Fast Resource Allocation for Network-Coded Traffic — A Coded-Feedback Approach

Chih-Chun Wang and Xiaojun Lin
School of Electrical and Computer Engineering, Purdue University
Email: {chihw,linx}@purdue.edu.

***Abstract*—In this paper, we develop a fast resource allocation algorithm that takes advantage of intra-session network coding. The algorithm maximizes the total utility of multiple unicast (or multicast) sessions subject to capacity constraints, where packets are coded within each session. Our solution is a primal solution that does not use duality or congestion prices. Thus, it does not require building up queues to achieve the optimal resource allocation. Hence, the queueing delay of the packets can be tightly controlled. The existing primal solution in the literature requires a separate graph-theoretic algorithm to find the min-cut of each session, whose complexity grows quadratically with the total number of nodes. In contrast, we provide a new *coded-feedback* approach whose complexity grows only linearly with the total number of nodes. More explicitly, by *letting the ACK/feedback packets on the return paths also carry coding coefficients* as does the forward coded traffic, key network information can be obtained more efficiently, which leads to a fast resource allocation scheme fully integrated with the network coding operation.**

I. Introduction

In this paper, we are interested in developing fast resource allocation algorithms that take advantage of intra-session network coding. For multicast sessions, intra-session network coding has been found to offer two benefits over routing. First, the achievable throughput with coding is often larger than that with routing [1]. Further, the maximum throughput can be achieved with high probability by using simple random linear network codes [2]. Second, the resource allocation problem with coding can be modeled as a convex optimization problem that is easy to solve. In contrast, finding the optimal multicast tree with routing is often NP-hard [3].

Although network coding leads to considerable performance improvement for multicast sessions, the majority of the traffic in today's Internet is by unicast. Intra-session network coding does not offer throughput advantage over routing for unicast. Nonetheless, we argue in this paper that, even for unicast traffic, intra-session network coding offers an attractive alternative to routing. We use the following simple and motivational example. Consider the case when there is only a single unicast session on a network (e.g., on a community LAN or a Virtual Private Network). Simple random linear network coding can be used to easily approach the maximum rate (often referred to as the *min-cut max-flow* rate) between its source and destination. There is no need to run a separate max-flow algorithm (such as push-&-relabel [4]), which incurs additional complexity. Further, unlike the back-pressure algorithm [5], *there is no queue build-up and hence the packet delay can be tightly controlled.*

In real networks, multiple unicast (or multicast) sessions must share the network resources. Existing resource allocation algorithms are devised separately from the network coding operations and can be classified into two categories: primal solutions and dual solutions. In a typical dual solution, the queue at each link is interpreted as a price signal, which indicates the level of imbalance between the offered-load and the capacity. The dual solution then updates the resource allocation based on this price signal [3], [6], [7]. One of the main disadvantage of dual solutions is that there can be a disconnection between the optimality of the dual variables (i.e., the price) and the optimality of the primal variables (i.e., the resource allocation). First, even if the price converges, the rate allocation may not converge. An additional step (e.g., using proximal algorithms [7]) is often required to enforce the convergence of the primal variables, which increases the complexity of the solution. Second, before the price converges, the system utility does not necessarily improve in each iteration. In fact, before the algorithm converges, the rate allocation often violates the capacity constraints, which leads to queue-length dynamics that are difficult to predict and control.

In contrast, primal solutions adjust the primal control variables (i.e., the rate allocation) directly within the capacity constraints [8], [9]. Hence, the resulted rate allocation typically improves the system utility (towards the optimal) over each iteration. Further, even before the algorithm converges, the rate-allocation always satisfies the capacity constraints, and hence there is no queue build-up.

However, a key step in the existing primal solution with network coding [8] is to use a separate graph-theoretic algorithm to find the min-cut of the network. To the best of our knowledge, the time-complexity of existing min-cut algorithms in the literature is at least $\mathcal{O}(|V|^2)$, which significantly increases the overall complexity of the solution. In this paper, we develop a much faster resource allocation algorithm. Specifically, by exploiting and refining a novel concept of *coded feedback* (first introduced in [10]), we develop a new and fully distributed algorithm that only takes $\mathcal{O}(|V|)$ time to compute the min-cut. This is an order of degree faster than existing min-cut algorithms[1] (including the $\mathcal{O}(|V|^2)$ coded-feedback min-cut/max-flow algorithms devised in [10]). Our coded-feedback

[1]Throughout this paper, we ignore the computation time within a network node because we are mostly interested in the speed of distributed algorithms. Thus, the time-complexity discussion is based on the delay caused by exchanging control messages rather than caused by computation within a node.

approach computes the min-cut as an integral part of network-coding operation, by coding the feedback packets on the return path. Hence, it incurs minimum overhead and is easy to implement. We then develop a low-complexity and distributed resource allocation scheme working in conjunction with the coded-feedback algorithm. Our solution enjoys the benefits of both primal solutions and the simplicity of network coding with minimal control and communication overhead. Even for unicast traffic, our algorithm can serve as an attractive alternative to traditional non-coded solutions, such as the back-pressure algorithm [5]. Our algorithm has comparable convergence speed and overhead as the back-pressure algorithm. On the other hand, since our solution does not require the build-up of queues as price signals, the queue-length and the packet delay can be tightly controlled.

II. System Model & The Overall Methodology

A. System Model

We represent a wireline network by a graph $G = (V, L)$, where V is the set of nodes and L is the set of directed links. Each link $l \in L$ has a fixed capacity R^l. For the sake of simplicity, we assume that there is a set I of *unicast* sessions that share the resources in such a network. For each session $i \in I$, let s_i and d_i denote its source node and destination node, respectively. Let r_i^l denote the capacity of link l that is allocated to session i. Let $\vec{r}_i = [r_i^l, l \in L]$. From session i's point of view, its packets are transmitted on a sub-network with the same topology $G = (V, L)$ except that the capacity of each link is r_i^l. In addition, to simplify the description of the network coding operation, we assume that each session i only uses an acyclic subgraph $G_i = (V, L_i)$ of G, where the link subset L_i forms an acyclic graph. (Note that the problem formulation and the solution can be readily extended to the case with multicast sessions and with cyclic subgraphs. Readers can refer to [11] for details.)

Define a cut C_i for session i to be a subset of links from L_i such that when these links are removed from the subgraph G_i, the source s_i and the destination d_i are disconnected. Given $\vec{r}_i$, define the value $\xi_i(C_i|\vec{r}_i)$ of the cut C_i as the total rate (allocated to session i) of the links that belong to C_i. Let $\hat{\mathcal{C}}_i$ denote the collection of all cuts for session i. Define a minimum-cut (or min-cut) $C_i^{\min}$ for session i as the cut with the smallest value, i.e.,

$$\xi_i(C_i^{\min}|\vec{r}_i) = \min_{C_i \in \hat{\mathcal{C}}_i} \xi_i(C_i|\vec{r}_i). \tag{1}$$

This value $\xi_i(C_i^{\min}|\vec{r}_i)$, known as the min-cut max-flow value of session i, is the maximum rate that session i can transfer packets from source s_i to destination d_i. We will denote it as $\mathbf{MCMF}_i(\vec{r}_i)$.

Assume that each session has a utility function $U_i(x_i)$ that is strictly concave and non-decreasing. The utility function $U_i(x_i)$ characterizes the satisfactory level of the user of session i when the service rate that it receives is x_i. We assume that the derivative of $U_i(\cdot)$ is bounded by M_0 for all i. Let $\vec{r} = [\vec{r}_i, i = 1, ..., |I|]$. We are interested in the following utility-maximization problem subject to capacity constraints:

$$\max_{\vec{r}=[\vec{r}_i] \geq 0} \quad F(\vec{r}) \triangleq \sum_{i=1}^{|I|} U_i(\mathbf{MCMF}_i(\vec{r}_i)) \tag{2}$$

$$\text{subject to} \quad \sum_{i=1}^{|I|} r_i^l \leq R^l \text{ for all links } l. \tag{3}$$

This problem formulation is similar to the one proposed in [8]. Next, we first present a high-level overview of our solution methodology, which also has some similarity to the solution of [8]. We then highlight the main step where our solution differs from [8].

B. The Overall Solution Methodology

One can show that Problem (2) is a convex optimization problem [8], and thus can be solved by a subgradient-ascent algorithm. With a given $\vec{r}_i$, consider any min-cut $C_i^{\min}$ for session i. Define a vector $\vec{\mu}_i = [\mu_i^l, l \in L]$ such that

$$\mu_i^l = \begin{cases} 1 & \text{if } l \in C_i^{\min} \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

It has been shown in [8] that the vector $\vec{\mu}_i$ is a subgradient of $\mathbf{MCMF}_i(\cdot)$ at $\vec{r}_i$. We can then construct a subgradient $\partial F(\vec{r})$ of the objective function $F(\vec{r})$ of Problem (2) such that its (i, l)-th component is given by [8]:

$$[\partial F(\vec{r})]_{il} = U_i'(\mathbf{MCMF}_i(\vec{r}_i))\mu_i^l, \tag{5}$$

where $U_i'(\cdot)$ is the derivative of the utility function $U_i(\cdot)$. Let Λ denote the set of $\vec{r}$ that satisfies the capacity constraints in (3). We can then use the following constrained subgradient-ascent algorithm to solve (2). Let $\vec{r}(t)$ denote the resource allocation at iteration t, and let $\partial F(\vec{r}(t))$ denote the corresponding subgradient (5) of the objective function at $\vec{r}(t)$. The subgradient-ascent iteration is then given by:

$$\vec{r}(t+1) = [\vec{r}(t) + \alpha \partial F(\vec{r}(t))]_\Lambda, \tag{6}$$

where α is a positive step-size and $[\cdot]_\Lambda$ denotes the projection to Λ. Like [8], the above solution is a primal solution, and hence it has the desirable features of primal solutions as discussed in Section I.

C. Mapping to the Unit-Capacity Network Model

The basic algorithm in Section II-B is generic and applies to links of arbitrary (fractional) capacity. With network coding, the coding operation is carried out on a packet-by-packet (integer) basis. We next present an integer-capacity network model for network coding, and discuss its connection to the network model in Section II-A.

Practical network coding divides the source packets into *generations*, and packets of the same generation are mixed/coded together with the corresponding coefficients recorded in the header [12]. Fix a session i. To reduce the overhead (the size of the header), the generation size n is generally fixed to some number ≤ 100. Our work assumes

that the source node s_i switches to a new generation every D_i time. For each intermediate node, after the last packet of the new generation arrives, the node will switch to the new generation and start transmitting mixed packets for the new generation on all out-going links. Thus, each node also switches to a new generation every D_i time. Therefore, a link with capacity r_i^l packets-per-second can carry $\lfloor r_i^l D_i \rfloor$ packets for one generation. We can then construct a unit-capacity network by mapping each link of capacity r_i^l to $\lfloor r_i^l D_i \rfloor$ parallel edges, each of which carries one packet for one generation. Let $G_i(V, E_i)$ denote this unit-capacity network for session i. Let $\mathbf{MCMF}(\{\lfloor r_i^l D_i \rfloor\}_l)$ denote the min-cut max-flow value of the new unit-capacity network. Due to this practical implementation, the achievable throughput is $\frac{1}{D_i}\min(n, \mathbf{MCMF}(\{\lfloor r_i^l D_i \rfloor\}_l))$ packets-per-second instead of $\mathbf{MCMF}(\{r_i^l\})$. Note that if D_i is too large, then the achievable throughput is dominated by $\frac{n}{D_i}$. If D_i is too small, the non-linearity of the floor function $\lfloor \cdot \rfloor$ also reduces the rate. With a fixed $n \leq 100$, D_i is best to be chosen such that

$$\frac{1}{D_i}\min(n, \mathbf{MCMF}(\{\lfloor r_i^l D_i \rfloor\}_l)) \approx \mathbf{MCMF}(\{r_i^l\}). \quad (7)$$

Note that the propagation delay of the original network can also be captured by this unit-edge-capacity model in a way that long-delay links are modelled by multi-hop paths with added auxiliary nodes.

The above unit-capacity model will be used for describing the new min-cut algorithm in the next section, which can compute a min-cut in $\mathcal{O}(|V|)$ steps. Note that finding a min-cut $C_i^{\min}$ for each session i is a critical step in our overall solution methodology. The algorithm in the next section replaced the push-&-relabel (PNR) algorithm (used in [8]) that requires $\mathcal{O}(|V|^2)$ communication time [4]. In addition, a highly-desirable feature of our new approach is that the computation is tightly integrated with the network coding operations. Hence, it incurs minimum overhead and is easy to implement.

III. Finding Min-Cut Using Coded Feedback

For the following, we focus on the unit-capacity directed acyclic subnetwork $G_i(V, E_i)$ for a given session i, as first discussed in Section II-C. For simplicity, we drop the subscript i whenever there is no confusion. Let s and d denote the source and destination, respectively. Without loss of generality, we assume s being the most upstream node and d being the most downstream node. Nodes are labeled according to their topological order (i.e. $s = 1$ and $d = |V|$). We use $\mathrm{In}(v) \subseteq E$ and $\mathrm{Out}(v) \subseteq E$ to denote all edges entering/leaving node v. For simplicity, we use finite Galois field $\mathrm{GF}(2^b)$, although our algorithm works for finite fields of any size.

Consider network coding of generation size n, for which each packet along a unit-capacity edge e carries an n-dimensional *global coding vector* $m_e = (m_1, \cdots, m_n)$ in its header, such that the coded symbol $X_e \in \mathrm{GF}(2^b)$ in the payload is $X_e = m_1 X_1 + \cdots + m_n X_n$ where X_1 to X_n are the information symbols. The description of the following algorithm will focus mainly on m_e while the associated payload computation X_e is carried out correspondingly. We assume each m_e being a row vector. For any subset $E' \subseteq E$ of edges, $[m_e : e \in E']$ is an $|E'| \times n$ matrix constructed by concatenating $|E'|$ row vectors m_e vertically.

A. The Algorithm

Using this unit-edge-capacity model, *random linear network coding* can be expressed by the following two sub-routines:

§ Choose Random Mixing Coefficients

1: Each node $v \in V \backslash \{s, d\}$ chooses randomly an $|\mathrm{Out}(v)| \times |\mathrm{In}(v)|$ matrix $\Gamma(v) = [\gamma_{w,u}(v)]$ where each entry $\gamma_{w,u}(v)$ is chosen *independently* and *uniformly randomly* from $\mathrm{GF}(2^b)$ for all $(v, w) \in \mathrm{Out}(v)$ and $(u, v) \in \mathrm{In}(v)$.

§ Compute Coded Traffic

1: Source s sends out randomly coded symbols along $\mathrm{Out}(s)$. As a result, the corresponding m_e is chosen uniformly randomly.
2: **for** $v = s + 1, \cdots, d - 1$ (i.e. from the most upstream node to the most downstream node) **do**
3: $\quad [m_e : e \in \mathrm{Out}(v)] \leftarrow \Gamma(v)[m_{e'} : e' \in \mathrm{In}(v)]$.
4: $\quad$ Node v sends out m_e along all edges in $\mathrm{Out}(v)$.
5: **end for**

The novel component of our new min-cut algorithm is *the coded feedback message* q_e, which is an n-dimensional row vector sent in the opposite direction of m_e. Namely, for any $e = (u, v) \in \mathrm{In}(v)$, its coded feedback vector q_e is a linear combination of $\{q_{e'} : e' \in \mathrm{Out}(v)\}$. In practice, the coded feedback vector q_e can also be embedded in the header of the acknowledgement packet in a similar way as the coding coefficients m_e for the forward coded traffic. A detailed description of the min-cut algorithm is as follows.

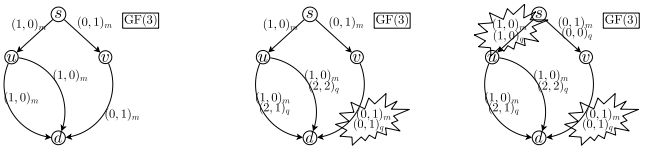
§ **A New Min-Cut Algorithm — The Main Algorithm**

1: Run Choose Random Mixing Coefficients
2: Run Compute Coded Traffic
3: Run Compute Coded Feedback
4: **return** $\{e \in E : m_e q_e^{\mathrm{T}} = 1\}$

The first two sub-routines are exactly the same as those in random linear network coding. The only new sub-routine Compute Coded Feedback is described as follows. Let $\mathrm{Rank}(d)$ denote the rank of the matrix $[m_e : e \in \mathrm{In}(d)]$.

§ Compute Coded Feedback

1: Destination d *randomly* constructs $(n - \mathrm{Rank}(d))$ vectors $\{m_a^* : a \in \{1, \cdots, n - \mathrm{Rank}(d)\}\}$ such that jointly $\{m_e : e \in \mathrm{In}(d)\}$ and $\{m_a^*\}$ span the entire n-dimensional vector space.
2: Destination d *randomly* constructs two sets of vectors $\{q_e : e \in \mathrm{In}(d)\}$ and $\{q_a^* : a \in \{1, \cdots, n - \mathrm{Rank}(d)\}\}$ such



(a) Normal Network Coding.

(b) Construct the initial coded feedback $[q_e : e \in \mathrm{In}(d)]$ and identify the first edge with $m_e q_e^{\mathrm{T}} = 1$.

(c) Continue the propagation of q_e and identify the second edge with $m_e q_e^{\mathrm{T}} = 1$.

Fig. 1. An illustration of the new min-cut algorithm.

that the following matrices

$$\mathbf{q} \triangleq \begin{pmatrix} [q_e : e \in \mathrm{In}(d)] \\ [q_a^* : a \in \{1, \cdots, n - \mathrm{Rank}(d)\}] \end{pmatrix}$$
$$\mathbf{m} \triangleq \begin{pmatrix} [m_e : e \in \mathrm{In}(d)] \\ [m_a^* : a \in \{1, \cdots, n - \mathrm{Rank}(d)\}] \end{pmatrix}$$

satisfy $\mathbf{q}^{\mathrm{T}}\mathbf{m} = \mathbf{I}_n$ where $\mathbf{I}_n$ is an n by n identity matrix and $\mathbf{q}^{\mathrm{T}}$ is the transpose of $\mathbf{q}$.

3: Destination d sends out the newly constructed $\{q_e : e \in \mathrm{In}(d)\}$ along each edge in $\mathrm{In}(d)$.
4: **for** $v = d-1, \cdots, s+1$ (i.e. from the most downstream node to the most upstream node) **do**
5: Construct $\mathbf{q}_v \leftarrow [q_e : e \in \mathrm{Out}(v)]$.
6: For those $e \in \mathrm{Out}(v)$ satisfying $m_e q_e^{\mathrm{T}} = 1$, replace the corresponding rows in $\mathbf{q}_v$ by all-zero vectors.
7: $[q_e : e \in \mathrm{In}(v)] \leftarrow \Gamma(v)^{\mathrm{T}}\mathbf{q}_v$.
8: Node v sends q_e along edges in $\mathrm{In}(v)$.
9: **end for**

We provide an example of this min-cut algorithm. Consider a simple network as in Fig. 1(a). Suppose there are $n = 2$ uncoded symbols and GF(3) is used. Fig. 1(a) describes normal network coding, in which nodes u and v simply forward the coding vectors $(1, 0)$ and $(0, 1)$, or equivalently the corresponding mixing matrices are $\Gamma(u) = (1, 1)^{\mathrm{T}}$ and $\Gamma(v) = 1$. (The forward coding vectors are illustrated with subscript m.) Line 1 of COMPUTE CODED FEEDBACK is redundant in this particular example because $\mathrm{Rank}(d) = 2 = n$. Fig. 1(b) illustrates one choice of $[q_e : e \in \mathrm{In}(d)]$ such that

$$[q_e : e \in \mathrm{In}(d)]^{\mathrm{T}}[m_e : e \in \mathrm{In}(d)]$$
$$= \begin{pmatrix} 2 & 2 & 0 \\ 1 & 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = \mathbf{I}_2 \text{ in GF(3)}$$

satisfying Line 2 of COMPUTE CODED FEEDBACK. (The coded feedback vectors q are labeled with subscript q.) Fig. 1(b) shows that edge (v, d) has $m_{(v,d)} q_{(v,d)}^{\mathrm{T}} = 1$. Fig. 1(c) continues the propagation of q_e back to the source s. Since $\Gamma(u) = (1, 1)^{\mathrm{T}}$, $q_{(s,u)} = \Gamma(u)^{\mathrm{T}}[q_e : e \in \mathrm{Out}(u)] = (2, 1) + (2, 2) = (1, 0)$. Since $\Gamma(v) = 1$ and $m_{(v,d)} q_{(v,d)}^{\mathrm{T}} = 1$, we have $q_{(s,v)} = \Gamma(v)^{\mathrm{T}}\mathbf{q}_v = 1 \cdot (0, 0) = (0, 0)$ by Lines 6 and 7. The final m_e and q_e are illustrated in Fig. 1(c) and there are exactly two edges $\{(s, u), (v, d)\}$ satisfying $m_e q_e^{\mathrm{T}} = 1$. Our new coded-feedback algorithm outputs $\{(s, u), (v, d)\}$, which is indeed a minimum cut separating s and d.

B. Correctness & Complexity

Proposition 1: Let $P(\text{success})$ denote the probability that the output of our new min-cut algorithm is a minimum cut separating s and d. We then have

$$P(\text{success}) \geq (1 + |E|)\left(1 - 2^{-b}\right)^{|E|} - |E|. \tag{8}$$

When the field size $2^b \to \infty$, $P(\text{success})$ tends to one.

Note that (8) converges exponentially fast to one with respect to b, the number of bits assigned for each symbol. A detailed proof of Proposition 1 is provided in [11].

Proposition 2: The **Main Algorithm** stops in $2|V|$ steps.

Proof: Since both COMPUTE CODED TRAFFIC and COMPUTE CODED FEEDBACK process the nodes in their topological order and finish in $|V|$ steps, the proof is straightforward. Note that once the feedback vector q_e is computed, each edge e immediately knows whether it belongs to the output edge set or not by checking the inner product of m_e and q_e. ■

We compare the time-complexity of the above algorithm with that of push-&-relabel [4]. Assuming that each node has very high processing power, then the processing time is negligible and the delay is mainly caused by exchanging control messages. For any fixed generation size n, the time-complexity of our coded-feedback algorithm is $\mathcal{O}(|V|)$, which is lower than the $\mathcal{O}(|V|^2)$ time-complexity of push-&-relabel, especially when $|V|$ is large. When $|V|$ is not very large, the comparison will need to take into account the parameter n. For typical implementations, the generation size n is taken to be no greater than 100. When the generation interval D_i is chosen according to (7), each link will be mapped to at most $\mathcal{O}(n)$ parallel edges. Since each edge carries an n-dimensional vector, the total length of the control messages along each link is $\mathcal{O}(n^2)$. Let t_{tran} denote the transmission delay *per byte* and let t_{prop} denote the propagation delay. The overall delay of our algorithm becomes $\mathcal{O}((n^2 t_{\text{tran}} + t_{\text{prop}})|V|)$. For comparison, for push-&-relabel the control message on each link is of fixed length and the associated delay is $\mathcal{O}((t_{\text{tran}} + t_{\text{prop}})|V|^2)$. The exact impact of the additional $\mathcal{O}(n^2)$ factor depends on the relative order between t_{tran} and t_{prop}. For example, assume that the link capacity is 10M bps. Take $n = 100$ and $GF(2^8)$ as used in [12]. Then the transmission time for $\mathcal{O}(n^2)$ coding co-efficients is of the order of 8 ms. If t_{prop} on each link is of the same order, then the coded feedback algorithm will converge faster than push-&-relabel in spite of the additional $\mathcal{O}(n^2)$ factor. On the other hand, if the propagation delay is much less than 8 ms, then the push-&-relabel algorithm may converge more quickly when $|V|$ is not very large.

IV. COMBINING CODED FEEDBACK WITH SUBGRADIENT-ASCENT

Recall the practical coding implementation discussed in Section II-C. Each edge carries $\lfloor r_i^l(t) D_i \rfloor$ packets for one generation, and the effective rate is $\bar{r}_i^l(t) = \frac{1}{D_i} \lfloor r_i^l(t) D_i \rfloor$. We next describe how the updates in (4)-(6) should be modified according to the result computed by the coded-feedback algorithm. Once the coded-feedback algorithm returns a subset $\tilde{C}_i$

of *edges* (which with high probability is the min-cut for the unit-capacity sub-network for session i), we then construct a subset of *links* $C_i \subset L_i$ from the *edge* subset $\tilde{C}_i$ as follows. For any link $l \in L_i$, if $\tilde{r}_i^l(t) = 0$, then include l in C_i. If $\tilde{r}_i^l(t) > 0$ and if all $\lfloor \tilde{r}_i^l(t) D_i \rfloor$ unit-capacity edges that link l maps to belong to $\tilde{C}_i$, then include l in C_i. For other cases, exclude l from C_i. The reason behind the above construction is as follows. If $\tilde{C}_i$ is indeed a min-cut for the unit-capacity sub-network of session i, and if a link l maps to an edge $e \in \tilde{C}_i$, then all other edges that link l maps to must also belong to $\tilde{C}_i$. Hence, the subset C_i constructed above must be a min-cut of the sub-network $G(V, L_i)$ of session i where the capacity of each link l is given by $\tilde{r}_i^l$. Let $\mathrm{Rank}_i(\tilde{\vec{r}}_i(t))$ denote the rank collected by destination d_i for session i. Then with high probability, $\mathbf{MCMF}_i(\tilde{\vec{r}}_i(t)) = \mathrm{Rank}_i(\tilde{\vec{r}}_i(t))/D_i$. We can then form the vector $\tilde{\vec{\mu}}_i(t)$ as in (4), and calculate a vector $\partial \tilde{F}(\vec{r}(t))$ such that its (i,l)-th component is given by:

$$[\partial \tilde{F}(\vec{r}(t))]_{il} = U_i'(\mathrm{Rank}_i(\tilde{\vec{r}}_i(t))/D_i)\tilde{\mu}_i^l(t). \tag{9}$$

Note that this vector $\partial \tilde{F}(\vec{r}(t))$ can be viewed as an approximation to the subgradient $\partial F(\vec{r}(t))$ given in (5). Finally, we replace $\partial F(\vec{r}(t))$ by $\partial \tilde{F}(\vec{r}(t))$ in the subgradient-ascent iteration (6), i.e.,

$$\vec{r}(t+1) = [\vec{r}(t) + \alpha \partial \tilde{F}(\vec{r}(t))]_\Lambda. \tag{10}$$

Let $D = \min_i D_i$, which is related to the generation size n according (7). Note that $1/D$ represents the loss of capacity due to granularity of the unit-capacity network model. Although the vector $\partial \tilde{F}(\vec{r}(t))$ is no longer a true subgradient of the objective function $F(\vec{r})$ at $\vec{r}(t)$, the following proposition shows that, if D is large and if the probability with which the coded-feedback method returns a min-cut of the unit-capacity sub-network is close to 1, then the above iteration will result in rate allocation that is close to optimal. Let F^* denote the optimal value of (2).

Proposition 3: For any $\epsilon > 0$, there exist $\alpha_0, \delta_0, D_0 > 0$ such that for all $\alpha \le \alpha_0$, $\delta \le \delta_0$ and $D \ge D_0$, if the coded-feedback algorithm can output the min-cut for the unit-capacity sub-network of each session i with probability no less than $1-\delta$, then starting from any initial rate-allocation vector $\vec{r}(0)$, the iteration (10) will produce rate-allocation vectors that satisfy the following:

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} \mathbf{E}[F^* - F(\tilde{\vec{r}}(t))] \le \epsilon.$$

V. Conclusions

We conclude this paper by discussing the benefits of the proposed coding-based primal rate-control algorithm.

A. Controlling the Queue Length

In our simulations (available in [11]), our proposed algorithm demonstrates similar convergence speed as the back-pressure algorithm [5]. On the other hand, a key advantage of our proposed solution with coded-feedback is that the queue-length can be tightly controlled. Specifically, the required buffer size for each session is of the order $\mathcal{O}(n)$ as we need to store roughly only packets from the same generation. For comparison, the queue-length dynamics under the back-pressure algorithm depend on many factors, such as the step-size, the network topology and the traffic pattern. Hence, the queue-length (and correspondingly the packet delay) is much harder to control.

B. Quick Start

For the back-pressure algorithm, one needs to build up queues before packets can be routed to the destination. Once the queues have been built up for exsiting sessions, a new session needs to build up its own queues (over the entire network) in order to compete and grab a fair share of the network resources. For comparison, our algorithm enjoys a "quick start" feature. Namely, if a link l is shared by I^l sessions, we can simply let each competing session grab $1/I^l$ share of the available capacity R^l as the initial rate-allocation. Network coding guarantees that each session can achieve at least $\frac{1}{\max_l I^l}$ of the optimal rate *at the very beginning of the iteration.*

References

[1] S.-Y. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inform. Theory*, vol. 49, no. 2, pp. 371–381, February 2003.

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

[3] D. S. Lun, N. Ratnakar, M. Médard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, , and F. Zhao, "Minimum-cost multicast over coded packet networks," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2608–2623, June 2006.

[4] A. Goldberg and R. Tarjan, "A new approach to the maximum-flow problem," *Journal of the ACM*, vol. 35, no. 4, pp. 921–940, October 1988.

[5] L. Tassiulas and A. Ephremides, "Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, December 1992.

[6] Y. Wu and S.-Y. Kung, "Distributed Utility Maximization for Network Coding Based Multicasting: A Shortest Path Approach," *IEEE J. Select. Areas Commun.*, vol. 24, no. 8, pp. 1475–1488, August 2006.

[7] A. Khreishah, C.-C. Wang, and N. Shroff, "An optimization-based rate control for communication networks with inter-session network coding," in *Proc. 27th IEEE Conference on Computer Communications (INFOCOM)*. Phoenix, USA, April 2008.

[8] Y. Wu, M. Chiang, and S.-Y. Kung, "Distributed utility maximization for network coding based multicasting: A critical cut approach," in *Proc. 2nd Workshop on Network Coding, Theory, & Applications (NetCod)*. Boston, Massachusetts, April 2006.

[9] Y. Xi and E. M. Yeh, "Distributed Algorithms for Minimum Cost Multicast with Network Coding," in *43rd Allerton Annual Conference on Communication, Control, and Computing*, Monticello, IL, September 2005.

[10] C.-C. Wang, "Pruning network coding traffic by network coding — a new max-flow algorithm," in *Proc. IEEE Int'l Symp. Inform. Theory*. Toronto, Canada, July 2008.

[11] C.-C. Wang and X. Lin, "Fast Resource Allocation for Network-Coded Traffic — A Coded-Feedback Approach," *Technical Report, Purdue University, http://min.ecn.purdue.edu/∼linx/papers.html*, 2008.

[12] P. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. 41st Annual Allerton Conf. on Comm., Contr., and Computing*. Monticello, IL, October 2003.