Opportunities for Network Coding: To Wait or Not to Wait

Yu-Pin Hsu*, Navid Abedini*, Solairaja Ramasamy*, Natarajan Gautam†, Alex Sprintson* and Srinivas Shakkottai*
*Dept. of ECE, Texas A&M University
†Dept. of ISE, Texas A&M University
Email: {yupinhsu, novid_abed, solairaja, gautam, spalex, sshakkot}@tamu.edu

***Abstract*—It has been well established that reverse-carpooling based network coding can significantly improve the efficiency of multi-hop wireless networks. However, in a stochastic environment when there are no opportunities to code because of packets without coding pairs, should these packets wait for a future opportunity or should they be transmitted without coding? To help answer that question we formulate a stochastic dynamic program with the objective of minimizing the long-run average cost per unit time incurred due to transmissions and delays. In particular, we develop optimal control actions that would balance between costs of transmission against those of delays. In that process we seek to address a crucial question: what should be observed as the state of the system? We analytically show that just the queue lengths is enough if it can be modeled as a Markov process. Subsequently we show that a stationary policy based on queue lengths is optimal and describe a procedure to find such a policy. We further substantiate our results with simulation experiments for more generalized settings.**

I. Introduction

In recent years, there has been a growing interest in the applications of network coding techniques in wireless communication networks. It was shown that the network coding technique can result in significant improvements in the performance of multi-hop wireless networks. For example, consider a wireless network coding scheme depicted in Figure 1(a). Here, wireless nodes n_1 and n_2 need to exchange packets x_1 and x_2 through a relay node. A simple *store-and-forward* approach needs four transmissions. However, 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 broadcast to nodes 1 and 2 simultaneously. Nodes n_1 and n_2 can then decode this coded packet to obtain the packets they need.

Effros *et al*. [1] introduced the strategy of *reverse carpooling* that allows two information flows traveling in opposite directions to share a 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 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.

This work was supported in part by NSF grant CMMI-0946935, NSF grant CNS-0954153, and DTRA grant HDTRA1-09-1-0051.

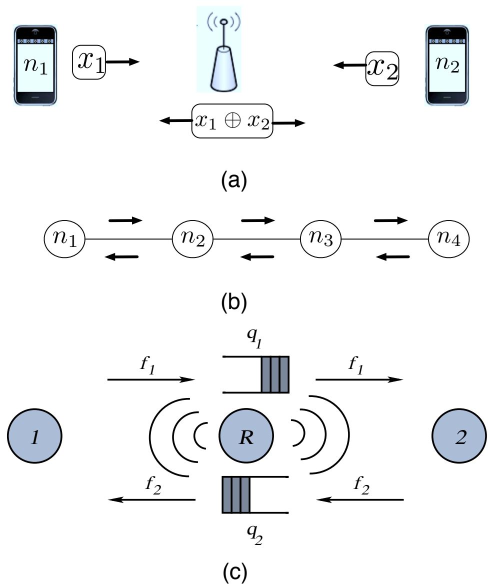


Fig. 1. (a) Wireless Network Coding (b) Reverse carpooling (c) 3-Node Relay Network.

In this paper, we focus on design and analysis of scheduling protocols 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.

Consider a relay node that transmits packets between two of its adjacent nodes that has flows in opposite directions, as depicted in Figure 1(c). 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. If both queues are not empty, then it can relay two packets from these queues by performing an XOR operation. However, what should the relay do if one of the queues have packets to transmit, 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? 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).

A. *Related Work*

Network coding research was initiated by seminal work by Ahlswede *et al*. [2] 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.

Network coding technique for wireless networks has been considered by Katabi *et al*. [3]. They proposed an architecture, referred to as COPE, which contains a special network coding layer between the IP and MAC layers. In our earlier work [4], we showed how to design coding-aware routing controllers that maximize coding opportunities (and hence reduce the number of transmissions) in multihop networks. However, in contrast to all the above literature the objective of this paper is to study the delicate tradeoff between transmission costs and waiting costs when network coding is an option.

B. *Main Results*

Our objective is to develop policies that yield a transmit/do not transmit decision at each time instant. We define our objective as the long-run average transmission cost plus waiting time cost on a per-unit-time basis. Thus, the relay can transmit and incur a transmission cost, or wait in the hope that a codable packet will arrive, which allow the transmission cost to be shared between two packets.

We first consider the case of a single relay, and assume that arrivals into both queues follow independent Bernoulli processes. We find that the optimal policy is a stationary queue-length threshold policy with one threshold for each queue at the relay, and whose action is simple: if a coding opportunity exists, code and transmit; else transmit a packet if the threshold for that queue is reached. We show how to find the optimal thresholds, and find exact expressions for the expected cost based on the stationary distribution of the Markov Chain when controlled by this policy.

II. System Overview

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 paper 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 relay that has the potential for network coding packets from flows in opposing directions. We will revert to the entire network only in Section IV.

A. *Scenario from a Relay's Perspective*

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

Notice that at the end of a slot, the relay can transmit a maximum of one packet. 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 relay 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 the relay node faces one of three kinds 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.

B. *Markov Decision Process Model*

To develop a strategy for the relay 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 in queue i 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 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,

$$C(Y_n, A_n) = C_h([Y_n^1 - A_n]^+ + [Y_n^2 - A_n]^+) + C_t A_n \quad (1)$$

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

$$V(u) = \lim_{N \to \infty} \frac{1}{N+1} E_u \left[\sum_{n=0}^{N} C(Y_n, A_n) | Y_0 = (0,0) \right] \quad (2)$$

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 $V(u)$. For that we first describe the probability law for our MDP and then in subsequent section 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 following in terms of $P_a(Y_n, Y_{n+1})$, the transition probability from state Y_n to Y_{n+1} associated with action $a \in \{0, 1\}$.

$$\begin{cases} P_1\big((i,j), ([i-1]^+, [j-1]^+)\big) = \hat{p}_1 \\ P_1\big((i,j), (\max(i,1), [j-1]^+)\big) = \hat{p}_2 \\ P_1\big((i,j), ([i-1]^+, \max(j,1))\big) = \hat{p}_3 \\ P_1\big((i,j), (\max(i,1), \max(j,1))\big) = \hat{p}_4 \\ P_0\big((i,j), (i,j)\big) = \hat{p}_1 \\ P_0\big((i,j), (i+1,j)\big) = \hat{p}_2 \\ P_0\big((i,j), (i,j+1)\big) = \hat{p}_3 \\ P_0\big((i,j), (i+1,j+1)\big) = \hat{p}_4 \end{cases} \quad (3)$$

where $\hat{p}_1 = (1-p_1)(1-p_2)$, $\hat{p}_2 = p_1(1-p_2)$, $\hat{p}_3 = (1-p_1)p_2$, and $\hat{p}_4 = p_1 p_2$. 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 $A_n = 1$.

III. Analysis

As we described in the previous section, our goal is to obtain the optimal policy u^* that minimizes $g(u)$, defined in (2). To that end, we first find the space of possible policies and then identify the optimal policy within this 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 [5]) 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.

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 $V(u)$ defined in (2). Further, one cannot find a policy which also uses waiting time information that would yield a better solution than $V(u^*)$.

Proof: See [6]. ■

B. Structure of the optimal policy

In the previous sections, we showed that there exists an optimal policy that does not include the waiting time in the state of the system. In this section we focus on queuelength-based policies and determine the structure of the optimal policy. In the MDP literature (see Sennott [7]), 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. Accordingly, for our MDP $\{(Y_n, A_n), n \geq 0\}$, the total expected discounted cost incurred by a policy θ is

$$V_{\theta,\beta}(i,j) = E_\theta \left[\sum_{n=0}^{\infty} \beta^n C(Y_n, A_n) | Y_0 = (i,j) \right]. \quad (4)$$

In addition, we define $V_\beta(i,j) = \min_\theta V_{\theta,\beta}(i,j)$ as well as $v_\beta(i,j) = V_\beta(i,j) - V_\beta(0,0)$.

Proposition 3: $V_\beta(i,j)$ is finite for all i, j, and discount factor β.

Proof: See [6]. ■

Proposition 3 implies that $V_\beta(i,j)$ satisfies the optimality equation [7],

$$V_\beta(i,j) = \min_{a \in \{0,1\}} [C_h([i-a]^+ + [j-a]^+) + C_t a + \beta \sum_{k,\ell} V_\beta(k,\ell) P_a\big((i,j),(k,l)\big)]. \quad (5)$$

The next lemma specifies the conditions that must be satisfied by the optimal stationary policy.

Lemma 4: (Sennott [7]) There exists a stationary policy that is optimal for the MDP $\{(Y_n, A_n), n \geq 0\}$ if the following conditions are satisfied: (i) $V_\beta(i,j)$ is finite for all i, j, and discount factor β; (ii) there exists a nonnegative N such that $v_\beta(i,j) \geq -N$ for all i, j, and β; and (iii) there exists a nonnegative $M_{i,j}$ such that $v_\beta(i,j) \leq M_{i,j}$ and

$$\sum_{k,l} P_a\big((i,j),(k,l)\big)M_{k,l} < \infty \tag{6}$$

for every i, j, β, and action a.

Using Lemma 4 we show next that the MDP defined in this paper has an optimal policy that is stationary.

Theorem 5: For the MDP $\{(Y_n, A_n), n \geq 0\}$, there exists a stationary policy u^* that minimizes $V(u)$ defined in (2).

Proof: See [6]. ■

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 a generic 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 6: (Serin and Kulkarni [8]) 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:

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

As described in Ross [9], 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 in fact 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.

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 structural properties then it is possible to search through the space of stationary deterministic policies and obtain the optimal one.

Theorem 7: For the MDP $\{(Y_n, A_n), n \geq 0\}$, the 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: See [6]. ■

C. Analysis: obtaining the optimal deterministic stationary policy

We have shown in the previous section that the optimal policy is stationary, deterministic and threshold type. The next step is to find it. Notice that we only need to consider the subset of deterministic stationary policies.

Theorem 8: The optimal thresholds L_1^* and L_2^* are

$$(L_1^*, L_2^*) = \arg\min_{L_1, L_2} C_t \tau(L_1, L_2) + C_h \lambda(L_1, L_2) \tag{7}$$

where

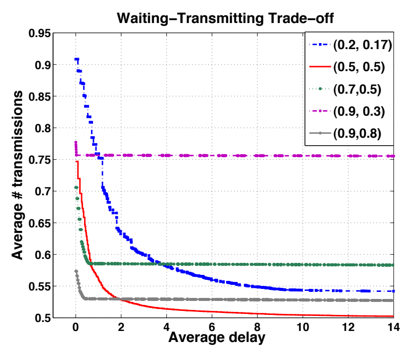
$$\begin{aligned} \tau(L_1, L_2) &= p_1 p_2 \pi_{0,0} + p_2 \sum_{i=1}^{L_1} \pi_{i,0} + p_1 \sum_{j=1}^{L_2} \pi_{0,j} + \\ & \quad p_1(1-p_2)\pi_{L_1,0} + p_2(1-p_1)\pi_{0,L_2} \quad (8) \\ \lambda(L_1, L_2) &= \sum_{i=1}^{L_1} i\pi_{i,0} + \sum_{j=1}^{L_2} j\pi_{0,j} \quad (9) \end{aligned}$$

for which

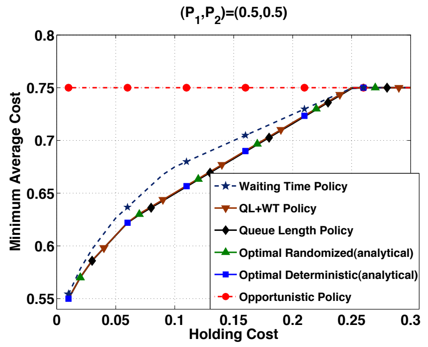
$$\begin{aligned} \pi_{0,0} &= \frac{1}{\left(\frac{1-\alpha^{L_1+1}}{1-\alpha}\right) + \left(\frac{1-1/\alpha^{L_2+1}}{1-1/\alpha}\right) - 1} \quad (10) \\ \pi_{i,0} &= \alpha^i \pi_{0,0} \quad (11) \\ \pi_{0,j} &= \pi_{0,0}/\alpha^j \quad (12) \\ \alpha &= \frac{(1-p_2)p_1}{(1-p_1)p_2} \quad (13) \end{aligned}$$

Proof: See [6]. ■

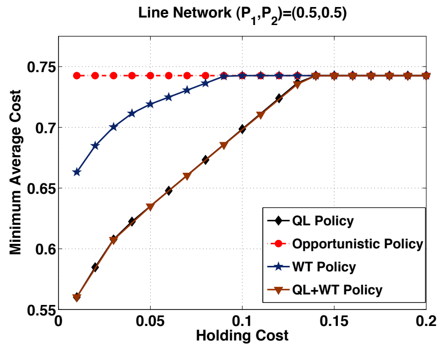
Whenever $C_h > 0$, it is relatively straightforward to obtain L_1^* and L_2^*. Since it costs C_t to transmit a packet and C_h for a packet to wait for a slot, it would be better to transmit a packet than make a packet wait for more than C_t/C_h slots.



(a)



(b)



(c)

Fig. 2. (a) Trade-off between average delay and number of transmissions in a single relay using queue-length threshold policy for different Bernoulli arrival rates (p_1, p_2), (b) Comparison of the minimum average cost (per packet) in a single relay with Bernoulli arrival rates $(0.5, 0.5)$, for different policies, (c) Comparison of different policies in a line network with two intermediate nodes and two Bernoulli flows with mean arrival rates $(0.5, 0.5)$

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.

IV. Numerical Experiments and Results

In this section we present several numerical results to demonstrate the analytical formulation as well as its extensions. We study the performance of a number of policies:

1) Opportunistic Coding: when a packet arrives, coding is performed if a compatible packet is available, otherwise transmission takes place immediately.
2) Queue-length-based threshold: a Stationary Deterministic (SD) policy that our analysis suggests it should be optimal for the Bernoulli case.
3) Randomized-Queuelength-based threshold: a Stationary policy that Randomizes (SR) over deterministic policies. We expect that it would not perform any better than deterministic queue-length-based policies.
4) Queue-length-plus-Waiting-time-based thresholds: a History Dependent policy (HR) which is likely to give the best possible performance.
5) Waiting-time-based thresholds: an HR policy that we create for the purpose of comparison to illustrate that history on its own is only of limited value.

We simulate these policies on two different cases: (i) the single relay with Bernoulli arrivals (Figures 2(a) and 2(b)) and (ii) a line network with 4 nodes, in which the sources are Bernoulli (Figure 2(c)). Note that in this case, since the departures from one queue determine the arrivals into the other queue, the arrival processes are significantly different from Bernoulli. Our simulations is done in Java and for each scenario we report the average results of 10^5 iterations.

Our numerical studies illustrate that, as expected, a deterministic queue-length based policy is optimal for different network scenarios. The results are intriguing as they suggest that achieving a near-perfect tradeoff between waiting and transmission costs is possible using simple policies; and coupled with optimal network-coding aware routing policies like the one in our earlier work [4], have the potential to exploit the positive externalities that network coding offers.

V. Conclusion

In this paper we developed 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 simulation how the performance of the policy is optimal in the Bernoulli arrival scenario, and how it also does well in other situations such as for line networks. Our results also have some bearing on the general problem of queuing networks with shared resources that we will explore in the future.

References

[1] M. Effros, T. Ho, and S. Kim, “A Tiling Approach to Network Code Design for Wireless Networks,” *Proc. of IEEE Information Theory Workshop (ITW)*, pp. 62–66, 2006.

[2] 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.

[3] S. Katti, H. Rahul, D. Katabi, W. H. M. Médard, and J. Crowcroft, “XORs in the Air: Practical Wireless Network Coding,” *Proc of ACM Special Interest Group on Data Communication (SIGCOMM)*, 2006.

[4] V. Reddy, S. Shakkottai, A. Sprintson, and N. Gautam, “Multipath Wireless Network Coding: A Population Game Perspective,” *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, 2010.

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

[6] Y. P. Hsu, N. Abedini, S. Ramasamy, N. Gautam, A. Sprintson, and S. Shakkottai, “Opportunities for Network Coding: To Wait or Not to Wait,” *Technical Report*, 2011. [Online]. Available: http://arxiv.org/abs/1105.4143

[7] L. Sennott, “Average Cost Optimal Stationary Policies in Infinite State Markov Decision Processes with Unbounded Costs,” *Operations Research*, vol. 37, pp. 626–633, 1989.

[8] Y. Serin and V. Kulkarni, “Markov Decision Processes Under Observability Constraints,” *Math. Meth. Oper. Res.*, vol. 61, pp. 311–328, 2005.

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