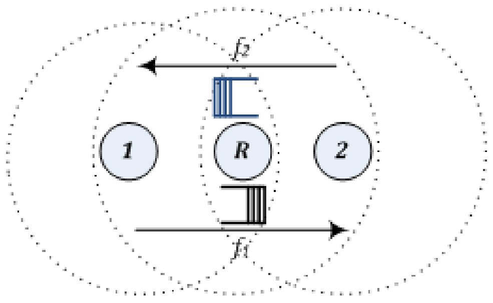

Fig. 2. 3-Node relay network

The controllers at intermediate nodes use policies that yield a *transmit* or *do not transmit* decision at each time instant. The optimal policy is defined as the one that minimizes long-run average system cost which includes costs of transmissions plus holding costs. 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 positive externality of network coding, hence are not energy-efficient. However, by waiting for packets to code, we might be able to achieve energy efficiency at the cost of packets getting delayed further.

C. Related Work

Network coding research was initiated by seminal work by Ahlswede *et al.* [1] and since then attracted major interest from the research community. Many initial works on the network coding technique focused on establishing *multicast* connections between a fixed source and a set of terminal nodes. Li *et al.* [5] showed that the maximum rate of a multicast connection is equal to the minimum capacity of a cut that separates the source and any terminal. In a subsequent work, Koetter and Médard [6] developed an algebraic framework for network coding and investigated linear network codes for directed graphs with cycles.

Network coding technique for wireless networks has been considered by Katabi *et al.* [7]. They propose an architecture called COPE, which contains a special network coding layer between the IP and MAC layers. In [8] Chachulski *et al.* proposed an opportunistic routing protocol, referred to as MORE, that randomly mixes packets that belong to the same flow before forwarding them to the next hop. Sagduyu and Ephremides [9] focused on the applications of network coding in simple path topologies (referred to in [9] as *tandem* networks) and formulated related cross-layer optimization problems. Similarly, [10] considered the problem of utility maximization when network coding is possible. However, their focus is on opportunistic coding as opposed to *creating* coding opportunities that we focus on. In the work by Reddy *et al.* [11], it was showed how to design coding-aware routing controllers that would maximize coding opportunities (and hence reduce the number of transmissions) in multihop networks. However, in contrast to all the above literature our objective here is to study the delicate tradeoff between transmission costs and waiting costs when network coding is an option.

For that we formulate and solve a stochastic dynamic program, in particular a Markov decision process (MDP), to determine the optimal control actions in various states. However, although there have been several excellent books (Puterman [12], Ross [13] and Bertsekas [14] to name a few) on MDPs, there are relatively few articles that provide a methodology to find optimal policies for problems like ours that are infinite horizon, average cost optimization with a countably infinite state space. In fact, Bertsekas [14] specifically says that such problems are difficult to analyze and obtain optimal policies. But the literature is extremely rich for discounted cost infinite horizon problems and average cost finite state-space problems.

D. Thesis Organization

This thesis is organized as follows. In the Chapter II, we formally introduce our system model with appropriate notations and definitions. Then in the Chapter III we introduce the space of controllers where we need to look for and find the optimal control. Next we look into the possible information that can be collected locally and based on which the strategies for the controller can be devised. Having listed down the available information, we then introduce different strategies that can be used by the controllers and discuss their analysis using Discrete Markov Chain model. As the amount of local information that need to be tracked and maintained determines the complexity of controllers, in the beginning of Chapter IV, we establish what information *queue length, vector of waiting times for all packets etc.*, is actually required and then we find the structure of optimal policy. Once the structure of optimal policy is found, we show how the exact policy will look like in the following sections. In the Chapter V, we discuss the experiments that are run to numerically validate our analytical findings. Finally, we conclude in the Chapter VI by giving the summary of our research work and possible future extensions to it.

CHAPTER II

SYSTEM MODEL

Consider a multi-hop wireless network operating a time-division multiplexing scheme to store and forward packets from various sources to destinations. Time is slotted into small intervals and in each interval every node gets to transmit at most one packet of a flow. This packet is transmitted during a “mini-slot” that the node has been assigned. We assume that this mini-slot is instantaneous for all practical purposes. Also, in this model we will not consider any scheduling issues and assume that we have scheduled mini-slots assigned to each node for each flow where nodes have opportunities to transmit if they choose to. With that said, we will now describe the scenario from the perspective of a single node, especially a hyperlink that has the potential for network coding packets from flows in opposing directions.

A. Scenario from a Hyperlink's Perspective

Consider the network on Figure 3. We call two of the adjacent nodes to the hyperlink R as nodes 1 and 2. Say 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 hyperlink under consideration. The packets from both flows go through separate queues, q_1 and q_2, at node R. With respect to the hyperlink we now define a *slot* as the time between successive opportunities for the hyperlink to transmit. The packet arrivals from node i follow Bernoulli distribution with rate p_i i.e. in each slot a packet arrives from node i (during its transmission opportunity) to q_i with probability p_i for $i = 1, 2$ and with probability $(1 - p_i)$ no packet arrives from node i in a slot. Thus, a maximum of 1 packet arrives from each adjacent node to the hyperlink during a slot (this is according to the network definition and scheduling we described earlier). At the end of a slot, the hyperlink gets an opportunity to transmit and it can transmit a maximum of one packet.

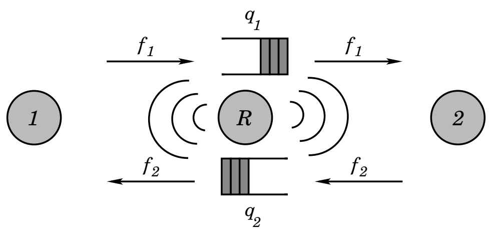

Fig. 3. System model

Whenever the hyperlink gets transmission opportunities, when both queues are non-empty, one packet from q_1 and one from q_2 can be transmitted together as a single packet using XOR coding. This scenario, in which transmitting a combination of packets results in decreasing the required number of transmissions, is referred as a *coding opportunity*. Whenever such a coding opportunity exists between the packets of two flows, the hyperlink encodes the packets and transmits the coded packet back to the adjacent nodes. However, if there is only one type of packet at the end of a slot, there are two options: (a) one of those packets gets transmitted without coding or (b) we wait for a future slot to receive a matching packet in the other queue to utilize the coding opportunity. We assume that transmissions within a type is according to a *first-in-first-out* basis.

Note that if we started with an empty system, at the end of every time slot, once a transmission (if any) is completed, there would be at most one type of packet. Therefore, the relay node faces one of three types of situations: (i) one packet of one type and at least one packet of another type; (ii) only one type of packet(s); (iii) no packets. The decision in situations (i) and (iii) is straightforward, one would code using XOR in situation (i) and transmit, whereas do nothing in situation (iii).

However in situation (ii), it is unclear as to what is the best course of action, do nothing (thus worsening delay) or transmit without coding (thus being inefficient). In other words, to wait or not to wait, that is the question. The hyperlink pays a price of C_t units for each transmission and C_h units to hold a packet for one time slot.

B. Markov Decision Process Model

This section focuses on the stochastic optimal control framework developed to find optimal control actions at the hyperlink. We model the system as a Markov Decision Process (MDP) [12, 13, 14]. This model uses state information based on what can be observed and prescribes an optimal action for each state. Then by solving the MDP model, we develop the structure of the optimal policy (such as stationary versus non-stationary, threshold versus switching curve, etc.) [15]. While we defer discussions on finding an optimal policy to the next chapter, we will give a brief background of MDP and details of formulating our problem as MDP here.

1. MDP Background

Markov Decision Processes (MDP) is mathematical framework used in modeling decision making problems where an optimal decision has to be made at each state in presence of uncertainty. An MDP contains: (i) A set of possible states called *State space (S)* (ii) A set of possible actions, *Action space (A)*, (iii) A real valued *reward or cost function R(s,a)* that defines the reward or cost of performing the action a at state s and (iv) *Transition probability matrix (P_a) for each* $a \in A$ that define the impact of action a on each state as probabilities of transition into other states. Refer to the Figure 4 for a sample MDP containing three states $\{S_0, S_1, S_2\}$ and two actions

$\{a_0, a_1\}$.

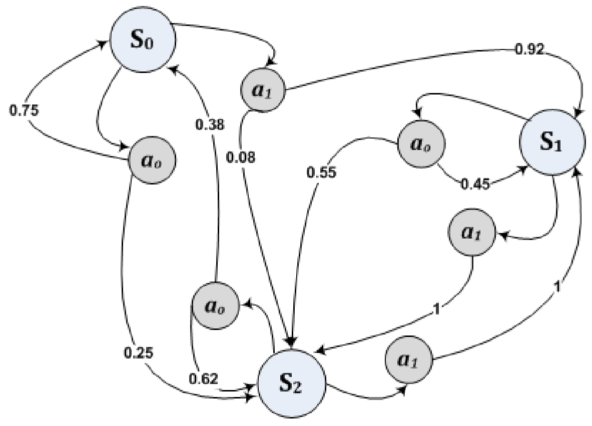

Fig. 4. An example of a Markov Decision Process with 3 states and 2 actions

The policy π is a mapping from S to A i.e. policy determines an action for each state. The performance of a policy is evaluated by computing the expected total reward (or cost). However, for infinite horizon problems, this typically yields infinite value. In that case, the most widely used and analytically tractable method to overcome this difficulty is *Discounting* which discounts a reward (or cost) $n-steps$ away by γ^n for discount rate $0 < \gamma < 1$. From the the γ-discounted cost case, the long run average cost can be obtained by letting the discount factor γ to approach 1.

A value function $V : S \rightarrow \Re$ that maps each state to a real value, where $V_n(s)$ for $n \rightarrow \infty$ represents the expected objective value in long run given that the system starts from the state s at time $n = 0$. MDPs are generally solved using *dynamic programming* and *reinforcement learning methods.* For example, the problem dynamics can be defined using *Bellman equations* which relates the value function to itself using a recurrence relation as follows,

$$V_{n+1}(s) = max_{a \in A} \left\{ R(s,a) + \gamma \Sigma_{s' \in S} \, P_a(s'|s,a) V_n(s') \right\} \tag{2.1}$$

Then the above equation can be solved using *value iteration* which is one of the algorithms to solve MDP. This technique iterates over all states and updates the value function V until it converges to the optimal value. *Note* than in the above equation, it is assumed that the problem under consideration has the objective of maximizing rewards(costs). Otherwise, we choose an action that minimizes the rewards(cost). At the end of value iteration, the mapping between each state in S and an optimal action in A is given as,

$$arg\,max_{a \in A}\,\{R(s,a) + \gamma \Sigma_{s' \in S}\, P_a(s'|s,a)V^*(s')\} \tag{2.2}$$

where $V^*(s)$ is the optimal value for state s at the time of convergence.

2. Model

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

To develop a strategy for that, we first define costs for latency and transmission. Let C_t be the cost for transmitting a packet and C_h be the cost for holding a packet for a length of time equal to one slot. Without loss of generality, we assume that if a packet was transmitted in the same slot it arrived, its latency is zero. Also, the cost of transmitting a coded packet is the same as that of a non-coded packet. That said, our objective is to derive an *optimal policy* that minimizes the *long-run average cost per slot.* For that we define the MDP $\{(Y_n, A_n), n \geq 0\}$ where $Y_n = (Y_n^1, Y_n^2)$ is the state of the *system* and A_n the control action chosen at time n. The state space (i.e. all possible values of Y_n) is the set $\{(i,j) : i \geq 0, j \leq 1 \text{ or } j \geq 0, i \leq 1\}$.

Let $C(Y_n, A_n)$ be the cost incurred at time n if action A_n is taken when the system is in state Y_n. Therefore,

$$C(Y_n, A_n) = C_h([Y_n^1 - A_n]^+ + [Y_n^2 - A_n]^+) + C_t A_n, \tag{2.3}$$

where $[x]^+ = \max(x, 0)$. The long-run average cost for some policy u is given by

$$g(u) = \lim_{N \to \infty} \frac{1}{N+1} E_u \left[\sum_{n=0}^{N} C(Y_n, A_n) | Y_0 = (0,0) \right], \tag{2.4}$$

where E_u is the expectation operator taken for the system under policy u. Notice that our initial state is an empty system, although the average cost would not depend on it. Our goal is to obtain the optimal policy u^* that minimizes $g(u)$. For that we first describe the probability law for our MDP and then in subsequent chapters develop a methodology to obtain the optimal policy u^*.

For the MDP $\{(Y_n, A_n), n \geq 0\}$, the probability law can be derived for $i \geq 0$ and

$j \geq 0$ as:

$$\begin{aligned}
&P_{\{A_n=1\}}\left((i,j),([i-1]^+,[j-1]^+)\right) = (1-p_1)(1-p_2),\\
&P_{\{A_n=1\}}\left((i,j),(\max(i,1),[j-1]^+)\right) = p_1(1-p_2),\\
&P_{\{A_n=1\}}\left((i,j),([i-1]^+,\max(j,1))\right) = (1-p_1)p_2,\\
&P_{\{A_n=1\}}\left((i,j),(\max(i,1),\max(j,1))\right) = p_1p_2,\\
&P_{\{A_n=0\}}\left((i,j),(i,j)\right) = (1-p_1)(1-p_2),\\
&P_{\{A_n=0\}}\left((i,j),(i+1,j)\right) = p_1(1-p_2),\\
&P_{\{A_n=0\}}\left((i,j),(i,j+1)\right) = (1-p_1)p_2,\\
&P_{\{A_n=0\}}\left((i,j),(i+1,j+1)\right) = p_1p_2,
\end{aligned}$$

where $P_{\{A_n=a\}}\left(Y_n, Y_{n+1}\right)$ is the transition probability from state Y_n to Y_{n+1} when the action $a \in \{0,1\}$ is chosen. Also note the caveats that: i and j cannot both be greater than 1; if $i = j = 0$, then $A_n = 0$; if $i > 0$ and $j > 0$, then $A_n = 1$.

CHAPTER III

DISTRIBUTED CONTROL FRAMEWORK

In this chapter, we discuss the design of distributed controllers that are used in intermediate nodes to make the right decision of whether to wait for future coding opportunity or to transmit without coding. The main objective is to minimize the average cost incurred due to transmissions and delays. In designing such cost minimizing controllers, we can define system states to include just queue lengths and/or the vector of waiting times associated with each of the packets. Depending upon the amount of local information required in designing the controllers, we group them into following categories:

- The set Π^{HR} of *randomized history dependent* policies, i.e. policies with actions that depend on knowing the history of states and actions up to the time when the decision needs to be made. Also these policies are *randomized* because the resulting action could be chosen randomly (as opposed to deterministically).
- The set Π^{MR} of all *randomized Markov* policies, i.e. policies where actions depend on knowing just the current state when the decision needs to be made. By definition, the action taken at time n could depend on n for Markov policies.
- The set Π^{SR} of *randomized stationary* policies, which are essentially randomized Markov policies that do not depend on n.
- Finally, we have the set Π^{SD} of *deterministic stationary* policies, in which actions are deterministic and solely depend on the current state but not n.

It can be seen (as shown in Puterman [12]) that

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

and the complexity of the algorithms, and hence our inability to determine them exactly also increases from left to right. It is also often true that the nearness to optimal performance decreases from left to right.

A. Transmission Policies

The controllers can apply different policies to make the decision of whether to wait or not to wait whenever there is no coding opportunity at the relay node. These policies use local information to decide the suitable action at every state. The local information could include: *queue length*, *arrival timestamps for each packet in the queue*, *total delay experienced in the queue by packets forwarded so far*, *total number of packets forwarded so far*, *entire history of states and actions performed at each state* etc. The important point to notice here is that with increase in the amount of information that need to be maintained, the design and implementation of controller become more complex.

In the rest of this section we explain a set of policies that uses thresholds to determine how long packets without coding pairs need to wait before being transmitted without coding. The thresholds are defined on either *queue length*, *waiting time of packets* or *both*. Based on the parameter(s) on which the threshold is defined, these policies are categorized into three classes: 1) *queue length threshold policies*, 2) *waiting time threshold policies* and 3) *queue length + waiting time threshold policies.*

1. *Queue Length Threshold Policies:* The *queue length threshold policies* require relay nodes to have a threshold L_i defined on the length of each queue i. The node will wait until either a matching packet arrives or the length of a non-empty queue exceeds its threshold.

2. *Waiting Time Threshold Policies:* In this group of policies, for each queue i, a

threshold W_i is defined on the time packets can wait before they are transmitted. Following are some of the variations of policies that use waiting time thresholds.

(a) *Longest wait time:* In this policy, the threshold W_i corresponds to the maximum time the packet at the head of queue i is allowed to wait. A packet will be forced to wait until either a matching packet arrives or it has waited for a sufficiently long time i.e. W_i time units.

(b) *Average wait time:* Here, the threshold W_i corresponds to the average waiting time of packets currently present in the queue i. Once the average waiting time of packets exceeds the threshold, the relay node will transmit the head of queue packet immediately irrespective of whether a matching packet is received or not.

(c) *Running average wait time:* This policy is a variation of Average wait time policy wherein while calculating the average waiting time, it also includes the waiting time of all packets which have been forwarded thus far.

(d) *Deficit based policy:* This policy is significantly different from other policies seen so far. It works around two parameters 1) *Target Average Waiting Time* $\bar{W_i}$ and 2) a *Threshold* W_i on the maximum deviation tolerable from the target waiting time. Relay nodes transmit, if needed, even without coding to ensure that the overall average waiting time of packets in the queue i is kept under $\bar{W_i} + W_i$.

3. *Queue Length + Waiting Time Threshold Policies:* These policies essentially combine queue length threshold policy with one of the waiting time policies discussed previously. Different combinations will yield multiple variants. Policies can have further variations by allowing the relay node to make a transmission without coding either when both queue length and waiting time thresholds are

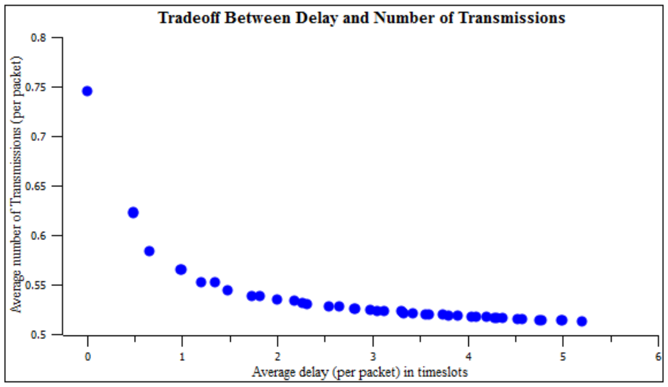

Fig. 5. Tradeoff between delay and number of transmissions at hyperlink using queue length threshold policy

exceeded (*Queue Length And Waiting Time*) or immediately after one of them is exceeded (*Queue Length Or Waiting Time*).

The most important design parameter in all the policies discussed above is the choice of thresholds. Large threshold values allow packets to wait for a long time looking for potential matching packets and hence leverage the benefits of network coding. But on the flip side, it worsens the delay. Whereas small thresholds are good in terms of delay but result in more transmissions by not exploiting network coding advantages. The Figure 5 shows the tradeoff between the average delay and number of transmissions per packet at an intermediate node using the queue length threshold policy.

B. Analysis Using Discrete Time Markov Chain Model

In this section, we introduce more formal notations and methods to analyze some of the policies discussed in the previous section. For the system model under consideration, we show how *Queue length threshold policy* and *Queue length Or Waiting* *time threshold policy* can be modeled as a Discrete Time Markov Chain (DTMC). We then perform steady-state analysis to derive exact expressions to compute the average delay and the average number of transmissions per timeslot.

1. Background

Consider a stochastic system that is being observed at discrete time slots $i = 1, 2, \ldots, n$. Then, a sequence of random variables $X_1, X_2, \ldots, X_n$ where X_i represents the state of the system at time slot i, is a Discrete Time Markov Chain provided that X_i satisfies the Markov property. The property refers to the condition that the state of the system in the next step (time slot) depends only on the current state and is independent of prior history. Formally the Markov property can be stated as,

$$Pr(X_{n+1} = x_{n+1} | X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = Pr(X_{n+1} = x_{n+1} | X_n = x_n) \tag{3.1}$$

The set of all possible states the system can be in during a time slot is defined using the state space S and the probability of jumping from a state i to state j in one step (p_{ij}) is defined using a transition probability matrix $P = (p_{ij})$ where,

$$p_{ij} = Pr(X_k = j | X_{k-1} = i)$$

$$and \sum_j p_{ij} = 1$$

The stationary distribution $\pi_j = lim_{n \to \infty} p_{ij}^n$, for all i, j exists for a Markov chain if the following conditions hold: if the Markov chain is (i) *irreducible* i.e. every state $i \in S$ is reachable from every other state $j \in S$. (ii) *aperiodic* where the period of a state i is defined as $gcd\{n \geq 1 : p_{ii}^n > 0\}$ and the state is aperiodic if its period is 1. Subsequently, if every state in S is aperiodic then the markov chain itself is aperiodic and (iii) *positive recurrent* i.e. once a state is left the probability of returning to it

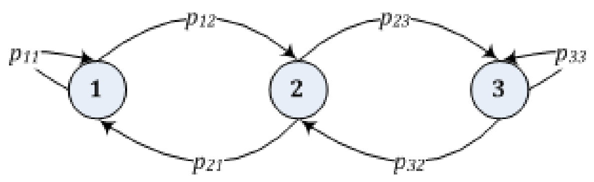

Fig. 6. An example of an irreducible, aperiodic and positive recurrent Markov chain

in a future time slot is 1 (recurrent) and the expected time between visits is finite. Figure 6 shows an example of a Markov chain that satisfies all of the above conditions.

Once it is found that the stationary distribution exists for a Markov chain it can be obtained by solving the equations,

$$\pi = \pi P \tag{3.2}$$

$$\Sigma_i \pi(i) = 1 \tag{3.3}$$

In the subsequent sections, we show how the policies using thresholds on only Queue length and both Queue length and Waiting time can be modeled as DTMC.

2. Analysis of Queue Length Policy

Consider a system of two queues as shown in Figure 7. Time is slotted so that in each slot an entity arrives into queue i with probability p_i for $i = 1, 2$. Also with probability $(1 - p_i)$ nothing arrives into queue i in a slot. A maximum of 1 transmission occurs at the end of a slot. If there is one packet of each type, then both are transmitted together (and we count that as 1 transmission). However if there is only one type of packet at the end of a slot we need to decide whether to transmit it individually or wait for a future slot to pair it with another packet.

We consider a queue-length based threshold policy such that if the number of

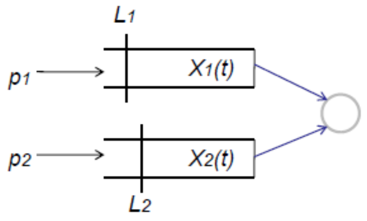

Fig. 7. Queueing model at an intermediate node using queue length threshold policy

packets of type i is less than or equal to L_i (and of type $3-i$ is zero), then we wait for a future slot and transmit nothing in the present slot. However, if the number of packets of type i is greater than L_i (and of type $3-i$ is zero), then we transmit a type i packet individually without pairing with type $3-i$. Further, we always transmit a pair of type i and $3-i$ if both are available. Note: $i = 1, 2$.

Let $X_i(t)$ be the number of packets in buffer i at the beginning of the t^{th} slot before any arrival or transmission. Then the bivariate stochastic process $\{(X_1(t), X_2(t)), t \geq 0\}$ is a discrete-time Markov chain. For a system using thresholds $L_1 = L_2 = 1$, the Markov chain is shown in Figure 8 and in general, for any system using the queue-length based threshold policy, the states are $(0,0)$, $(1,0)$, $(2,0)$, …, $(L_1,0)$, $(0,1)$, $(0,2)$, …, $(0,L_2)$. Define α as a parameter such that

$$\alpha = \frac{(1-p_1)p_2}{(1-p_2)p_1}.$$

Let $\pi_{i,j}$ be the steady-state probabilities of the Markov chain, then we can show that

$$\begin{aligned}
\pi_{0,0} &= \frac{1}{\left(\frac{1-\alpha^{L_1+1}}{1-\alpha}\right) + \left(\frac{1-1/\alpha^{L_2+1}}{1-1/\alpha}\right) - 1} \\
\pi_{i,0} &= \alpha^i \pi_{0,0} \\
\pi_{0,j} &= \pi_{0,0}/\alpha^j
\end{aligned}$$

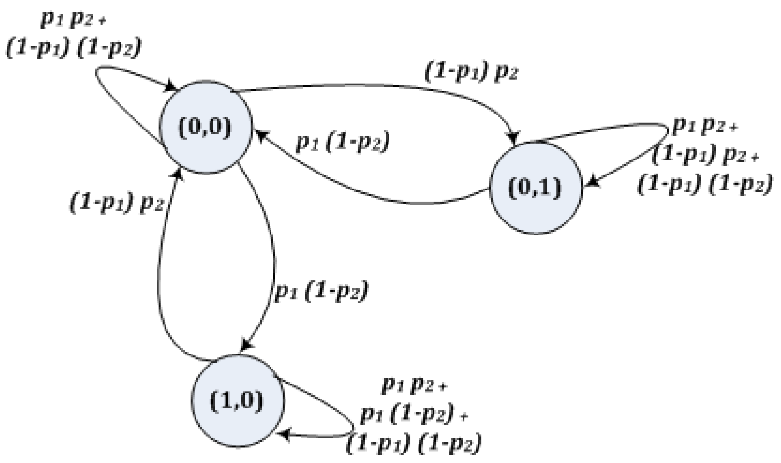

Fig. 8. Markov chain model for the queue length threshold policy with thresholds $L_1 = L_2 = 1$

for all $0 < i \leq L_1$ and $0 < j \leq L_2$.

Using that we can obtain the following performance measures: the expected number of transmissions per slot (we count an individual and a paired transmission using network coding both as 1 transmission) is

$$p_1p_2\pi_{0,0} + p_2\sum_{i=1}^{L_1}\pi_{i,0} + p_1\sum_{j=1}^{L_2}\pi_{0,j} + p_1(1-p_2)\pi_{L_1,0} + p_2(1-p_1)\pi_{0,L_2},$$

and the average delay (i.e. number of slots to transmit) for entity of type 1 is

$$\sum_{i=1}^{L_1} i\pi_{i,0}/p_1,$$

and type 2 is

$$\sum_{j=1}^{L_2} i\pi_{0,j}/p_2.$$

3. Analysis of Queue Length and Waiting Time Policy

This policy is an extension of Queue length based threshold policy. Here, if the number of packets of type i is less than or equal to L_i (and of type $3-i$ is zero) and if the waiting time of packet at the head of type i queue is less than or equal to W_i, then we wait for a future slot and transmit nothing in the present slot. However, if the number of packets of type i is greater than L_i (and of type $3-i$ is zero) or if the earliest packet in the queue has waited for more than W_i time slots, then we transmit the type i packet individually without pairing with type $3-i$. Further, we always transmit a pair of type i and $3-i$ if both are available. Note: $i = 1, 2$.

To model the system as a Markov chain, in addition to queue length, we include the vector of waiting times of all packets (currently waiting in the queue) into state definition. Let $X_i(t)$ be the number of packets in buffer i at the beginning of the t^{th} slot before any arrival or transmission. Let $W(t)$, where $0 \leq |W(t)| \leq \max\{X_1(t), X_2(t)\}$ be an array of waiting times of packets either in queue 1 or 2. Note that we need just a single array of waiting times because as per our system definition, one of the queues will be empty after the transmission.

Then the multivariate stochastic process $\{(X_1(t), X_2(t), W(t)), t \geq 0\}$ is a discrete-time Markov chain. However, unlike in the model where we considered only queue length, here the state space is very large and it is extremely difficult to derive closed-form expressions for performance measures. Hence, we provide the following expressions to compute the expected number of transmissions per slot and and the average delay (i.e. number of slots to transmit) in terms of steady state probabilities $\pi_{X_1,X_2,W}$. These probabilities can be easily computed using a computer program.

The expected number of transmissions per slot (we count an individual and a paired transmission using network coding both as 1 transmission) is

$$p_1 p_2 \pi_{0,0,W} +$$
$$p_2 \sum_{i=1}^{L_1} \pi_{i,0,W} + p_1 \sum_{j=1}^{L_2} \pi_{0,j,W} +$$
$$p_1(1-p_2)\pi_{L_1,0,W} + p_2(1-p_1)\pi_{0,L_2,W} +$$
$$(1-p_2)\sum_{i=1}^{L_1} \pi_{i,0,\{W_1,\ldots\}} + (1-p_1)\sum_{j=1}^{L_2} \pi_{0,j,\{W_2,\ldots\}},$$

and the average delay (i.e. number of slots to transmit) for a packet of type 1 is

$$\sum_{i=1}^{L_1} \pi_{i,0,W} \sum_{k=1}^{i} W[k],$$

and type 2 is

$$\sum_{j=1}^{L_2} \pi_{0,j,W} \sum_{k=1}^{j} W[k].$$

CHAPTER IV

FINDING THE OPTIMAL POLICY

As we described in the previous chapter, our goal is to obtain the optimal policy u^* that minimizes $g(u)$ defined in Equation (2.4). For that we first find the space the policy would live in and then find the optimal policy within that space.Our first question is: what is the appropriate state space: is it just queue length, or should we also consider waiting time?

A. Should We Maintain Waiting Time Information?

Intuition tells us that if a packet has not been waiting long enough then perhaps it could afford waiting a little more but if a packet has waited too long, it may be better to just transmit it. That seems logical considering that we tried our best to code but we cannot wait too long because it hurts in terms of holding costs. Also, one could get waiting time information from time-stamps on packets that are always available. Given that, would we be making better decisions by also keeping track of waiting times of each packet? We answer that question by means of a theorem which requires the following lemma for a generic MDP $\{(X_n, D_n), n \geq 0\}$ where X_n is the state of the MDP and D_n is the action at time n.

***Lemma* 1** *(Puterman [12]) For an MDP $\{(X_n, D_n), n \geq 0\}$, given any history dependent policy and starting state, there exists a randomized Markov policy with the same long-run average cost.*

Proof: The proof is adapted from Puterman [12]. Let S be the state space corresponding to all possible values of X_n and D be the action space corresponding to all possible values of D_n. Consider a policy $\rho \in \Pi^{HR}$ and another policy $\sigma \in \Pi^{MR}$. For a given state $i \in S$, for every $j \in S$ and $d \in D$, define the randomized Markov policy σ via the decision rule to adopt action d with the same probability as that with the policy ρ. Therefore if the history-dependent policy ρ picks action d at time n given the current state is j and initial state i with probability

$$P^{\rho}\{D_n = d | X_n = j, X_0 = i\}$$

then the Markov randomized policy also picks action d with the same probability in state j. Therefore we have

$$\begin{aligned} P^{\sigma}\{D_n = d | X_n = j\} = P^{\sigma}\{D_n = d | X_n = j, X_0 = i\} = \\ P^{\rho}\{D_n = d | X_n = j, X_0 = i\}. \end{aligned} \tag{4.1}$$

Next, we would like to show that

$$P^{\sigma}\{X_n = j, D_n = d | X_0 = i\} = P^{\rho}\{X_n = j, D_n = d | X_0 = i\}. \tag{4.2}$$

For that we use the principle of mathematical induction. For $n = 0$, Equation (4.2) is satisfied by letting $n = 0$ in Equation (4.1) when $i = j$ and trivially when $i \neq j$. Assume that Equation (4.2) holds for $n = 1, 2, \ldots, k-1$. To show that also holds for $n = k$, we start by considering the following (with the second equation is due to the induction hypothesis that it holds for $n = k - 1$):

$$\begin{aligned} P^{\rho}\{X_k = j | X_0 = i\} = \\ \sum_{\ell \in S} \sum_{d \in D} (P^{\rho}\{X_{k-1} = \ell, D_{k-1} = d | X_0 = i\} \\ P\{X_k = j | X_{k-1} = \ell, D_{k-1} = d\}) = \\ \sum_{\ell \in S} \sum_{d \in D} (P^{\sigma}\{X_{k-1} = \ell, D_{k-1} = d | X_0 = i\} \\ P\{X_k = j | X_{k-1} = \ell, D_{k-1} = d\}) = \\ P^{\sigma}\{X_k = j | X_0 = i\}. \end{aligned}$$

Thus we have

$$\begin{aligned} P^{\sigma}\{X_k = j, D_n = d | X_0 = i\} = \\ P^{\sigma}\{D_n = d | X_k = j\} P^{\sigma}\{X_k = j | X_0 = i\} = \\ P^{\rho}\{D_n = d | X_k = j\} P^{\rho}\{X_k = j | X_0 = i\} = \\ P^{\rho}\{X_k = j, D_n = d | X_0 = i\}. \end{aligned}$$

Therefore, by the principle of mathematical induction Equation (4.2) is satisfied for all $n \geq 0$. Since both policies ρ and σ yield the same joint probability distribution of states and actions, they both will yield the same long-run average cost.

Using the above lemma we show next that it is not necessary to maintain waiting time information.

***Theorem* 2** *For the MDP $\{(Y_n, A_n), n \geq 0\}$, if there exists a randomized history dependent policy that is optimal, then there exists a randomized Markov policy u^* that minimizes $g(u)$ defined in Equation (2.4). Further, one cannot find a policy which also uses waiting time information that would yield a better solution than $g(u^*)$.*

Proof: From Lemma 1, if the MDP $\{(Y_n, A_n), n \geq 0\}$ has a history dependent policy that is optimal, then we can construct a randomized Markov policy that yields the exact same long-run average cost given $Y_0 = (0,0)$. Therefore, if there exists a randomized history dependent policy that is optimal, then there exists a randomized Markov policy u^* that minimizes $g(u)$ defined in Equation (2.4).

Knowing the entire history of states and actions one can always determine the history of waiting times as well as the current waiting times of all packets. Therefore the optimal policy u' that uses waiting time information is equivalent to a policy in Π^{HR}. From Lemma 1, we can always find a randomized Markov policy that yields the same optimal solution as $g(u')$.

B. Structure of the Optimal Policy

Having made a case for not considering the waiting times in the state of the system, the next question is how does the optimal policy look like and in what space of policies does it live. In the MDP literature (see Sennott [16]), the conditions for the structure and location of optimal policy usually rely on the results of the infinite horizon β-discounted cost case and let β approach 1 to obtain the average cost case. In that light, for our MDP $\{(Y_n, A_n), n \geq 0\}$, the total expected discounted cost incurred by a policy θ is

$$V_{\theta,\beta}(i,j) = E_\theta \left[\sum_{n=0}^{\infty} \beta^n C(Y_n, A_n) | Y_0 = (i,j) \right].$$

If $V_\beta(i,j) = \min_\theta V_{\theta,\beta}(i,j)$ corresponds to the policy that minimizes the total expected discounted cost, then $V_\beta(i,j)$ satisfies the optimality equation

$$\begin{aligned} V_\beta(i,j) = \min_{a \in \{0,1\}} [C_h([i-a]^+ + [j-a]^+) + C_t a \\ +\beta \sum_{k,\ell} V_\beta(k,\ell) P\{Y_{n+1} = (k,\ell) | Y_n = (i,j), A_n = a\}]. \end{aligned}$$

Further, the stationary policy that minimizes the above equation is an optimal policy for the infinite horizon β-discounted cost problem.

However, the long-run average cost case is not as easy to state when the optimal policy is stationary. In particular, to determine if there exists a stationary policy u^* that minimizes $g(u)$ defined in Equation (2.4) one must find a constant $\hat{g}$ and a bounded function $v(i,j)$ (if they exist) that satisfy the average cost optimality equation

$$\hat{g} + v(i,j) = \min_{a\in\{0,1\}} [C_h([i-a]^+ + [j-a]^+) + C_t a$$
$$+ \sum_{k,\ell} v(k,\ell) P\{Y_{n+1} = (k,\ell) | Y_n = (i,j), A_n = a\}]$$

In that case, the stationary policy that minimizes the above equation is an optimal policy with $\hat{g} = g(u^*)$. Next we describe a lemma that specifies the conditions when $\hat{g}$ and $v(i,j)$ exist. For that define

$$v_\beta(i,j) = V_\beta(i,j) - V_\beta(0,0).$$

***Lemma* 3** *(Sennott [16]) There exist a constant $\hat{g}$ and a function $v(i,j)$ satisfying the average cost optimality equation if the following two conditions hold: (i) there exist non-negative $M_{i,j}$ such that $v_\beta(i,j) \le M_{i,j}$ and*

$$\sum_{k,\ell} P\{Y_{n+1} = (k,\ell) | Y_n = (i,j), A_n = a\} M_{k,\ell} < \infty$$

for all i, j, β and a; (ii) there exists a non-negative N such that $v_\beta(i,j) \ge -N$ for all i, j and β.

Proof: See Sennott [16] for a proof for the more generic MDP.

Using the above lemma we show next that the MDP defined in this paper has an optimal policy that is stationary.

***Theorem* 4** *For the MDP $\{(Y_n, A_n), n \geq 0\}$, there exists a stationary policy u^* that minimizes $g(u)$ defined in Equation (2.4).*

Proof: As described earlier it is sufficient to show that the two conditions in Lemma 3 are met to show the existence of $\hat{g}$ and $v(i, j)$. Due to lack of space we only provide an idea of the proof. Refer to Sennott [16] and follow the proof of the example from communication networks. Consider the stationary policy $\hat{\theta}$ of always transmitting in states $(i, 0)$ and $(0, j)$ for any $i > 0$ and $j > 0$. Using the policy $\hat{\theta}$ we can find an upper bound on $V_{\theta,\beta}(0, 0)$ as $C_t(p_1 + p_2 - p_1p_2)/(1 - \beta)$. Therefore we can carefully obtain a bound on $V_{\theta,\beta}(i, j)$ in terms of $V_{\theta,\beta}(0, 0)$, p_1, p_2, β, i and j to obtain $M_{i,j}$. The condition (ii) on finding an N is straightforward since all the costs are positive.

Now that we know that the optimal policy is stationary, the question is how do we find it. The standard methodology to obtain stationary policy for infinite-horizon average cost minimization problem is to use a linear program as described below.

Consider an MDP $\{(X_n, D_n), n \geq 0\}$ where X_n is the state and D_n is the action at time n. Assume that the MDP has a finite number of states in the state space and the number of possible actions is also finite. Assume that the Markov chain resulting out of any policy is irreducible. Let u be a stationary randomized policy described for state $X_n = i$ and action $D_n = a$ as follows:

$$u_{ia} = P\{D_n = a | X_n = i\}$$

for all i in the state space and all a in the action space. Note that u_{ia} is the probability of choosing action a when the system is in state i. Further, define the expected cost incurred when the system is in state i and the action is a as

$$c_{ia} = E[C(X_n, D_n) | X_n = i, D_n = a]$$

where $C(X_n, D_n)$ is the cost incurred at time n if action D_n is taken when the system is in state X_n.

Lemma **5** *(Serin and Kulkarni [17]) The optimal randomized policy u^*_{ia} that minimizes the long-run average cost per unit time (equal to the length of a slot) can be computed as*

$$u^*_{ia} = \frac{x^*_{ia}}{\sum_b x^*_{ib}}$$

where $x^* = [x^*_{ia}]$ is the optimal solution to the linear program:

$$\begin{aligned}
\textit{Minimize} \quad & \sum_i \sum_a c_{ia} x_{ia} \\
\textit{subject to} \quad & \sum_i \sum_a x_{ia} = 1 \\
& \sum_a x_{ja} - \sum_i \sum_a p_{ij}(a) x_{ia} = 0 \ \ \forall j \\
& x_{ia} \geq 0 \ \ \forall i, a.
\end{aligned}$$

Proof: See Ross [13] for a proof for the maximizing average rewards case.

As described in Ross [13], the linear program (LP) produces for each i optimal values x^*_{ia} that are all zero except one a which would be 1. Hence the optimal policy would infact be a stationary deterministic policy.

However, we cannot directly apply the above results to our MDP $\{(Y_n, A_n), n \geq 0\}$, as our MDP has infinite states and the Markov chain under every policy is not irreducible (for example if we always transmit, it is not possible to reach some of the states). To circumvent that, we construct a finite size LP with N states and force it to be irreducible by creating dummy transitions with probability $\epsilon > 0$ between some states. Let us call this $LP(N, \epsilon)$. From the lemma above, $LP(N, \epsilon)$ has a stationary deterministic policy that is optimal. By letting $N \to \infty$ and $\epsilon \to 0$ we argue that our MDP would have an optimal deterministic policy. With that said, it is not efficient to obtain the optimal policy by solving $LP(N, \epsilon)$ for large N and small ϵ.

However, we now know that the optimal policy is stationary deterministic. But, how do we find it? If we know that the optimal policy satisfies some monotonicity properties then it is possible to search through the space of stationary deterministic policies and obtain the optimal one.

***Lemma* 6** *(Puterman [12]) For the MDP $\{(Y_n, A_n), n \geq 0\}$ the optimal policy that minimizes the long-run average cost is non-decreasing in i and j if the following conditions are met: (i) $C_h([i-a]^+ + [j-a]^+) + C_t a$ is non-decreasing in i and j for all $a \in \{0, 1\}$ and super-additive; (ii) the function $q(r, s|i, j, a)$ defined as*

$$\sum_{k \geq r, \ell \geq s} P\{Y_{n+1} = (k, \ell) | Y_n = (i, j), A_n = a\}$$

is non-decreasing in i and j for all $a \in \{0, 1\}$, r and s as well as super-additive.

Proof: See Puterman [12] for a proof for a generic MDP. By rewriting those for this specific MDP, we can prove the Lemma.

Using the above lemma we show that the structure of the optimal policy for our model is stationary deterministic and monotonic in terms of the number in the system.

C. Obtaining the Optimal Deterministic Stationary Policy

We have shown in the previous section that the optimal policy is stationary, deterministic and monotonic. The next thing to do is find it. Notice that we only need to consider the subset of deterministic stationary policies, Π^{SD}. From among the policies in this set Π^{SD} we obtain the optimum policy. Given that the structure of the optimal policy is monotone, it is fairly straightforward to see that it is threshold-type.

In the next theorem we show how to compute 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 transmit without coding if $i > L_1^*$; $(0,j)$ is to wait if $j \leq L_2^*$ and transmit without coding if $j > L_2^*$.

***Theorem* 7** *The optimal thresholds L_1^* and L_2^* are*

$$(L_1^*, L_2^*) = \arg\min_{L_1,L_2} C_t\tau(L_1,L_2) + C_h\lambda(L_1,L_2)$$

where

$$\tau(L_1,L_2) = p_1p_2\pi_{0,0} + p_2\sum_{i=1}^{L_1}\pi_{i,0} +$$
$$p_1\sum_{j=1}^{L_2}\pi_{0,j} + p_1(1-p_2)\pi_{L_1,0} + p_2(1-p_1)\pi_{0,L_2},$$
$$\lambda(L_1,L_2) = \sum_{i=1}^{L_1} i\pi_{i,0} + \sum_{j=1}^{L_2} j\pi_{0,j},$$

for which

$$\begin{aligned}
\pi_{0,0} &= \frac{1}{\left(\frac{1-\alpha^{L_1+1}}{1-\alpha}\right) + \left(\frac{1-1/\alpha^{L_2+1}}{1-1/\alpha}\right) - 1}\\
\pi_{i,0} &= \alpha^i\pi_{0,0}\\
\pi_{0,j} &= \pi_{0,0}/\alpha^j \quad \textit{with}\\
\alpha &= \frac{(1-p_2)p_1}{(1-p_1)p_2}.
\end{aligned}$$

Proof: Let L_1 and L_2 be the thresholds and our objective is to find their corresponding optimal value. Let $X_i(t)$ be the number of type i packets at the beginning of the t^{th} slot before any arrival or transmission. It is crucial to note that this observation time is different from when the MDP is observed. Then the bivariate stochastic process $\{(X_1(t), X_2(t)), t \geq 0\}$ is a discrete-time Markov chain. The states are $(0,0)$,

$(1,0)$, $(2,0)$, …, $(L_1,0)$, $(0,1)$, $(0,2)$, …, $(0,L_2)$. Define α as a parameter such that

$$\alpha = \frac{(1-p_2)p_1}{(1-p_1)p_2}.$$

Let $\pi_{i,j}$ be the steady-state probabilities of the Markov chain. The balance equations for $0 < i \leq L_1$ and $0 < j \leq L_2$ are:

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

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

$$\begin{aligned} \pi_{0,0} &= \frac{1}{\left(\frac{1-\alpha^{L_1+1}}{1-\alpha}\right) + \left(\frac{1-1/\alpha^{L_2+1}}{1-1/\alpha}\right) - 1} \\ \pi_{i,0} &= \alpha^i \pi_{0,0} \\ \pi_{0,j} &= \pi_{0,0}/\alpha^j \end{aligned}$$

for all $0 < i \leq L_1$ and $0 < j \leq L_2$.

The expected number of transmissions per slot (we count an individual and a paired transmission using network coding both as one transmission) is

$$\tau(L_1, L_2) = p_1p_2\pi_{0,0} + p_2\sum_{i=1}^{L_1}\pi_{i,0} +$$
$$p_1\sum_{j=1}^{L_2}\pi_{0,j} + p_1(1-p_2)\pi_{L_1,0} + p_2(1-p_1)\pi_{0,L_2}.$$

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

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

Thus the long-run average cost per slot is

$$C_t\tau(L_1, L_2) + C_h\lambda(L_1, L_2)$$

which 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.

D. Optimal Offline Scheduling

In offline scheduling, it is assumed that entire packet arrival sequence for both sources for a time period of τ slots is known in advance. Then the problem of finding an optimal schedule that minimizes the long run average system cost reduces to a minimum cost perfect matching problem.

1. Construction of Bipartite Graph

Given the packet arrival schedule of two sources $Sch_i(i = 1, 2)$ for a period of τ slots, where $Sch_i[k] \in 0, 1$ representing no arrival and exactly one packet arrival from source i at k^{th} time slot respectively, a complete bipartite graph $G(V_1, V_2, E)$ used to find an optimal schedule can be constructed as follows.

$V_i(i = 1, 2)$ includes a set of vertices corresponding to packet arrivals from source of type i i.e. for each instance k when there is an arrival of type i packet i.e. $Sch_i[k] = 1$, a vertex is added to V_i. In addition, V_i includes a special type of nodes called *dummy* *nodes* $D_i \subset V_i$. These dummy nodes are added to V_i when there is no arrival of type i in a time slot k but there is an arrival from other source $3-i$ in the same time slot.

Once vertices are created in V_1 and V_2, edges are added between certain pairs of nodes (v_1, v_2) where $v_1 \in V_1$, $v_2 \in V_2$ and edge weights are assigned. There are three types of edges created:

1. Edges connecting two non-dummy nodes: These edges represent coded transmissions. The weight of an edge connecting two non-dummy nodes represents the cost of coding the two packets represented by these nodes together and transmitting them as a single packet. If these two packets are not from the same time slot, then the cost involves the cost of holding the packet which had arrived first until the time slot when the next one arrives.

2. Edges between a dummy and non-dummy node: There may be cases where sending a packet without coding will prove to be more cost effective than to wait for a matching packet and then to code *(Example: when the rate of type i packets is too low while the holding cost for a packet of type* $3-i$ *is high, then it is always better to send* $3-i$ *type packets without coding).* To allow transmissions without coding, dummy nodes are connected with their respective non-dummy nodes (of other type) through edges whose weights include just the cost of transmission.

   Also, it can be noted that the cost of holding a packet for a few time slots anticipating a matching packet and then transmitting it without coding is always higher than transmitting the packet without coding immediately on its arrival. To impose this condition, the edges of this type are created only between nodes from the same time slot.

3. Edges connecting two dummy nodes: These edges are created with *zero weights*

and they have no significance in the final cost. They are added just to allow the matching found to be complete.

Let C_t and C_h represent the cost per transmission (coded transmissions are counted as 1) and the cost of holding a packet per time slot respectively. The weights are assigned to edges as follows,

$$W(v_{1k_1}, v_{2k_2}) = \begin{cases} 0, & \text{both nodes are dummy} \\ |k_1 - k_2|C_h + C_t, & \text{both nodes are non-dummy} \\ C_t, & \text{only one of the nodes is dummy and } k_1 = k_2 \\ \infty, & \text{only one of the nodes is dummy and } k_1 \neq k_2 \end{cases}$$

where v_{ik} represents a node created for time slot k in V_i.

A bipartite graph constructed for packet schedules $Sch_1 = \{1,0,1,1,0\}$ and $Sch_2 = \{0,0,1,0,1\}$ is shown in the Figure 9. Note that, edges with ∞ cost are not shown in the graph and those with zero cost are displayed as dashed lines.

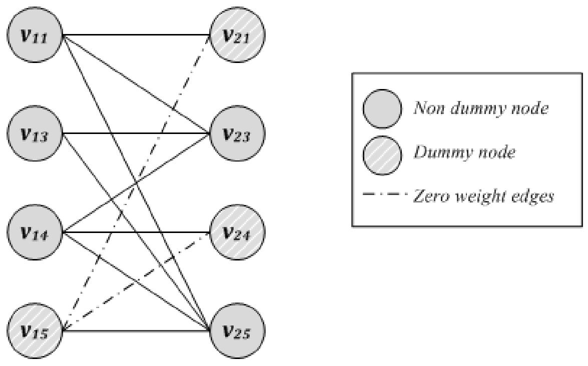

Fig. 9. Bipartite graph for packet schedules {1,0,1,1,0} $and\,\{0,0,1,0,1\}$.

2. Finding Optimal Schedule

A *matching* of a graph G is a subset M of edges E, that connects a node in V_1 to exactly one node from V_2 and vice versa such that no two edges share a common node. A perfect matching is a matching that connects every vertex in one of the partitions to a vertex in the other partition i.e. every vertex $v \in V_1 \cup V_2$ is incident to exactly one edge of M. The cost of a matching is the sum of the weights of all edges in M.

The problem of finding a minimum cost perfect matching for a graph has been widely studied and there are quite a few algorithms proposed to solve this problem (e.g. Hopcroftś [18], Hungarian [19] etc.,). Once a matching M is found, the optimal schedule i.e. the actions to be performed by relay nodes for every packet can be inferred from the matching using the logic explained below.

For an edge $e \in M$,

- if both incident nodes are non-dummy i.e. both correspond to real packets, then these two packets must be coded and transmitted together. If they are not from the same time slot, then the packet which arrived earlier must be forced to wait until the other one arrives.

- if exactly one of the incident nodes is dummy, then the packet represented by the other node must be transmitted immediately on its arrival without coding.

- if both incident nodes are dummy, then this edge does not relate to the schedule of any packet.

Except for edges where both incident nodes are dummy, every edge $e = (v_{1k_1}, v_{2k_2}) \in M$ corresponds to a transmission. In case of transmissions without coding, the delay experienced by the packet is zero. For coded transmissions, the packet which arrived first has to wait for $|k_1 - k_2|$ time slots. Thus, the optimal schedule and the performance measures such as average delay and number of transmissions can be found from the maximum matching found.

CHAPTER V

EXPERIMENTS AND RESULTS

In this chapter, we present the results of various experiments we performed to validate the analytical findings. In particular, we consider the following three scenarios.

1. *Single hyperlink (Bernoulli arrivals)* Experiments are run for the simple 3-node relay network model (Figure 3) with packet arrivals from end nodes following Bernoulli distribution.

2. *Single hyperlink (Gilbert-Elliot arrivals)* In order to further validate the findings from the previous experiments, additional simulations are run for the single hyperlink scenario with end nodes using Gilbert-Elliot model to generate packets.

3. *Line network (Bernoulli arrivals):* Finally, we show how our findings from the experiments for a single hyperlink case apply to a larger network by running simulations for a line network containing 4 nodes.

In simulations we use different policies that are based on queue length only, waiting time only, both waiting time and queue length and randomized thresholds. The following are the policies that we mainly focus on:

1. *Opportunistic Coding:* This is a naive algorithm that does not wait for coding opportunities. At every transmission opportunity, intermediate node will perform coding if packets are available from all compatible flows. Otherwise, packets from the non-empty queue will be transmitted immediately without coding.

2. *Queue Length Threshold:* This is a stationary deterministic (SD) policy with a fixed threshold (L_i) defined on the maximum length of queue i. In our analysis

we claim that this policy is optimal for the Bernoulli case.

3. *Randomized Queue Length Threshold:* This is a stationary policy that randomizes (SR) over the deterministic thresholds used in stationary deterministic (SD) policy. We expect that it would not perform any better than deterministic queue length threshold based policies.

4. *Queue Length -plus- Waiting Time Threshold:* This is a history dependent policy (HR) that uses information related to both queue length and waiting time of packets. It is likely to give the best possible performance. We perform a brute-force search over the space of thresholds and find the optimal case.

5. *Waiting Time Threshold:* This is a history dependent policy (HR) that uses only information related to waiting time of packets. Using the results for this policy we try to illustrate that history on its own is only of limited value.

These policies are implemented at relay nodes and we compare their performance in terms of ollowing measures: (i)*average delay* which is the average number of time slots a packet has to wait in the queue of an intermediate node before getting forwarded, (ii)*average number of transmissions* which is the measure of average number of transmissions that are required at an intermediate node to forward a single packet and (iii) *average cost* incurred due to transmissions and delays.

A. Bernoulli Arrival Model

Our first set of simulations is with a single hyper-link and packet flows into the two queues following Bernoulli distribution of rates p_1 and p_2 respectively. We illustrate the performance of different policies in Figures 10, 11, 12 and 13. We see that the queue-length-based policy is optimal as our analysis has suggested. Further as ex-

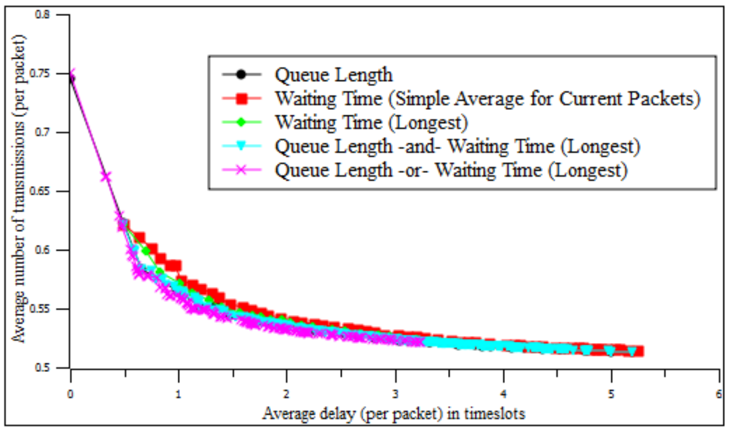

Fig. 10. Tradeoff between delay and transmissions in a single hyperlink with Bernoulli arrival rates $(p_1, p_2) = (0.5, 0.5)$

pected the randomized policy is identical to the queue-length-based policy. It can also be observed that while the queue-length-plus-waiting-time based policy performs well, the pure waiting-time-based policy is sub-optimal.

B. Evaluating Policies for Gilbert-Elliot Arrival Model

In our next model, the packet arrivals into queues follow Gilbert-Elliot model (Figure 14). It is a two state (ON/OFF) Markov process. It generates exactly one packet per time slot when the system is in ON state. The probabilities p_{on} and p_{off} control how long the system stays in ON and OFF states respectively. These parameters can be modified suitably to create the desired level of burstiness in traffic. The objective of experiments using Gilbert-Elliot arrival process is to test out the policies under different levels of bursty traffic conditions. Results are shown in Figures 15 and 16. We observe that the queue-length-based policy is robust under this arrival model too and it is near optimal.

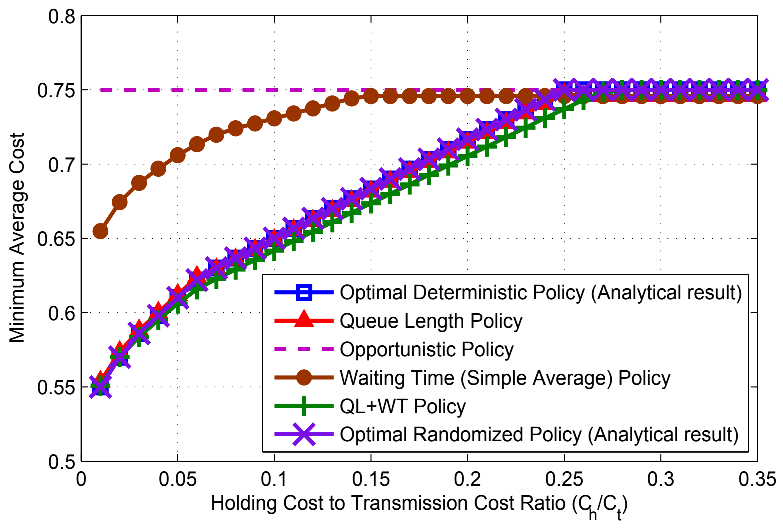

Fig. 11. Comparison of the minimum average cost (per packet) in a single hyperlink with Bernoulli arrival rates $(p_1, p_2) = (0.5, 0.5)$

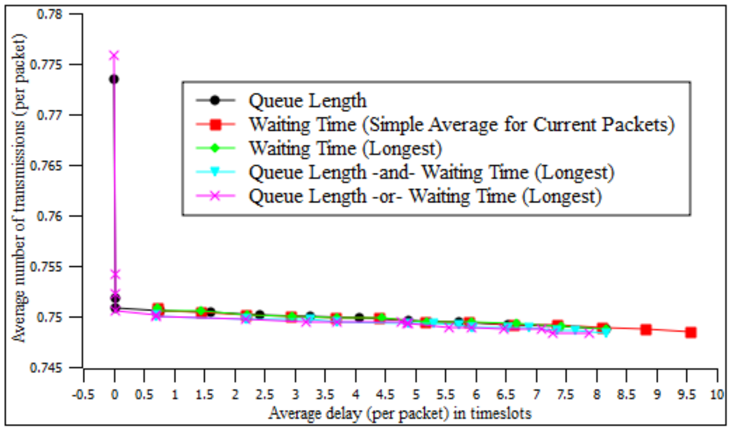

Fig. 12. Tradeoff between delay and transmissions in a single hyperlink with Bernoulli arrival rates $(p_1, p_2) = (0.9, 0.3)$

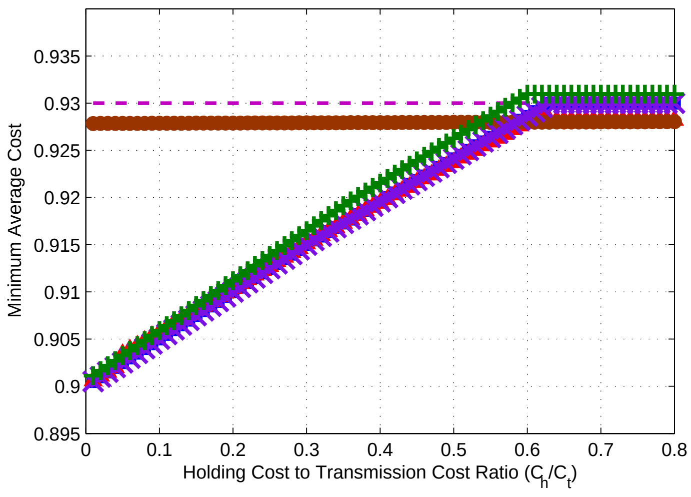

Fig. 13. Comparison of the minimum average cost (per packet) in a single hyperlink with Bernoulli arrival rates $(p_1, p_2) = (0.9, 0.3)$

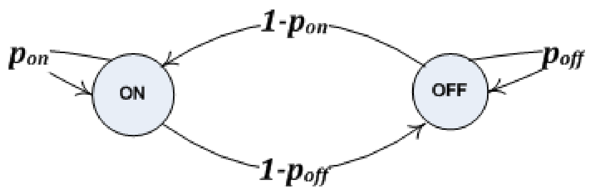

Fig. 14. Gilbert-Elliot model

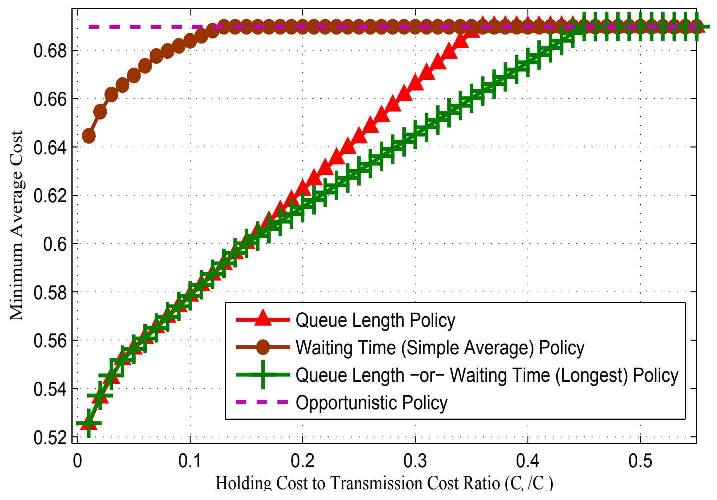

Fig. 15. Comparison of threshold policies using smooth Gilbert-Elliot sources ($p_{on} = p_{off} = 0.3$)

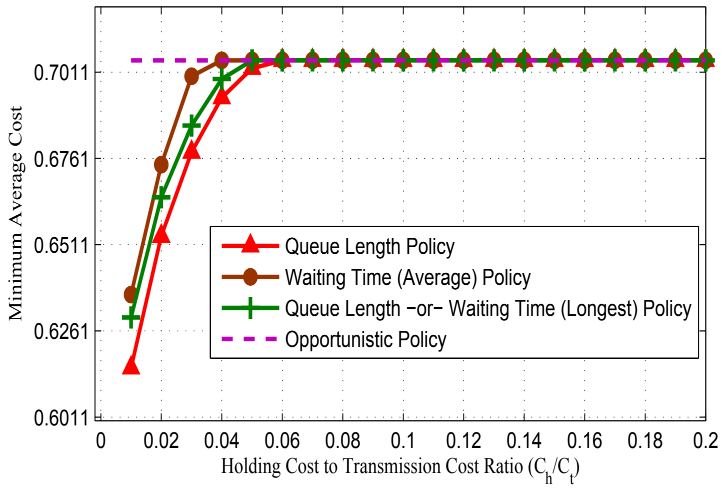

Fig. 16. Comparison of threshold policies using bursty Gilbert-Elliot sources ($p_{on} = p_{off} = 0.9$)

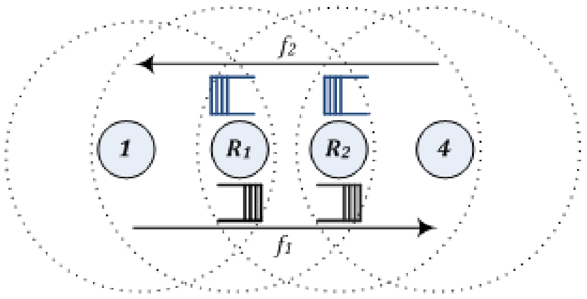

Fig. 17. Line network with two intermediate nodes

C. Extending to a Line Network

Our final model is that of a line network with 4 nodes (Figure 17). Here, the departures from one queue are the arrivals into queue at the next node and so there is a high degree of correlation between queues at neighboring nodes. It is highly difficult to characterize the arrival process into a queue whose input is packets from another queue. We would like to test whether the queue-length-based policy is near-optimal even when the arrival processes are significantly different from Bernoulli. As seen in Figures 18 and 19, the queue length policy is found to perform well in this case too.

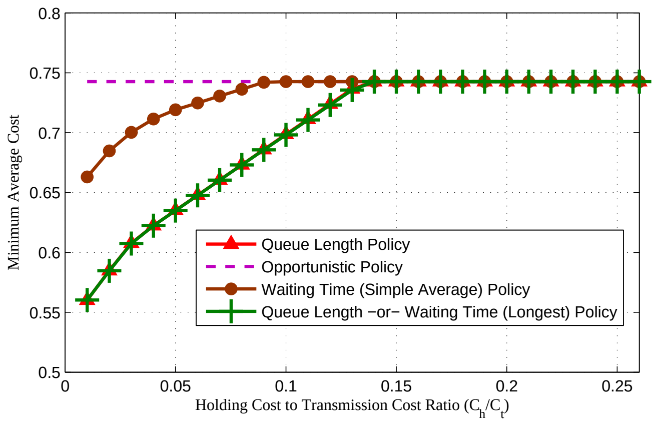

Fig. 18. Comparison of different policies in a line network with two intermediate nodes and two Bernoulli flows with mean arrival rates $(0.5, 0.5)$

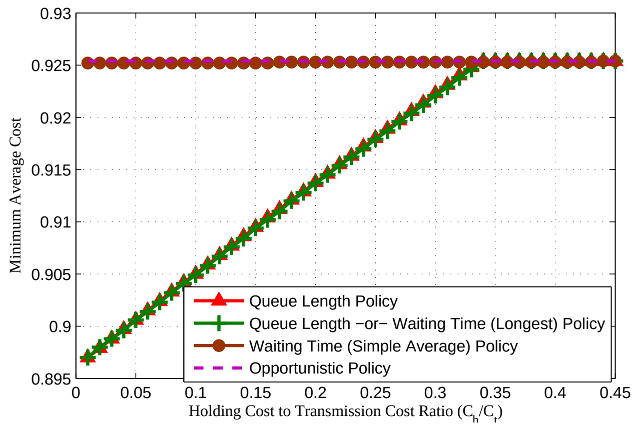

Fig. 19. Comparison of different policies in a line network with two intermediate nodes and two Bernoulli flows with mean arrival rates $(0.9, 0.3)$

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

We looked into the algorithms that explore the delicate tradeoff between waiting and transmitting using network coding. We started with the idea of exploring the whole space of history dependent policies, but showed step-by-step how we could move to simpler regimes, finally culminating in a stationary deterministic queue-length threshold based policy. The policy is attractive because its simplicity enables us to characterize the thresholds completely, and we can easily illustrate its performance on multiple networks. We showed by simulations how the performance of the policy is optimal for the Bernoulli arrival scenario in a simple 3-node relay network model we considered, and how it also does well in other situations such as for bursty arrivals and for line networks. Our results also have some bearing on the general problem of queuing networks with positive externalities that can be explored further.

A natural extension is to consider multiple arrivals in a time slot, multiple number of transmissions in a slot as well as time slots not being of equal lengths. That needs some explanation. Let the time line be divided into slots alternating between mega slots and mini slots. A mini slot is when the transmitter is "scheduled" to transmit packets and a mega slot is the time for the next scheduled mini-slot. Assume that a scheduled slot is of fixed duration and a maximum of a fixed number of packets can be transmitted (individually or coupled). If the packet arrivals are according to a Poisson process (and any arrivals during a mini slot cannot be transmitted in the same mini slot), then the system can be observed at the beginning of a mini slot. We believe that this system can also be modeled as a Markov chain, may be as a semi-Markov decision process. Next, we need to see if the threshold policy is optimal here too.

Another extension will be to look into larger networks, especially line networks with more than one intermediate node to start with. Contrasting to our model where the decision made at an intermediate node solely depends upon the information that can be collected locally, in line network model as there will be correlation between queues at neighboring nodes, a certain degree of coordination is required among neighboring nodes in order to design a distributed controller. In that case, MDP may not be a viable model for analysis because of dimensionality issues and it will be worth looking into alternative techniques like partially observable MDP, approximate dynamic programming etc.

REFERENCES

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

[2] J. Le, J.C.S. Lui, and D.-M. Chiu, "Dcar: Distributed coding-aware routing in wireless networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 4, pp. 596 –608, 2010.

[3] S. Sengupta, S. Rayanchu, and S. Banerjee, "Network coding-aware routing in wireless networks," *IEEE/ACM Transactions on Networking*, vol. 18, no. 4, pp. 1158 –1170, 2010.

[4] M. Effros, T. Ho, and S. Kim, "A tiling approach to network code design for wireless networks," in *Proc. IEEE Information Theory Workshop*, Punta del Este, Uruguay, 2006, pp. 62–66.

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

[6] R. Koetter and M. Medard, "An Algebraic Approach to Network Coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782 – 795, 2003.

[7] S. Katti, H. Rahul, D. Katabi, W. Hu, M. Médard, and J. Crowcroft, "XORs in the Air: Practical Wireless Network Coding," in *Proc. of ACM SIGCOMM*, Pisa, Italy, 2006, pp. 243–254.

[8] S.Chachulski, M.Jennings, S.Katti, and D.Katabi, "Trading structure for randomness in wireless opportunistic routing," in *Proc. of ACM SIGCOMM*, New York, 2007, pp. 169–180.

[9] Y.E. Sagduyu and A. Ephremides, "Cross-layer optimization of MAC and network coding in wireless queueing tandem networks," *IEEE Transactions on Information Theory*, vol. 54, no. 2, pp. 554–571, 2008.

[10] U.Kozatzy H.Seferogluy, A.Markopoulouy, "Network coding-aware rate control and scheduling in wireless networks," in *Proc. IEEE International Conference on Multimedia and Expo*, New York, 2009, pp. 1496–1499.

[11] V. Reddy, S. Shakkottai, A. Sprintson, and N. Gautam, "Multipath Wireless Network Coding: A Population Game Perspective," in *Proc. IEEE INFOCOM*, March 2010, pp. 1936 –1944.

[12] M.L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, New York: John Wiley and Sons, 1994.

[13] S.M. Ross, *Introduction to Stochastic Dynamic Programming*, New York: Academic Press, 1994.

[14] D.P. Bertsekas, *Dynamic Programming: Deterministic and Stochastic Models*, Englewood Cliffs, NJ: Prentice-Hall, 1987.

[15] B. Hajek, "Optimal control of two interacting service stations," *IEEE Transactions on Automatic Control*, vol. 29, no. 6, pp. 491–499, June 1984.

[16] 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.

[17] Y. Serin and V.G. Kulkarni, "Markov decision processes under observability constraints," *Mathematical Methods of Operations Research.*, vol. 61, pp. 311–328, 2005.

[18] John E. Hopcroft and Richard M. Karp, “A n5/2 algorithm for maximum matchings in bipartite,” in *12th Annual Symposium on Switching and Automata Theory*, East Lansing, MI, USA, 1971, pp. 122–125.

[19] Harold W. Kuhn, “The Hungarian method for the assignment problem,” *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955.

VITA

Solairaja Ramasamy did his master's in computer engineering and graduated from Texas A&M University in December 2010. He received his undergraduate degree, Bachelors of Technology *(Major: Information Technology),* in May 2005 from the College of Engineering, Guindy, Anna University, Chennai, India. Prior to his master's, he was working as a programmer in Infosys Technologies Ltd, India.

He may be contacted through email: *r.solairaja@gmail.com*. The mailing address is,

Solairaja Ramasamy,
Department of Electrical and Computer Engineering,
331F WERC,
Texas A&M University,
College Station,
TX, USA 77843-3128.

The typist for this thesis was Solairaja Ramasamy.