Network Coding-Aware Queue Management for TCP Flows over Coded Wireless Networks

Hulya Seferoglu, Athina Markopoulou
EECS Dept, University of California, Irvine
{hseferog, athina}@uci.edu

***Abstract*—We are interested in unicast traffic over wireless networks that employ constructive inter-session network coding, including single-hop and multi-hop schemes. In this setting, TCP flows do not fully exploit the network coding opportunities due to their bursty behavior and due to the fact that TCP is agnostic to the underlying network coding. In order to improve the performance of TCP flows over coded wireless networks, we take the following steps. First, we formulate the problem as network utility maximization and we present a distributed solution. Second, mimicking the structure of the optimal solution, we propose a network-coding aware queue management scheme (NCAQM) at intermediate nodes; we make no changes to TCP or to the MAC protocol (802.11). We demonstrate, via simulation, that NCAQM significantly improves TCP performance compared to TCP over baseline schemes.**

***Index Terms*—Network coding, wireless networks, congestion control, transport protocol design, queue management.**

I. Introduction

Wireless environments naturally lend themselves to network coding, thanks to the inherent broadcast and overhearing capabilities of the wireless medium. We are particularly interested in wireless mesh networks that employ constructive network coding schemes (such as COPE [1] and BFLY [6]), to mention some concrete examples). We consider unicast flows (particularly TCP, which is the dominant traffic type today) transmitted over such coded wireless networks.

In this setting, it has been demonstrated that network coding can significantly increase throughput [1], [2]. However, it has also been observed [1] that TCP does not exploit the full potential of the underlying network coding, mainly due to its bursty behavior. Rate mismatch between flows can significantly reduce the coding opportunities, as there may not be enough packets from different flows at intermediate nodes to code together. One possible solution is to artificially delay packets at intermediate nodes [3], until more packets arrive and can be coded together. However, the throughput increases with small delay (due to more coding opportunities), but decreases with large delay (which reduces the TCP rate); the optimal delay depends on the network topology and the background traffic and also may change over time. Thus, in many practical networking scenarios, introducing delay at intermediate nodes is not practical.

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

First, we formulate congestion control for unicast flows over wireless networks with inter-session network coding within the network utility maximization (NUM) framework [4], [5]. We assume that a known constructive network coding scheme is deployed in a wireless mesh network; examples include COPE [1] for one-hop network coding and BFLY [6] for two-hop network coding. The optimal solution of the NUM problem decomposes into several parts, each of which has an intuitive interpretation, such as rate control, queue management, and scheduling.

Second, motivated by the analysis, we propose modifications to congestion control mechanisms, so as to mimic the optimal solution of the NUM problem and to fully exploit the potential of network coding. It turns out that the optimal solution dictates minimal and intuitive implementation changes. We propose a network coding-aware queue management scheme at intermediate nodes (NCAQM), which stores coded packets and drops packets based on both congestion state and network coding. We note that the queues at intermediate nodes, which are already used for network coding, are a natural place to implement such changes with minimal implementation cost. In contrast, we do not propose any practical modifications to TCP or MAC (802.11) protocols, which significantly simplifies practical deployment of our proposal. Finally, we evaluate our proposal via simulation in GloMoSim [23] and we show that TCP over NCAQM significantly outperforms TCP over baseline schemes (*e.g.*, doubles the throughput improvement in some scenarios), and achieves near-optimal performance.

The rest of the paper is organized as follows. Section II discusses related work. Sections III-VI focus on wireless networks with one-hop network coding: Section III presents the system model; Section IV presents the optimization problem and solution; Section V presents the design of our network coding-aware queue management scheme (NCAQM); Section VI presents simulation results. Section VII extends our framework to multi-hop network coding. Section VIII concludes the paper. Appendix A presents numerical results for the convergence of the solution.

II. Related Work

This paper builds on top of constructive network coding schemes in wireless mesh networks. We rely on such a given scheme to provide the available coded and uncoded flows to higher layers. We then seek to optimize the treatment of these flows at the end-points and/or at intermediate nodes so as to maximize network coding opportunities.

COPE and follow-up work. COPE [1] is a constructive network coding scheme for one-hop network coding across unicast sessions. Our framework can also consider any other constructive scheme for inter-session NC, such as BFLY [6] or tiling approaches [7]. COPE has generated a lot of interest in the research community. Some researchers tried to model and analyze COPE [13], [15], [17]. Some others proposed new coded wireless systems, based on the idea of COPE [16], [6]. Zhao and Medard tried to explain and improve COPE's performance by looking at its interaction with MAC fairness [14]. We note that the authors of COPE had noticed the problem with TCP performance over COPE. As discussed in the introduction, [3] addressed the problem of rate mismatch between flows that are coded together, by delaying packets. Here, we take a different approach and we create coding opportunities via queue management and congestion control. More specifically, we aim at improving TCP performance over COPE by complementing it with a network coding-aware queue management scheme (NCAQM).

NUM in coded systems. Our analysis falls within the classic framework of network utility maximization (NUM) [5]. A significant body of work has looked at the joint optimization of intra- or inter-session NC of unicast flows. For example, in [8], minimum cost multicast over network coded wireline and wireless networks was studied. This work was extended for rate control in [9] for wireline networks. The rate region of multicast flows when network coding is used is studied in [10], [11]. The most closely related to this paper are resource allocation problems for unicast flows. For example, rate control, routing, and scheduling for generation-based intra-session network coding over wireless networks is considered in [12]. Optimal scheduling and optimal routing for COPE are considered in [13] and [17], respectively. Network utility maximization is used in [18] for end-to-end pairwise inter-session network coding. Energy efficient opportunistic inter-session network coding over wireless are proposed in [19], following a node-based NUM formulation and its solution based on back-pressure. A linear optimization framework for packing butterflies is proposed in [20].

Compared to prior NUM problems in coded networks, we focus on the congestion control problem for multiple unicast flows over wireless with a given inter-session network coding scheme. The most similar formulation is [9], but for intra-session network coding.

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

Comparison to our prior work. This paper is an improved and extended version of our conference paper that was presented in NetCod 2010 [25]. It includes significantly extended sections on simulations of performance (Sections VI and VII) as well as new numerical results on the convergence (Appendix A) of our schemes. It also extends the framework from one-hop to multi-hop network coding (Section VII).

In another piece of recent work [24], we studied a related but orthogonal aspect: we added intra-session redundancy to inter-session network coding in order to deal with wireless losses and to eliminate the need to know the state of the neighbors. In contrast, in this paper we consider only inter-session coding and we focus on the interaction between local (coding and queue management) and end-to-end (TCP) schemes, which was out of the scope of [24].

III. System Model

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

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

Due to the shared nature of the wireless media, transmission over different hyperarcs may interfere with each other. We consider the protocol model of interference [21], according to which, each node can either transmit or receive at the same time and all transmissions in the range of the receiver are considered as interfering. Given a hypergraph $\mathcal{H}$, we can construct the conflict graph $\mathcal{C} = (\mathcal{A}, \mathcal{I})$, whose vertices are the hyperarcs of $\mathcal{H}$ and edges indicate interference between hyperarcs. A clique $\mathcal{C}_q \subseteq \mathcal{A}$ consists of several hyperarcs, at most one of which can transmit at the same time without interference.

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

Routing: We consider that each flow $s \in \mathcal{S}$ follows a single path $\mathcal{P}_s \subseteq \mathcal{N}$ from the source to the destination. This path is

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

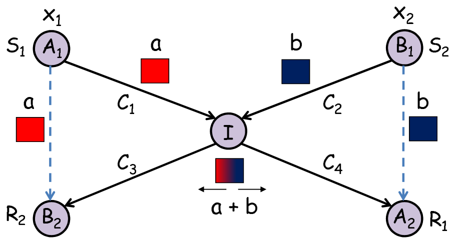


Fig. 1. "X topology". Source S_1 transmits a flow with rate x_1 to receiver R_1 and source S_2 transmits a flow with rate x_2 to receiver R_2, over the intermediate node I. A_1 and B_1 transmit their packets a and b, in two time slots, and node I receives them. Furthermore, A_2 overhears b and B_2 overhears a, because $A_1 - B_2$ and $B_1 - A_2$ are in the same transmission range and they can overhear each other. In the next time slot, I broadcasts the network coded packet, $a \oplus b$ over hyperarc $(I, \{A_2, B_2\})$. Since A_2 and B_2 have overheard b and a, they can decode their packets a and b, respectively.

pre-determined by a routing protocol, *e.g.*, OLSR or AODV, and given as input to our problem. However, note that several different hyperarcs may connect two consecutive nodes along the path. We set an indicator function $H_{i,\mathcal{J}}^{s,k} = 1$ if flow s is transmitted through hyperarc $(i, \mathcal{J})$ using network code $k \in \mathcal{K}_{i,\mathcal{J}}$. Otherwise, $H_{i,\mathcal{J}}^{s,k} = 0$.

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

IV. Optimal Congestion Control

A. Problem Formulation

The objective is to maximize the total utility function, by appropriately selecting: the flow rates x_s at sources $s \in \mathcal{S}$; their traffic splitting parameter $\alpha_h^{s,k}$ (following the terminology of [9]) into network codes $k \in \mathcal{K}_h$ over hyperarc h at intermediate nodes; and the percentage of time τ_h each hyperarc is used:

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

The first constraint is the capacity constraint. $H_h^{s,k}\alpha_h^{s,k}x_s$ indicates the part of flow rate x_s allocated to the k-th network code over hyperarc h. The rate of the k-th network code is the maximum rate among flows $s \in \mathcal{S}_k$ coded together in code k: $\max_{s\in\mathcal{S}_k}\{H_h^{s,k}\alpha_h^{s,k}x_s\}$ [8]. Different network codes $k \in \mathcal{K}_h$ over h share the available capacity $R_h\tau_h$, where R_h is the transmission capacity of h; since h is a set of links, R_h is the minimum: $R_h = \min_{j\in h(\mathcal{J})}\{R_{i,j}\xi_{i,j}\}$ where $R_{i,j}$ is the capacity of link (i,j), and $\xi_{i,j}$ is the probability of successful transmission over link (i,j). The second constraint is the flow conservation constraint: at every node i on the path $\mathcal{P}_s$ of source s, the sum of $\alpha_h^{s,k}$ over all network codes and hyperarcs should be equal to 1. Indeed, when a flow enters a particular node i, it can be transmitted to its next hop j as part of different network coded and uncoded flows. The third constraint is due to interference. As mentioned, τ_h is the percentage of time h is used. Its sum over all hyperarcs in a clique should be less than an over-provisioning factor, $\gamma \le 1$, because all hypearcs in a clique interferes, and should time share the medium.

B. Solution

By relaxing the capacity constraint in Eq. (1), we get the Lagrangian:

$$\begin{aligned}
&L(\boldsymbol{x},\boldsymbol{\alpha},\boldsymbol{\tau},\boldsymbol{q}) = \\
&\sum_{s\in\mathcal{S}} U_s(x_s) - \sum_{h\in\mathcal{A}} q_h \left(\sum_{k\in\mathcal{K}_h} \max_{s\in\mathcal{S}_k}\{H_h^{s,k}\alpha_h^{s,k}x_s\} - R_h\tau_h \right)
\end{aligned} \tag{2}$$

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

The Lagrange function in Eq. (2) is not strictly concave in $m_h^{s,k}$ and this causes oscillation in its solution. We use the proximal method [22] to eliminate oscillations;

$$\begin{aligned}
&\max_{\boldsymbol{m}} \sum_{s\in\mathcal{S}_k} (H_h^{s,k}\alpha_h^{s,k}x_s m_h^{s,k} - c(m_h^{s,k} - \mu_h^{s,k})^2) \\
&\text{s.t.} \sum_{s\in\mathcal{S}_k} m_h^{s,k} = 1,
\end{aligned} \tag{3}$$

where c is a constant and $\mu_h^{s,k}$ is an artificial variable of the proximal method [22]. Its value is set to $m_h^{s,k}$ periodically. Let $(m_h^{s,k})^*$ be the solution to this problem.

By rewriting the summation $\sum_{k\in\mathcal{K}_h} \sum_{s\in\mathcal{S}_k}$ as $\sum_{s\in\mathcal{S}} \sum_{k\in K_h|s\in\mathcal{S}_k}$, the Lagrange function in Eq. (2) can be expressed as: $L(\boldsymbol{x},\boldsymbol{\alpha},\boldsymbol{\tau},\boldsymbol{q}) = \sum_{h\in\mathcal{A}} q_h R_h \tau_h + \sum_{s\in\mathcal{S}} \left(U_s(x_s) - x_s \sum_{h\in\mathcal{A}} \sum_{k\in\mathcal{K}_h|s\in\mathcal{S}_k} q_h H_h^{s,k}\alpha_h^{s,k}(m_h^{s,k})^* \right)$. Now, we can decompose the Lagrangian into the following intuitive problems: rate control, traffic splitting, scheduling, and parameter update (queue management).

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

$$x_s = (U_s')^{-1}\left(\sum_{h\in\mathcal{A}}\sum_{k\in\mathcal{K}_h|s\in\mathcal{S}_k} q_h H_h^{s,k}\alpha_h^{s,k}(m_h^{s,k})^*\right), \quad (4)$$

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

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

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

$$\begin{aligned}\min_{\boldsymbol{\alpha}} & \sum_{h(J)|h\in\mathcal{A}}\sum_{k\in K_h|s\in\mathcal{S}_k} q_h H_h^{s,k}(m_h^{s,k})^*\alpha_h^{s,k}\\ \text{s.t.} & \sum_{h(J)|h\in\mathcal{A}}\sum_{k\in K_h|s\in\mathcal{S}_k} \alpha_h^{s,k} = 1, \ \forall i \in \mathcal{P}_s \end{aligned} \quad (5)$$

Similarly to Eq. (3), we also use the proximal method [22] to solve the optimization problem in Eq. (5).

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

$$\begin{aligned}\max_{\boldsymbol{\tau}} & \sum_{h\in\mathcal{A}} q_h R_h \tau_h\\ \text{s.t.} & \sum_{h\in\mathcal{C}_q} \tau_h \le \tau, \ \forall \mathcal{C}_q \subseteq A. \end{aligned} \quad (6)$$

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

$$q_h(t+1) = \{q_h(t) + c_t[\sum_{k\in\mathcal{K}_h}\max_{s\in\mathcal{S}_k}\{H_h^{s,k}\alpha_h^{s,k}x_s\} - R_h\tau_h]\}^+ \quad (7)$$

where t is the iteration number, c_t is a small constant, and the $^+$ operator makes the Lagrange multipliers positive. q_h can be interpreted as the queue size at hyperarc $\forall h \in \mathcal{A}$. Indeed, in Eq. (7), q_h is updated with the difference between the incoming $\sum_{k\in\mathcal{K}_h}\max_{s\in\mathcal{S}_k}\{H_h^{s,k}\alpha_h^{s,k}x_s\}$ and outgoing $R_h\tau_h$ traffic at h. Therefore, we call q_h the hyperarc-queue, or h-queue for brevity. We confirmed the convergence of q_h's via numerical calculations as seen in Appendix A.

V. Network Coding-Aware Implementation

In the previous section, we saw that the NUM problem decomposed into Eq. (4), Eq. (5), Eq. (6), Eq. (7), each of which has an intuitive interpretation. In this section, we mimic the properties of the optimal solutions to these problems and propose modifications to the corresponding protocols to make them network coding-aware. It turns out that only changes to queue management at intermediate nodes are crucial, while TCP and scheduling can remain intact. This makes our proposal amenable to practical deployment.

A. Queue Management at Intermediate Nodes (NCAQM)

1) Summary of Proposed Scheme: We refer to our *Network Coding-Aware Queue Management* scheme as NCAQM. NCAQM builds on and extends COPE [1]. Its goal is to interact with TCP congestion control in such a way that it matches the rates of TCP flows coded together and thus increases network coding opportunities. It achieves this goal through the following minimal changes at intermediate nodes. First, NCAQM stores coded packets in the output queue $\mathcal{Q}_i$, as opposed to COPE that stores uncoded packets. Second, NCAQM maintains state per hyperarc queue q_h and per network code transmitted over each hyperarc $k \in \mathcal{K}_h$; this is feasible in the setting of wireless mesh with limited number of flows. Third, during congestion, packets are dropped from the flow that has the largest number of packets, where this number is computed only over h-queues where the flow is dominant. Consider several flows coded together in the same code: the rate of the dominant flow is the rate of the code; and dropping from the dominant flow matches the rates, as desired. We note that intermediate nodes do already network coding operations and can be naturally extended to implement these changes.

2) Detailed Description of Proposed Scheme:

Maintaining Queues: In [1], a wireless node i stores all packets uncoded in a single output queue $\mathcal{Q}_i$ and takes decisions at every transmission opportunity about whether to code some of these packets together or not. In contrast, we propose to network code packets, if an opportunity exists, at the time we store them in the queue. Motivated by the fact that Lagrange multiplier (h-queue) q_h in Eq. (7) can be interpreted as the queue size at hyperarc h, we maintain h-queue virtually[2] for each hyperarc at every node, which keeps track of packets that are network coded and broadcast over h. The size of an h-queue is Q_h and how it is determined in practice will be explained later. Each node i maintains a single physical output queue, $\mathcal{Q}_i$, which stores all packets (coded and uncoded depending on the opportunities) passing through it.

Network Coding (Alg. 1): Motivated by the fact that the incoming traffic in Eq. (7) is the sum of the network coded flows over h, we code packets when they are inserted to output queues. If a network coding opportunity does not exist when the packet arrives at node i, we just store it in $\mathcal{Q}_i$ in a FIFO way. Periodically, Alg. 1 runs to check all packets in the queue for network coding.

Let $\mathcal{Q}_i = \{p_1, p_2, ..., p_l\}$ where p_1 is the first and p_l is the last packet in the queue; $l \le L$, where L is the buffer size, *i.e.*, the maximum number of packets that can be stored in $\mathcal{Q}_i$. First, p_1 is picked for network coding. Since

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

Algorithm 1 Network coding in output queue Q_i at node i

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

Algorithm 2 Packet dropping at node i during congestion

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

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

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

The traffic splitting parameters $\alpha_h^{s,k}$ are found through the optimization problem in Eq. (5). Through numerical calculations, we made the following observation: each $\alpha_h^{s,k}$ converges to the percentage of time that packets from flow s are transmitted with the k-th network code over h at node i. At each packet transmission, we calculate the probability that a network code k over hyperarc h can be used for flow s, over a time window. The average calculated over this window gives a heuristic estimate of the traffic splitting parameter, $\check{\alpha}_h^{s,k}$.

Packet Dropping (Alg. 2): When a node is congested, it decides which packet to drop. In order to eliminate the potential of rate mismatch between flows coded together, we propose that the node compares the number of all (coded and uncoded) packets of each flow, in queues where the flow is dominant ($m_h^{s,k} = 1$). This is motivated by the optimal rate control in Eq. (4). More specifically, for each flow s, we calculate $\Phi_i^s = \sum_{h(\mathcal{J})|h \in \mathcal{A}} Q_h \check{w}_h^s$, where $\check{w}_h^s = \sum_{k \in \mathcal{K}_h | s \in \mathcal{S}_k}$ and $H_h^{s,k}\check{\alpha}_h^{s,k}\check{m}_h^{s,k}$. Upon congestion, the Φ_i^s's are compared and a packet from the flow with the largest Φ_i^s is dropped, preferably the last uncoded packet. The choice of the last packet is to make it similar to DropTail. The choice of uncoded packet is so as to hurt only one flow, as opposed to several. If there is a tie in the Φ's between flows, one flow is randomly picked to drop a packet. If all packets from the selected flow are coded, a new coming packet(s) is dropped instead.

To estimate the dominance indicator $\check{m}_h^{s,k}$ needed in Alg. 2, we compute heuristically an estimate $\check{m}_h^{s,k}$ as follows. If $H_h^{s,k}\check{\alpha}_h^{s,k}Q_i^s < H_h^{s',k}\check{\alpha}_h^{s',k}Q_i^{s'}$ s.t. $\exists s' \in \mathcal{S}_k - \{s\}$, then $\check{m}_h^{s,k} = 0$. Otherwise, $\check{m}_h^{s,k} = (|\mathcal{S}_k^{max}|)^{-1}$ where $\mathcal{S}_k^{max} = \{s | s \in \mathcal{S}_k \wedge H_h^{s,k}\check{\alpha}_h^{s,k}Q_i^s = \max\{H_h^{s',k}\check{\alpha}_h^{s',k}Q_i^s \mid s' \in \mathcal{S}_k\}\}$.

B. Rate Control at the Sources

For logarithmic utility, we saw that the optimal rate control in Eq. (4) is $x_s = (\sum_{i \in \mathcal{P}_s} q_i^s)^{-1}$. q_i^s corresponds to the length of the network coded queue size of flow s at node i. The optimal rate x_s is inversely proportional to the sum of these queue sizes q_i^s across all nodes i on its path $\mathcal{P}_s$. This is essentially a generalization of standard optimal rate control [4], to account for network coding in the calculation of queue sizes.

When rate control is implemented, it is impractical to feed back to the source the full information $\sum_{i \in \mathcal{P}_s} q_i^s$, as required by the optimal control. Instead, when a queue is congested, a packet is dropped or marked [4]. The source uses this binary information as a signal to reduce its rate, mimicking the inverse relationship in the optimal control. The exact adaptation of the flow rate depends on the TCP version used. In the simulations, we used TCP-SACK without any modification. The only change we propose is the packet dropping scheme at the queue (Alg. 2), to take into account not only congestion but also network coding. Essentially, TCP still reacts to drops but these drops are caused when the flow is dominant in at least one network coded queue along the path.

Example 2: Let us re-visit the example in Fig. 1. There is only one network coded flow over $h = (I, \{A_2, B_2\})$ and assume that link transmission rates are the same. Then the two flows are always coded together and their traffic splitting parameters approach to 1. The network coded queue sizes are $\Phi_I^1 = Q_h\check{m}_h^1$ and $\Phi_I^2 = Q_h\check{m}_h^2$, where Q_h is the size of the h-queue for $h = (I, \{A_2, B_2\})$, and $\check{m}_h^1$ and $\check{m}_h^2$ are the dominance indicators for the two flows. Since Q_h is constant, Φ_I^1, Φ_I^2 depend on $\check{m}_h^1$ and $\check{m}_h^2$, *i.e.*, on which flow has more packets in the output queue. Upon congestion, a packet from the first is dropped if it has more packets in the queue. Then, S_1 will reduce its rate by transmitting less packets, while flow S_2 keeps increasing its rate, thus decreasing the probability that there is no packet from the second flow for coding at node

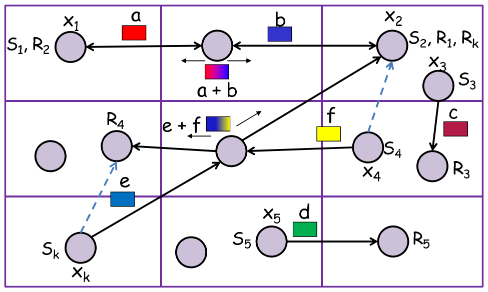


Fig. 3. Grid topology. Multiple unicast flows $S_1 - R_1$, $S_2 - R_2$, etc., meeting at different intermediate points.

I. More generally, the interaction of our queue management (NCAQM) mechanism and TCP tends to eliminate the rate mismatch of the flows coded together. □

C. Scheduling

The scheduling part in Eq. (6) has two parts: intra- and inter-scheduling that determine which packet to transmit from a node and which node should transmit, respectively. Both have difficulties in practice. Intra-scheduling causes packet reordering at TCP receivers. Inter-scheduling requires centralized knowledge and it is NP hard and hard to approximate [5]. Given these difficulties and our original goal to make minimal changes to protocols related to congestion control, we limit our proposed modifications to the queue management. We do not propose new scheduling and we use FIFO scheme for packet transmission and standard 802.11 as wireless MAC.

VI. Performance Evaluation

In this section, we evaluate the throughput of TCP over our proposed scheme (NCAQM) in various topologies and traffic scenarios. We compare it to TCP over the following baseline schemes: no network coding (noNC), which uses FIFO without network coding; COPE [1], which stores native packets in a FIFO and decides which packets to code together at each transmission opportunity; and the optimal control.

A. Simulation Setup

We used the GloMoSim simulator [23], which is well suited for wireless. We implemented from scratch the modules for one-hop network coding over wireless mesh networks (COPE) as well as for our proposed scheme (NCAQM).

1) Topologies: We simulated four illustrative topologies shown in Fig. 1, Fig. 2, and Fig. 3. In X and Alice-and-Bob topologies, shown in Fig. 1 and Fig. 2(a), two unicast flows $S_1 - R_1$ and $S_2 - R_2$ meet at intermediate node I. In the cross topology, shown in Fig. 2(b), four unicast flows S_1-R_1, S_2-R_2, S_3-R_3, and S_4-R_4 are transmitted via the relay I. In the wheel topology, shown in Fig. 2(c), multiple unicast flows such as $S_1 - R_1$, $S_2 - R_2$, $S_3 - R_3$, $S_4 - R_4$, and etc. are combined at the intermediate node I. Note that the wheel topology is the generalized version of the cross topology shown in Fig. 2(b). In all these topologies node I; (i) performs network coding, and (ii) is placed in the center of a circle with $90m$ radius over $200m \times 200m$ terrain and all other nodes are placed around the circle. Finally, we considered the grid topology shown in Fig. 3, in which nodes are distributed over a $300m \times 300m$ terrain, divided into 9 cells of equal size. 15 nodes are divided into sets consisting of 1 or 2 nodes and each set is assigned to a different cell. Nodes in a set are randomly placed within their cell. If both the transmitter and the receiver are in the same cell or in neighboring cells, there is a direct transmission; otherwise, a node in a neighboring cell acts as a relay. If there are more than one neighboring cells, one is chosen at random. In all topologies, a single channel is used for both uplink and downlink transmissions.

2) MAC: In the MAC layer, we simulated IEEE 802.11 with RTS/CTS enabled and with the following modifications for network coding. First, we need a broadcast medium, which is hidden by the 802.11 protocol. We used the pseudo-broadcasting mechanism of [1]: packets are XOR-ed in a single unicast packet, an XOR header is added for all nodes that should receive that packet, and the MAC address is set to the address of one of the receivers. A receiver knows whether a packet is targeted to it from the MAC address or the XOR header.

3) Wireless Channel: We used the two-ray path loss model and Rayleigh fading in Glomosim. We set the average loss rate to 15%. In our simulations 15% loss rate is medium loss rate, and residual loss rate after MAC re-transmissions is less than 1%.[3]

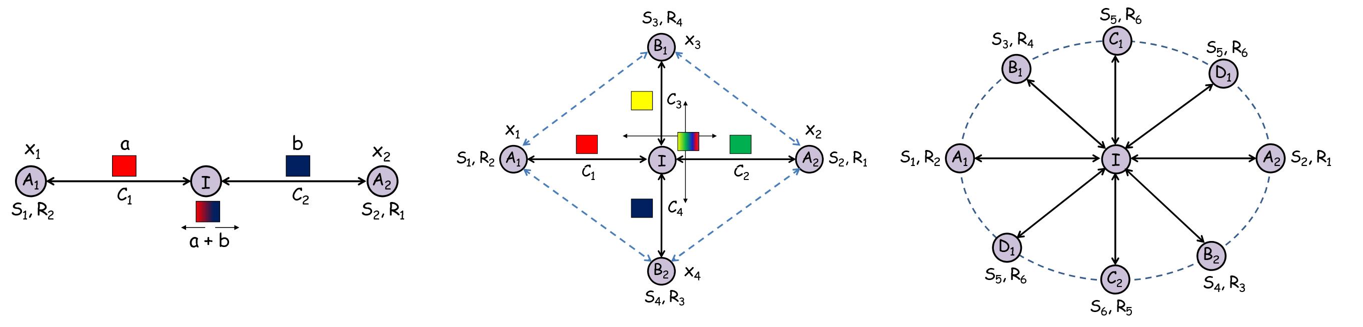
4) TCP Traffic: We consider FTP/TCP traffic on top of the wireless network. In the Alice-and-Bob, X, cross, and wheel topologies, TCP flows, between the pairs of nodes described above, start at random times within the first $5sec$ and live until the end of the simulation. In the grid topology, TCP flows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

B. Simulation Results

In this section, we present simulation results for the Alice-and-Bob, X, cross, wheel, and grid topologies. We compare to: (i) TCP over NCAQM (TCP+NCAQM), (ii) TCP over COPE (TCP+COPE), (iii) the optimal solution (optimal rate control in Eq. (4) working together with the optimal queue management in Eq. (7)). We report the average throughput of each scheme as % improvement over the throughput of the baseline TCP+noNC. In addition, we report transport level throughput. All throughput results reported in this section are averaged over $1min$ simulation duration first, then over 10 simulations with different seeds.

Table I presents the results for the following parameters: the

[3]When channel loss rate increases, there are two problems. First, the residual loss rate after MAC re-transmissions increases. Therefore, TCP is not able to utilize the medium effectively and benefit of network coding reduces. Second, network coding decision at intermediate nodes becomes erroneous, because intermediate nodes do not know which packets are overheard correctly. These issues are out of scope of this paper, and we have analyzed them separately in [24].



(a) Alice-and-Bob Topology (b) Cross Topology (c) Wheel Topology

Fig. 2. (a) Alice-and-Bob Topology. Two unicast flows, $S_1 - R_1$, and $S_2 - R_2$, meeting at intermediate node I. (b) Cross topology. Four unicast flows, $S_1 - R_1$, $S_2 - R_2$, $S_3 - R_3$, and $S_4 - R_4$, meeting at intermediate node I. (c) Wheel topology. Multiple unicast flows $S_1 - R_1$, $S_2 - R_2$, etc., meeting at intermediate node I. I opportunistically combine the packets and broadcast.

buffer size at each intermediate node is 10 packets[4]; the packet size is $500B$; the channel capacity is $1Mbps$. In this scenario, TCP+NCAQM has two advantages: (i) it stores network coded packets instead of the uncoded ones, thus uses the buffer more effectively, and (ii) it drops packets so that network coding opportunities increase. Thus, our scheme (TCP+NCAQM) significantly improves throughput as compared to TCP+COPE in all four topologies. It is also seen from the table that there is still a gap between our scheme and the optimal improvement due to the very limited buffer size for multiple flows at the relay. Yet, even in this challenging scenario, TCP+NCAQM significantly improves over TCP+COPE: it doubles the throughput improvement of TCP+COPE.

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

Fig. 4 presents the cumulative distributed function (CDF) of throughput improvement for the Alice-and-Bob, X, cross, and grid topologies and the same setup. The CDFs are calculated over 30 seeds. One can see that the CDF of TCP+NCAQM is shifted to significantly higher throughput levels compared to TCP+COPE in all four topologies. For example, TCP+NCAQM improves the throughput more than 20% and 40% in more than 60% of the realizations in Alice-and Bob and cross topologies, respectively. In contrast to Alice-and-Bob and cross topologies, we also observe that the CDF of TCP+NCAQM is shifted to higher throughput levels compared to the CDF of TCP+COPE in the cross and grid topologies. In the cross and grid topologies, it is possible to code more than two flows together, and when the number of flows coded together increases, the way that TCP+NCAQM uses buffers and balances the rates becomes more important. Thus, we see larger improvement in the cross and grid topologies.

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

TABLE I
AVERAGE THROUGHPUT IMPROVEMENT COMPARED TO NONC.

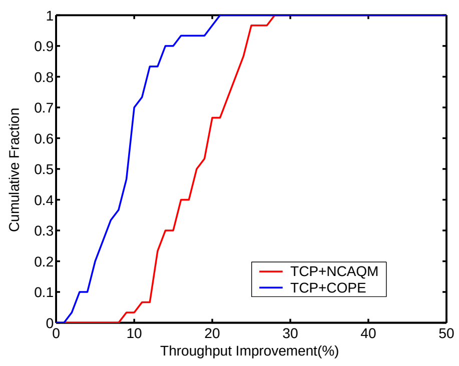
	Optimal	TCP+NCAQM	TCP+COPE
Alice-and-Bob Topology	33%	18%	8%
X Topology	33%	19%	9%
Cross Topology	60%	39%	21%
Grid Topology	-	35%	18%

Fig. 5 shows the average transport-level throughput versus the buffer size, for the Alice-and-Bob, X, cross, and grid topologies. Packet size is $500B$, and channel capacity is $1Mbps$. Our observations from Fig. 5 are in the following.

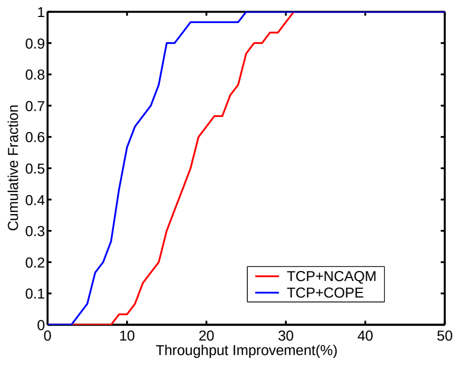
The throughput improvement of TCP+noNC for different buffer sizes is negligible in all topologies. The reason is that 10 packet buffer size is already matched to bandwidth-delay product (BDP) and TCP utilizes wireless medium effectively for almost all buffer sizes when network coding is not used (TCP+noNC).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

When buffer sizes are small, the improvement of TCP+NCAQM over TCP+noNC is significantly larger than that of TCP+COPE. This is for the same reason explained earlier: TCP+NCAQM stores network coded, instead of uncoded packets, thus using buffer more effectively, and it drops packets so that network coding opportunities increase. Thus, our scheme (TCP+NCAQM) significantly improves throughput as compared to TCP+COPE in all four topologies.

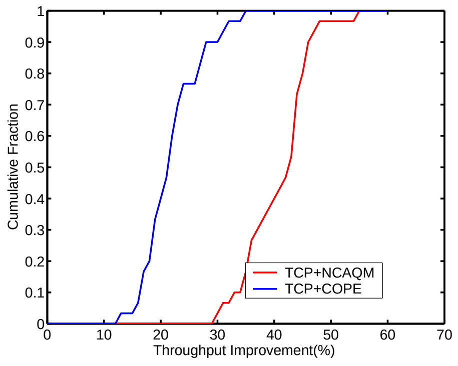
The throughput of TCP+COPE increases when buffer sizes increase, which is intuitively expected. The problem addressed



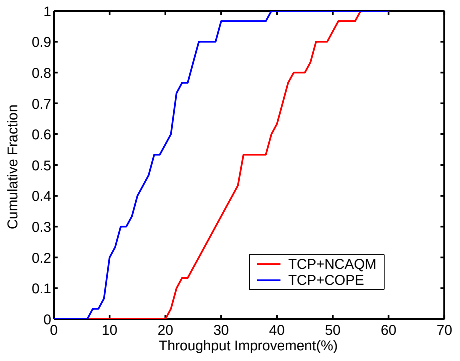
(a) Alice-and-Bob topology



(b) X topology



(c) Cross topology



(d) Grid topology

Fig. 4. Cumulative distribution function (CDF) of throughput improvement for Alice-and-Bob (shown in Fig. 2(a)), X (shown in Fig. 1), cross (shown in Fig. 2(b)), and grid (shown in Fig. 3) topologies. Buffer size is 10 packets, packet sizes are $500B$, and the channel capacity is $1Mbps$. The distributions are generated over 30 seeds.

in this paper was the mismatch between rates of flows coded together, due to the bursty nature of TCP, which reduces coding opportunities. However, when buffer sizes increase, there are more packets available in queues for coding. Thus, TCP+COPE exploits coding opportunities at larger buffers and its throughput increases. However, even at the large buffer sizes, TCP+NCAQM improves throughput more than TCP+COPE. For example, TCP+NCAQM improves throughput 7% more than TCP+COPE in X topology when buffer size is 50 packets. Fig. 5 demonstrates that our scheme is particularly beneficial in harsh buffer size conditions.

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

Fig. 6 shows the average transport-level throughput versus the number of flows in the wheel topology shown in Fig. 2(c). The buffer size is 30 packets, the packet size is $500B$, and channel capacity is $1Mbps$. One can see from the figure that the throughput of TCP+noNC reduces with increasing number of flows. This is expected, because when the number of flows increases, all flows share the same queue at the intermediate node I. As a result, the round trip time of each flow increases,

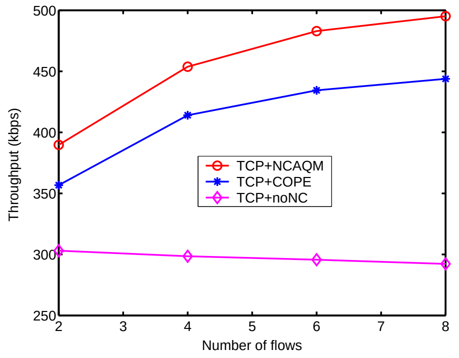
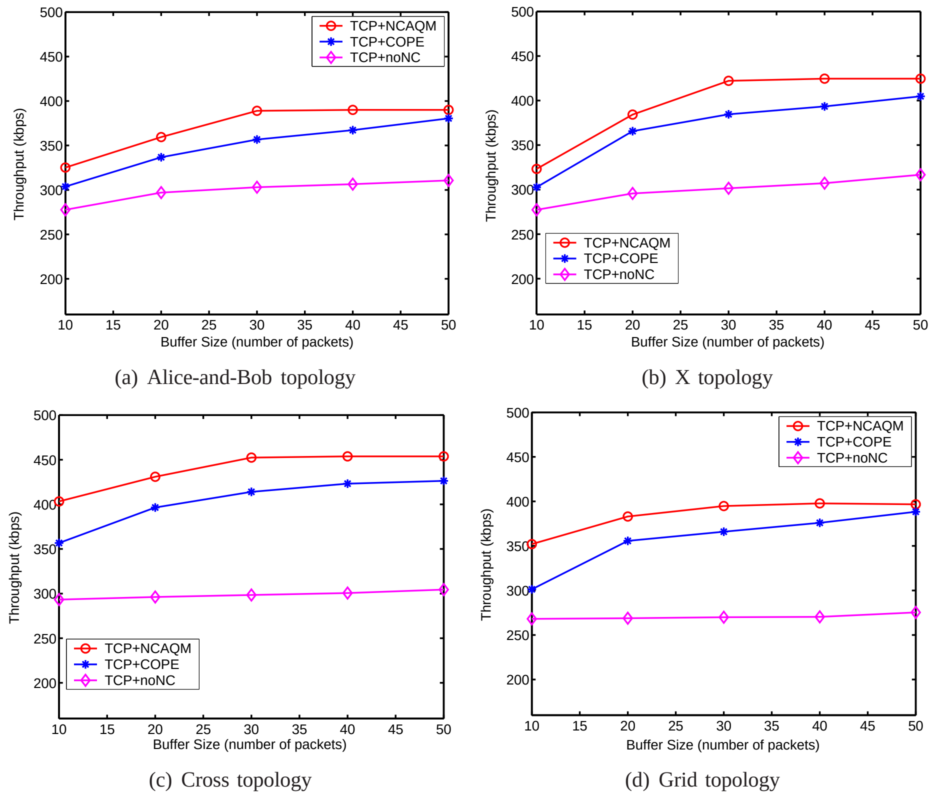


Fig. 6. Average throughput (averaged in $1min$ simulation first, then over 10 seeds) versus the number of flows in wheel topology shown in Fig. 2(c). Buffer size is 30 packets, packet size is $500B$, and the channel capacity is $1Mbps$.

and thus the TCP rate decreases. On the other hand, the throughput of TCP+NCAQM and TCP+COPE increases with the number of flows, because when the number of flows increases, there are more network coding opportunities and more packets can be combined together (*i.e.*, it is possible to combine 8 packets when the number of flows is 8). TCP+NCAQM significantly improves over TCP+COPE for all number of flows, especially when the number of flows is large. This is intuitive, because when the number of flows increases, network coding opportunities increases, and TCP+NCAQM exploits these opportunities effectively.

Fig. 7 presents the average transport-level throughput versus channel capacity for the Alice-and-Bob, X, cross, and grid



(a) Alice-and-Bob topology

(b) X topology

(c) Cross topology

(d) Grid topology

Fig. 5. Average throughput (averaged in $1min$ simulation first, then over 10 seeds) versus buffer size for Alice-and-Bob (shown in Fig. 2(a)), X (shown in Fig. 1), cross (shown in Fig. 2(b)), and grid (shown in Fig. 3) topologies. Packet size is $500B$, and channel capacity is $1Mbps$.

topologies. The buffer size is 30 packets, and the packet size is $500B$. One can see from the figure that when the channel capacity increases, the gap between TCP+NCAQM and TCP+COPE increases. Therefore, while the improvement of TCP+NCAQM over TCP+noNC increases with increasing channel capacity, it decreases for TCP+COPE. Namely, the improvement of TCP+NCAQM increases from 40% to 42%, while the improvement of TCP+COPE decreases from 27% to 16% in X the topology. The improvement of TCP+NCAQM is quite significant; more than double the improvement of TCP+COPE at $11Mbps$ channel capacity. The reason is that when the channel capacity increases, more packets share buffer at intermediate node. TCP+NCAQM can improve the throughput by using the shared buffers more effectively, and by dropping packets so as to increase network coding opportunities.

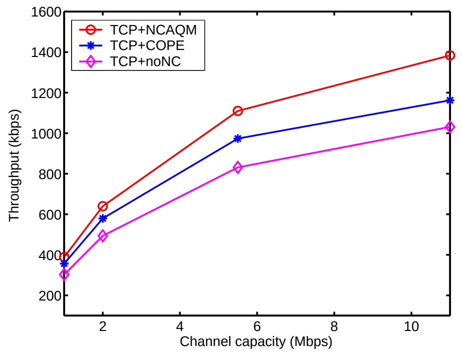
VII. Multi-Hop Network Coding

In this section, we extend our framework from one-hop to multi-hop network coding. We note that our framework can accommodate any given multi-hop network coding scheme, but we use BFLY [6] in our simulations, as an example.

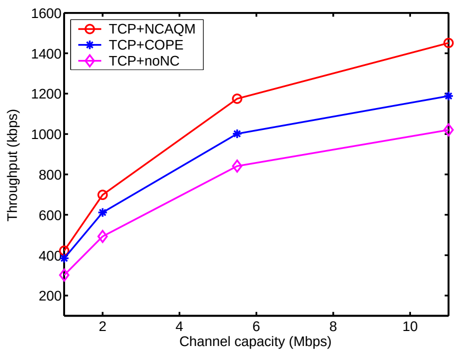
A. System Model

We consider the same system model as in Section III, with the difference of multi-hop, as opposed to one-hop, network coding. A flow s can be network coded and decoded several times over its path $\mathcal{P}_s$. The network coded flow may be transmitted over multiple (M) hops, which we call M-hop network coding. M-hop network coding is implemented by COPE [1] for $M = 1$, BFLY [6] for $M = 2$, or other network coding schemes for $M > 2$. We assume that a flow s cannot be network coded if it (or a part of it) is already coded. This assumption allows us to divide the path $\mathcal{P}_s$ to F_s intermediate paths which we call *network coding paths*. Over its f-th network coding path, where $f \in \{1, \ldots, F_s\}$, flow s can be network coded with $\Gamma_f^s \in \{0, 1, \ldots, |\mathcal{S} - \{s\}|\}$ other flows. Without loss of generality, we can assume that a flow may be transmitted over the f-th network coding path without network coding; *i.e.*, $\Gamma_f^s = 0$. A flow s can be divided into network coded and non-network coded parts over a network coding path f, where $\mathcal{Z}_s^f$ is the set of partitions of flow s over its f-th network coding path. Each partition $z \in \mathcal{Z}_s^f$ transmitted over hyperarc h has one-to-one mapping with the k-th network code over h such that $k \in \mathcal{K}_h$, *i.e.*, $z = \eta(k)$ over h where η is an injective function.

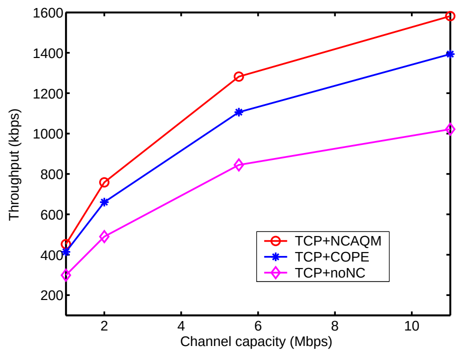
Example 3: The example shown in Fig. 8 illustrates the problem with 2-hop network coding. The flow from source S_1 is transmitted over the link $A_1 - I_1$ without network coding and it is network coded over the links $I_1 - I_2$ and $I_2 - A_2$. Over the network coding path, including the set of nodes I_1, I_2, A_2, the flow rate x_1 is partitioned into a network coded and a non-network coded part. The network coded part is combined with the corresponding part of the flow from source S_2, transmitted over $I_1 - I_2$, and broadcast over $(I_2, \{A_2, B_2\})$. The other part is transmitted over $I_1 - I_2$ and $I_2 - A_2$ without network coding. Similar to the one-hop network coding in Example 1,



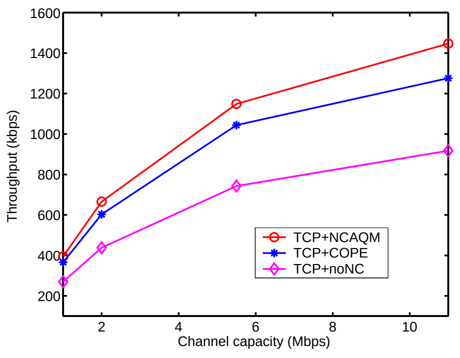
(a) Alice-and-Bob topology



(b) X topology



(c) Cross topology



(d) Grid topology

Fig. 7. Average throughput (averaged in $1min$ simulation first, then over 10 seeds) versus channel capacity for Alice-and-Bob (shown in Fig. 2(a)), X (shown in Fig. 1), cross (shown in Fig. 2(b)), and grid (shown in Fig. 3) topologies. Buffer size is 30 packets, and packet size is $500B$.

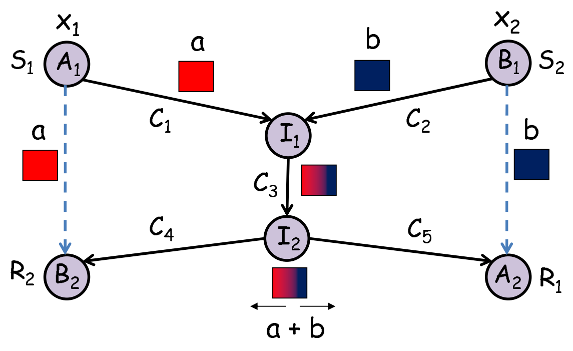


Fig. 8. "Butterfly topology". Source S_1 transmits a flow with rate x_1 to receiver R_1 and source S_2 transmits a flow with rate x_2 to receiver R_2, over the intermediate nodes I_1 and I_2. Nodes A_1 and B_1 transmit their packets a and b, in two time slots, and node I_1 receives them. Node B_2 overhears a and A_2 overhears b, because $A_1 - B_2$ and $B_1 - A_2$ are in the same transmission range and they can overhear each other. In the next time slot, I_1 transmits the network coded packet $a \oplus b$ to node I_2. Finally, I_2 broadcast $a \oplus b$ over hyperarc $(I_2, \{A_2, B_2\})$. Since A_2 and B_2 have overheard b and a, they can decode their packets a and b, respectively.

if there is a mismatch between the rates x_1, x_2 of the two flows, network coding benefit is not fully exploited. The goal is to solve this problem, assuming a given multi-hop network coding scheme. □

B. Problem Formulation

We consider the following NUM problem;

$$\begin{aligned}
&\max_{\boldsymbol{x},\boldsymbol{\alpha},\boldsymbol{\tau}} \sum_{s\in\mathcal{S}} U_s(x_s)\\
&\text{s.t.} \sum_{k\in\mathcal{K}_h} \max_{s\in\mathcal{S}_k}\{H_h^{s,k}\alpha_h^{s,k}x_s\} \le R_h\tau_h,\ \forall\ h\in\mathcal{A}\\
&\qquad \sum_{z\in\mathcal{Z}_s^f} \beta_f^{s,z} = 1,\ \forall\ s\in\mathcal{S}, f=1,...,F_s\\
&\qquad \alpha_h^{s,k} = \begin{cases}\beta_f^{s,z}, & \exists\ z=\eta(k), z\in\mathcal{Z}_s^f, f=1,...,F_s\\ 0, & \text{otherwise.}\end{cases}\\
&\qquad \sum_{h\in\mathcal{C}_q} \tau_h \le \tau,\ \forall\ \mathcal{C}_q\subseteq\mathcal{A}
\end{aligned} \tag{8}$$

The NUM problem in Eq. (8) is similar to the one in Eq. (1), in terms of the objective functions, capacity and interference constraints. We only need to update the flow conservation constraint (the second constraint) and add the third constraint, as explained below.

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

C. Solution

We use Lagrangian relaxation to solve the optimization problem in Eq. (8) by relaxing the capacity constraint with Lagrange multipliers q_h. We obtain the same Lagrange function in Eq. (2). The Lagrange function is decomposed into the same subproblems as in Eq. (3), Eq. (4), Eq. (6) and Eq. (7). The only different subproblem is the traffic splitting problem, which can be expressed as

$$\begin{aligned}
&\min_{\boldsymbol{\alpha}} \sum_{h\in\mathcal{A}} \sum_{k\in\mathcal{K}_h|s\in\mathcal{S}_k} q_h H_h^{s,k} \alpha_h^{s,k} (m_h^{s,k})^* \\
&\text{s.t.} \sum_{z\in\mathcal{Z}_s^f} \beta_f^{s,z} = 1,\ \forall\ s\in\mathcal{S}, f=1,...,F_s \\
&\quad \alpha_h^{s,k} = \begin{cases} \beta_f^{s,z}, & \exists\ z=\eta(k), z\in\mathcal{Z}_s^f, f=1,...,F_s \\ 0, & \text{otherwise.} \end{cases}
\end{aligned} \tag{9}$$

The objective function in Eq. (9) can be expanded to be $\sum_{f=1}^{F_s} \sum_{h\in\mathcal{A}^f} \sum_{k\in\mathcal{K}_h|s\in\mathcal{S}_k} q_h H_h^{s,k} \alpha_h^{s,k} (m_h^{s,k})^*$, where $\mathcal{A}^f$ is the set of hyperarcs that originate from the nodes in the f-th network coding path of flow s. The two objective functions are equivalent considering the fact that the objective function in Eq. (9) is equal to zero for hyperarcs which are not originated from the nodes over the flow's network coding paths, because the indicator functions ($H_h^{s,k}$) are zero for those hyperarcs.

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

$$\begin{aligned}
&\min_{\boldsymbol{\beta}} \sum_{f=1}^{F_s} \sum_{h\in\mathcal{A}^f} \sum_{z\in\mathcal{Z}_s^{f,h}} q_h H_h^{s,z} \beta_f^{s,z} (m_h^{s,z})^* \\
&\text{s.t.} \sum_{z\in\mathcal{Z}_s^f} \beta_f^{s,z} = 1,\ \forall\ s\in\mathcal{S}, f=1,...,F_s
\end{aligned} \tag{10}$$

The objective function in Eq. (10) can be expressed as $\sum_{f=1}^{F_s} \sum_{z\in\mathcal{Z}_s^f} \sum_{h\in\mathcal{A}^{f,z}} q_h H_h^{s,z} \beta_f^{s,z} (m_h^{s,z})^*$ where $\mathcal{A}^{f,z}$ which is the subset of $\mathcal{A}^f$ contains the hyperarcs over which the z-th partition of the f-th network coding path of flow s is transmitted. The two objective functions are equivalent, because the indicator functions ($H_h^{s,k}$) are zero for $h \notin \mathcal{A}^{f,z}$. Finally, the traffic splitting problem for $s\in\mathcal{S}, f=1,...,F_s$ is expressed as

$$\begin{aligned}
&\min_{\boldsymbol{\beta}} \sum_{z\in\mathcal{Z}_s^f} \beta_f^{s,z} \left(\sum_{h\in\mathcal{A}^{f,z}} q_h H_h^{s,z} (m_h^{s,z})^* \right) \\
&\text{s.t.} \sum_{z\in\mathcal{Z}_s^f} \beta_f^{s,z} = 1,\ \forall\ s\in\mathcal{S}, f=1,...,F_s
\end{aligned} \tag{11}$$

Similar to what we have done to solve Eq. (5), we use the proximal method [22] to solve this problem.

D. Simulation Results

In this section, we evaluate the throughput of TCP over NCAQM compared to TCP over the following baseline schemes: no network coding (noNC), which uses FIFO without network coding; BFLY [6], which utilizes knowledge of the local topologies by exchanging periodic messages that includes neighbors of nodes and source route information in the packet headers to exploit butterfly structures in wireless mesh networks. Similarly to COPE, BFLY stores native packets in a FIFO and decides which packets to code together at each transmission opportunity. We used the GloMoSim simulator [23] to implement the modules for two-hop network coding over wireless mesh networks (BFLY) as well as for our proposed scheme (NCAQM).

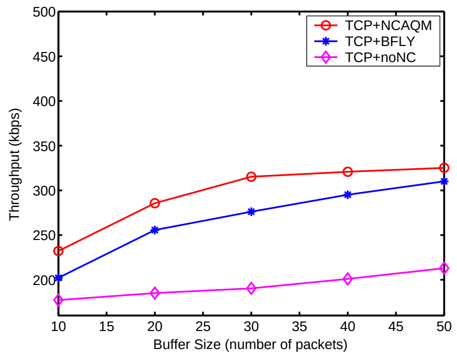
We simulate the butterfly topology shown in Fig. 8 in which two unicast flows S_1, R_1 and S_2, R_2 meet at intermediate node I_1. In this topology, nodes are placed over $300m \times 300m$ in butterfly like structure and a single channel is used for both uplink and downlink transmissions. We consider the same MAC update and wireless channel model as in Section VI. We consider FTP/TCP traffic over the wireless network. TCP flows, between the pairs of nodes described above, start at random times within the first $5sec$ and live until the end of the simulation.

Fig. 9(a) presents the average transport-level throughput vs. buffer size. Similarly to the simulation results in Section VI, TCP+NCAQM improves throughput much more than TCP+BFLY. Specifically, when buffer size is 10 packets, the improvement of TCP+BFLY over TCP+noNC is 13%, the improvement of TCP+NCAQM over TCP+noNC is 30%, while the optimum improvement is 50%. When buffer size increases, we see that TCP+NCAQM approaches and exceeds the optimum; *e.g.*, the improvement of TCP+NCAQM is 65% when buffer size is 30 packets, while it is 45% for TCP+BFLY. This shows that the advantages of TCP+NCAQM also apply to two-hop network coded wireless mesh networks.

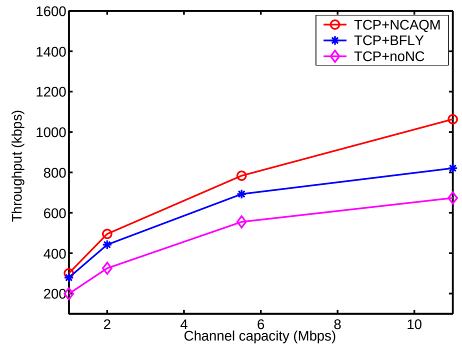
Fig. 9(a) presents the average transport-level throughput vs. channel capacity. We can see that the improvement of TCP+NCAQM is larger than TCP+BFLY for all channel capacities and it is especially significant for large channel capacities, since TCP+NCAQM uses buffer more effectively and drops packets so that network coding opportunities increase.

VIII. Conclusion

In this paper, we showed how to improve the performance of TCP over wireless networks with inter-session network coding. The key intuition was to eliminate the rate mismatch between flows that are coded together through a synergy of rate control and queue management. First, we formulated congestion control as a NUM problem and derived a distributed solution. Motivated by the structure of the solution, we proposed minimal modifications to queue management to make it network coding-aware, while TCP and MAC protocols remained intact. Simulation results show that the proposed NCAQM scheme doubles TCP performance compared to baseline schemes and achieves near-optimal performance. We plan to make the simulator modules publicly available to



(a) Buffer Size



(b) Channel capacity

Fig. 9. Average throughput (averaged in $1min$ simulation first, then over 10 seeds) in butterfly topology shown in Fig. 8. (a) Buffer size: packet size is $500B$, and the channel capacity is $1Mbps$. (b) Channel capacity: buffer size is 30 packets, and packet size $500B$.

the research community. We have also extended the NUM formulation and solution to multi-hop network coding and we have confirmed convergence through numerical calculations. The main ideas of this paper can potentially be extended from wireless mesh networks to wired networks with constructive inter-session network coding.

References

[1] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the Air: Practical Wireless Network Coding," *IEEE/ACM Trans. on Networking*, Vol. 16, No. 3, pp. 497-510, June 2008.

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

[3] Y. Huang, M. Ghaderi, D. Towsley, and W. Gong, "TCP performance in coded wireless mesh networks," *in Proc. of IEEE SECON*, San Francisco, CA, June 2008.

[4] R. Srikant, "The Mathematics of Internet Congestion Control", Birkhauser, 2003.

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

[6] S. Omiwade, R. Zheng, and C. Hua. "Butterflies in the mesh: lightweight localized wireless network coding," *in Proc. of NetCod*, Hong Kong, Jan. 2008.

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

[8] D. S. Lun, N. Ratnakar, M. Medard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao, "Minimum-cost multicast over coded packet networks," *in IEEE Trans. on Internet Technology*, vol. 52(6), June 2006.

[9] L. Chen, T. Ho, S. Low, M. Chiang, and J. C. Doyle, "Optimization based rate control for multicast with network coding," *in Proc. of Infocom*, Anchorage, AK, 2007.

[10] J. Yuan, Z. Li, W. Yu, and B. Li, "A cross-layer optimization framework for multi-hop multicast in wireless mesh networks," *in IEEE JSAC*, vol. 24(11), Nov. 2006.

[11] Z. Li, B. Li, and M. Wang, "Optimization models for streaming in multihop wireless networks," *in Proc. of ICCCN*, Honolulu, HI, Aug. 2007.

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

[13] P. Chaporkar and A. Proutiere, "Adaptive network coding and scheduling for maximizing througput in wireless networks," *in Proc. of ACM Mobicom*, Montreal, Canada, Sep. 2007.

[14] F. Zhao, M. Medard, "On analyzing and improving COPE performance," *in Proc. of ITA*, San Diego, CA, Feb. 2010.

[15] J. Le, J. Lui, and D. M. Chiu, "How many packets can we encode? - an analysis of practical wireless network coding," *in IEEE INFOCOM*, Phoenix, AZ, April 2008.

[16] Q. Dong, J. Wu, W. Hu, and J. Crowcroft, "Practical network coding in wireless networks," *in Proc. of MobiCom*, Montreal, Canada, Sept. 2007.

[17] S. Sengupta, S. Rayanchu, and S. Banarjee, "An Analysis of Wireless Network Coding for Unicast Sessions: The Case for Coding-Aware Routing," *in Proc. of Infocom*, Anchorage, AK, 2007.

[18] A. Khreishah, C. C. Wang, and N. B. Shroff, "Cross-layer optimization for wireless multihop networks with pairwise inter-session network coding," *in IEEE JSAC*, vol. 27(5), June 2009.

[19] T. Cui, L. Chen, and T. Ho, "Energy Efficient Opportunistic Network Coding for Wireless Networks," *in Proc. of Infocom*, Phoenix, AZ, April 2008.

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

[21] P. Gupta and P. R. Kumar, "The Capacity of Wireless Network," *in IEEE Trans. on Information Theory*, vol. 34(5), pp. 910-917, 2000.

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

[23] GloMoSim Version 2.0 , "Global Mobile Information Systems Simulation Library," *available at http://pcl.cs.ucla.edu/projects/glomosim/*.

[24] H. Seferoglu, A. Markopoulou, and K. K. Ramakrishnan, "I^{2}NC: Intra- and inter-session network coding for unicast flows in wireless networks," *in arXiv:cs.NI:1008.5217*, Aug. 2010.

[25] H. Seferoglu and A. Markopoulou, "Network Coding-Aware Queue Management for Unicast Flows over Coded Wireless Networks," *in Proc. of NetCod*, Toronto, Canada, June 2010.

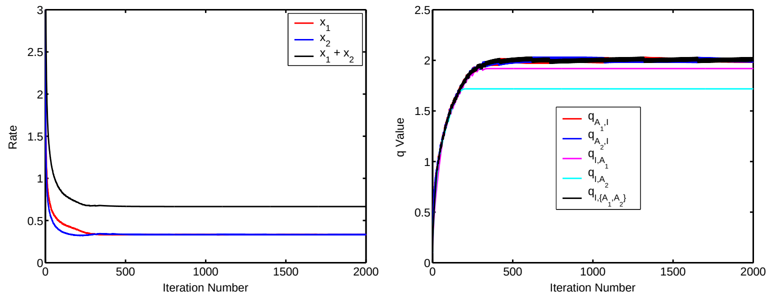
Appendix A: Numerical Results

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

A. *One-hop Network Coding*

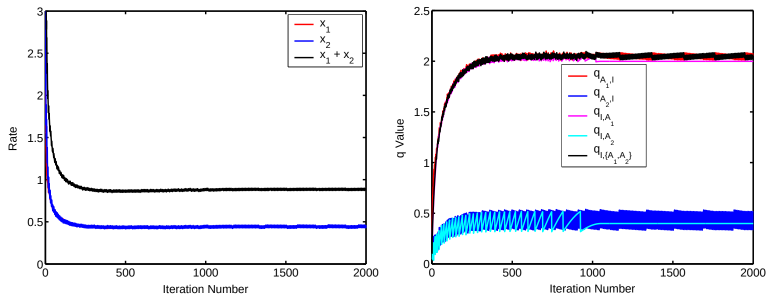
First, we consider the Alice-and-Bob topology presented in Fig. 2(a). We consider two cases for wireless channel capacities: (i) $C_1 = C_2 = 1$ units/transmission[5], and (ii) $C_1 = 1$, $C_2 = 4$. For the first case, the convergence of rates x_1, x_2, and x_1+x_2 is presented in Fig. 10(a). One can see that the total rate $x_1 + x_2$ converges to 0.66 which is the optimal achievable rate when network coding is used for this scenario. Note that total achieved throughput is 0.50 for this scenario when network coding is not used. For the second case, it is seen in seen in Fig. 11(a) that the total throughput approaches to the optimal achievable rate of 0.88 when network coding is used. Note that the total achievable throughput is 0.80 when network coding is not used. We also present the convergence of

[5]We omit the units in the rest of the paper for brevity.



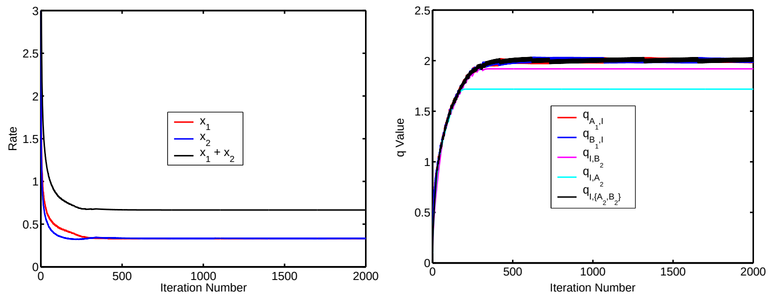
(a) Rate (b) Lagrange multipliers

Fig. 10. Convergence results for the Alice-and-Bob topology presented in Fig. 2(a). The total achieved rate approaches the optimum throughput 0.66. The optimum throughput is 0.50 when there is no network coding. $C_1 = C_2 = 1$.



(a) Rate (b) Lagrange multipliers

Fig. 11. Convergence results for the Alice-and-Bob topology presented in Fig. 2(a). The total achieved rate approaches the optimum throughput 0.88. The optimum throughput is 0.80 when there is no network coding. $C_1 = 1$, $C_2 = 4$.



(a) Rate (b) Lagrange multipliers

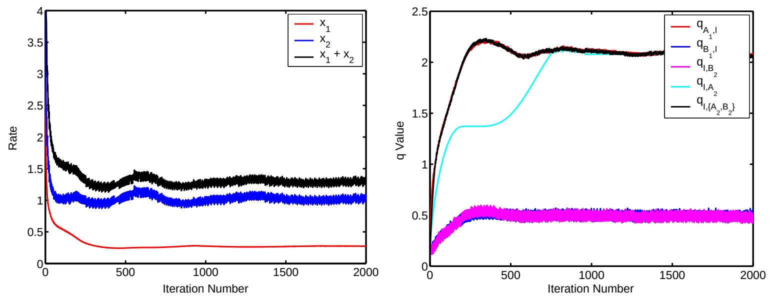
Fig. 12. Convergence results for the X topology presented in Fig. 1. The total achieved rate approaches to the optimum throughput 0.66. The optimum throughput is 0.50 when there is no network coding. $C_1 = C_2 = C_3 = C_4 = 1$.

Lagrange multipliers; $q_{A_1,I}$, $q_{A_2,I}$, q_{I,A_2}, q_{I,A_1}, and $q_{I,\{A_1,A_2\}}$ for both cases in Fig. 10(b) and Fig. 11(b), respectively.

Second, we consider the X topology presented in Fig. 1. We consider two cases for wireless channel capacities: (i) $C_1 = C_2 = C_3 = C_4 = 1$, and (ii) $C_1 = C_4 = 1$, $C_2 = C_3 = 4$. In both cases the total rate $x_1 + x_2$ approaches to the optimum achievable rates; 0.66 and 1.3 as seen in Fig. 12(a) and Fig. 13(a). We also show results for the convergence of the Lagrange multipliers for both cases in Fig. 12(b) and Fig. 13(b).

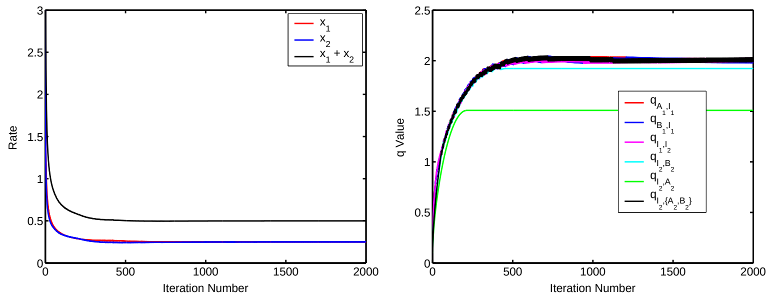
B. Multi-Hop Network Coding

We consider the butterfly topology presented in Fig. 8. We consider two scenarios for the wireless channel capacities; (i) $C_1 = C_2 = C_3 = C_4 = C_5 = 1$ and (ii) $C_1 = C_2 = C_4 = C_5 = 4$, $C_3 = 1$. The total rate approaches the optimal achievable rate in both scenarios: 0.5 for the first case as shown in Fig. 14(a), and 1.14 for the second case as shown in Fig. 15(a). In both scenarios, we show the convergence of the Lagrange multipliers, Fig. 14(b) and Fig. 15(b).



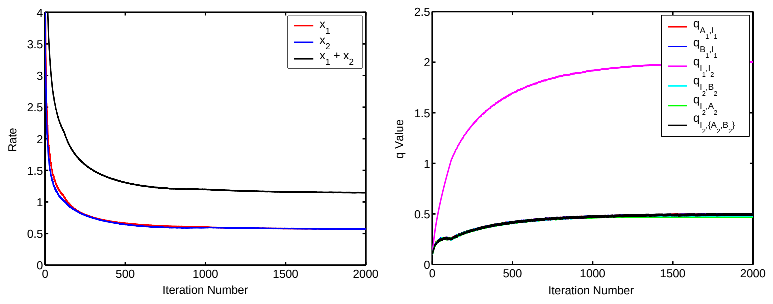
(a) Rate (b) Lagrange multipliers

Fig. 13. Convergence results for the X topology presented in Fig. 1. The total achieved rate approaches the optimum throughput 1.3. The optimum throughput is 0.80 when there is no network coding. $C_1 = C_4 = 1$, $C_2 = C_3 = 4$.



(a) Rate (b) Lagrange multipliers

Fig. 14. Convergence results for the butterfly topology presented in Fig. 8. The total achieved rate approaches the optimum throughput 0.50. The optimum throughput is 0.33 when there is no network coding. $C_1 = C_2 = C_3 = C_4 = C_5 = 1$.



(a) Rate (b) Lagrange multipliers

Fig. 15. Convergence results for the butterfly topology presented in Fig. 8. The total achieved rate approaches the optimum throughput 1.14. The optimum throughput is 0.66 when there is no network coding. $C_1 = C_2 = C_4 = C_5 = 4$, $C_3 = 1$.