Pruning Network Coding Traffic By Network Coding — A New Max-Flow Algorithm

Chih-Chun Wang
Center for Wireless Systems and Applications (CWSA)
School of Electrical and Computer Engineering
Purdue University, West Lafayette, IN 47906

***Abstract*— For a unicast/multicast session, network coding is generally implemented via random coding and broadcasting at intermediate nodes, an especially favorable solution in wireless networks. The corresponding broadcast traffic can be minimized by finding the max (s,t)-flow of the network and by sending packets only along the max flow. All existing algorithms for this task, e.g. the push-&-relabel algorithm, are based on the routing paradigm and do not take advantage of the network coded traffic, which undermines their practicality for network coding applications.**

This work provides the first network-coding-based max (s,t)-flow algorithm starting from broadcasting the coded packets and followed by continuously trimming the unnecessary traffic. The running time is $\mathcal{O}(|V|^2)$, asymptotically no slower than that of the existing routing-based, distributed algorithms. Other important features include: (i) Limited exchange of control packets enables straightforward distributed implementation. (ii) The source-sink pair sustains uninterruptedly the optimal max-flow rate throughout iterations. (iii) All control packets are sent in the opposite direction of the data packets and can be readily piggybacked to the acknowledgment packets with minimal overhead. (iv) The control packet contains only easily computed coding coefficients and the algorithm imposes no additional requirement on the intermediate nodes nor on the format of each packet.

I. Introduction

For a unicast/mulitcast session, it is shown in [1] that linear network coding is capable of achieving the optimal *max-flow rate*. Many practical schemes have since been developed to realize the promised throughput gain for both wireline [2] and wireless networks [3]. The main ideas behind all practical schemes consist of using random network coding for its distributedness [4] plus tagging the coded content with the associated coding coefficients. To minimize the network usage of network coding, the user has to run the distributed max-flow algorithm, such as the push-&-relabel algorithm [5] and the references in [6], to decide the max (s,t)-flow, along which the coded packets will be sent from the source s to the destination t (see [2]).

Existing max (s,t)-flow algorithms are based on the routing paradigm, in which the packets (or equivalently the flows/pre-flows) cannot be mixed with each other. The value of the flow is then "augmented" monotonically by ingeniously pushing/pulling the flow/pre-flow in each link while obeying the given capacity constraints. This direct analogy between sending/retracting non-coded packets versus pushing/pulling flows/pre-flows prompts straightforward implementation when the max-flow algorithm is built around the *multi-path routing* traffic. On the other hand, network coding packets are constantly mixed with each other and the corresponding pushing/pulling operations are not well defined. Therefore, to find the max flow, the user generally has to send "virtual flows" in addition to the network coding traffic, which restricts its practicality due to the associated overhead and complexity. An alternative solution is that instead of finding the max flow beforehand, each intermediate node simply *broadcasts* the coded packets under the rough guidance of probability-based heuristics and routing tables [3]. Nevertheless, once the broadcast pattern is decided, there is little feedback whether the traffic along a given edge is really necessary or is carrying redundant information. A suitable max-flow algorithm is in great need.

Another benefit of a new max (s,t)-flow algorithm for network coding is that although multi-path routing can be made as throughput efficient as network coding in a unicast session, the routing-based max (s,t)-flow algorithms take an overly conservative approach. The flow value is gradually augmented from zero to the optimal value. Therefore (s,t) pair *cannot transmit at the optimal rate before the max flow is found*, which generally takes too long to converge, usually $\mathcal{O}(|V|^2)$ where $|V|$ is the number of network nodes. In contrast, network coding achieves the optimal rate with *minimal delay*[1] since the network is aggressively loaded by broadcasting randomly coded packets, which is made possible since network coding eliminates the need of deciding which packet to use which edge. The drawback is that now the entire network is occupied by the network coding traffic and no other traffic can share the network.[2] A network-coding-based max-flow algorithm enables optimal transmission rate with minimal delay while mitigating its drawback by *gradually trimming/cancelling the network-coding flow without interrupting*

[1]The delay before achieving the optimal rate (defined as the number of independent packets per second) is minimal in the sense that it is the same as if the max flow is given by an oracle beforehand and the packets are "routed" along the max flow from the very beginning.

[2]For network coding, new coded packets are generated whenever there is transmission opportunity available. No packet is a "must-transmit" since missing any coded packet incurs no penalty of retransmission. Therefore, *the broadcast coded packets do not need to be queued for a slow link* and any other users can access the same link easily and fairly via time-sharing. The impact of broadcast in network coding is thus relatively minor when compared to flooding in the routing paradigm. However, for several users to *share the network efficiently*, each source-sink pair still has to find the max flow and unlock unnecessarily-used resources of the network.

978-1-4244-2571-6/08/$25.00 ©2008 IEEE

the forward data traffic. In contrast with the routing-based approaches of "tentative routing solutions" versus "path augmentation," the proposed algorithm is based on "full-strength network *coding* solutions" versus "traffic *cancellation*" and provides the first balancing measure against the broadcast of coded packets. This coding-cancellation algorithm can thus unlock precious resources in a wireless network while still taking advantages of the broadcast nature of wireless media with the help of network coding.

Other features of the coding-cancellation algorithm include: (i) The running time is also $\mathcal{O}(|V|^2)$. (ii) Limited exchange of control packets enables straightforward distributed implementation. (iii) All control packets are sent in the opposite direction of the data packets and can be readily piggybacked to the acknowledgement packets with minimal overhead. (iv) The control packet contains only easily computed coding coefficients and the algorithm imposes no additional requirement on the intermediate nodes nor on the format of each packet.

II. Settings & Background

A. The Network

For directed cyclic networks with transmission delay, the time-axis naturally decouples the network into its acyclic counterpart as illustrated in [1]. Therefore, we consider directed acyclic graphs (DAGs) $G = (V, E)$ where each edge is assigned a unit capacity and *sending one packet along any edge takes one time unit*, say, one second. Links with high capacity are modelled as parallel edges and links with long delay are modelled as long paths. We use $\mathrm{In}(v)$ and $\mathrm{Out}(v)$ to denote all edges entering/leaving node v. A *flow* is a subgraph of $F \subseteq G$ such that each intermediate node v other than s and t has $|\mathrm{In}_F(v)| = |\mathrm{Out}_F(v)|$. The value of a flow is $|\mathrm{Out}_F(s)|$. A max flow is a flow with the maximum flow value. Let MFV denote the value of the max (s,t)-flow. Without loss of generality, we can have the following assumptions. The vertices are labelled from 1 to $|V|$ and a directed edge $(u, v) \in E$ exists only if $u < v$. The source node $s = 1$ and the destination node $t = |V|$.

B. Linear Algebra

We briefly review several definitions and one lemma in linear algebra. Throughout this paper, we use Ω to denote the n-dimensional vector space in $\mathsf{GF}(q)$. Uppercase letters $A, B, C \subseteq \Omega$ correspond to the linear (sub-)spaces while lowercase letters $a, b, c \in \Omega$ correspond to individual vectors. If used in a matrix form, the vectors a, b, c are considered as row vectors. The inner product of $a, b \in \Omega$ is defined as $a \cdot b = ab^{\mathrm{T}} = \sum_{i=1}^{n} a_i b_i$ where all operations are in $\mathsf{GF}(q)$ and b^{T} is the transpose of b. $\langle a, b\rangle$ denotes the linear space spanned by a, b.

The key definition in this work is as follows.

Definition 1: For any subspace $B \subseteq \Omega$ and $a \notin B$, $a_{\models B}$ represents any $c \in \Omega$ such that $a \cdot c = 1$ and $\forall b \in B, b \cdot c = 0$.

Lemma 1: For all $a \notin B$, $a_{\models B}$ always exists (although may not be unique).

Proof: We use its basis $\{b_1, \cdots, b_k\}$ to describe the linear space B and let $x = a \notin B$. The definition of $a_{\models B}$ is equivalent to the following equations that

$$Y^{\mathrm{T}} \triangleq \left(a^{\mathrm{T}}, b_1^{\mathrm{T}}, \cdots, b_k^{\mathrm{T}}\right)$$
$$Yx^{\mathrm{T}} = (1, 0, \cdots, 0)^{\mathrm{T}}. \quad (1)$$

Since there are n variables in x, k constraints in (1) and matrix Y is of full row rank, such x always exists (although may not be unique). ■

Constructing $a_{\models B}$ from a and B can be efficiently achieved by solving the linear equation in (1). For readers' possible interest, $a_{\models B}$ is closely related to the pseudo-inverse of Y.

III. The Algorithm

We consider the following *time-invariant* network coding schemes. Source s sends continuously packets $\{X_{i,\tau}\}_{i=1,\cdots,|\mathrm{Out}(s)|}$ along edges $\mathrm{Out}(s)$ to its neighbors $\{v_i\}_{i=1,\cdots,|\mathrm{Out}(s)|}$ in the τ-th second.

In the practical network coding scheme [2], the incoming packets are buffered and packet mixing is allowed among packets of the same time stamp (also known as "generations"). See [2] for details. Let $n \triangleq |\mathrm{Out}(s)|$. The coding vector for each edge, denoted by $m(u, v)$, is a n-dimensional vector in $\mathsf{GF}(q)$. The i-th component corresponds to the coefficient with respect to $X_{i,\tau}$. For example, $m(s, v_i)$ is a delta vector δ_i with all but the i-th component being zero and the i-th component being one. Each node $v \notin \{s, t\}$ selects an $|\mathrm{Out}(v)| \times |\mathrm{In}(v)|$ coding coefficient matrix $\Gamma(v) = (\gamma_{i,j})$ in $\mathsf{GF}(q)$. For each $(v, w_i) \in \mathrm{Out}(v)$, the incoming packets of the same generation are mixed such that $m(v, w_i) = \sum_{j=1}^{|\mathrm{In}(v)|} \gamma_{i,j} m(u_i, v)$. *The same mixing coefficients $\Gamma(v)$ will be used throughout the entire session unless modified by the coding-cancellation algorithm.* Detailed description of the proposed new max (s,t)-flow algorithm is as follows.

§ The Coding-Cancellation (CC) Algorithm

1: Each node $v \notin \{s, t\}$ selects randomly a $|\mathrm{Out}(v)| \times |\mathrm{In}(v)|$ coding coefficient matrix $\Gamma(v) = (\gamma_{i,j})$.
2: Every second, generate $m(v, w_i) = \sum_{j=1}^{|\mathrm{In}(v)|} \gamma_{i,j} m(u_j, v)$.
3: Wait until $m(\cdot, \cdot)$ along the network are stabilized.
4: Set $v \leftarrow t = |V|$.
5: **while** $v > 1$ **do**
6: **if** $v = t = |V|$ **then**
7: Choose $E_t \subseteq \mathrm{In}(t)$ such that $m(\cdot, \cdot)$ in E_t form a basis of $\langle m(u_j, t) : (u_j, t) \in \mathrm{In}(t)\rangle$.
8: Remove edges in $\mathrm{In}(t) \backslash E_t$.
9: $\forall (u_j, t) \in E_t$, set

$$q(u_j, t) \leftarrow m(u_j, t)_{\models \langle m(u,t):(u,t)\in E_t \backslash (u_j,t)\rangle}.$$

10: Set $v \leftarrow |V| - 1$.
11: **else**
12: Construct an $|\mathrm{Out}(v)| \times |\mathrm{In}(v)|$ matrix $(\alpha_{i,j})$ with $\alpha_{i,j} = q(v, w_i) \cdot m(u_j, v)$.
13: Choose an $E_v \subseteq \mathrm{In}(v)$ with $|E_v| = |\mathrm{Out}(v)|$ such that the square matrix $(\alpha_{i,j})_{i\in\mathrm{Out}(v), j\in E_v}$ has full rank.
14: **if** $E_v \neq \mathrm{In}(v)$ **then**
15: Remove edges in $\mathrm{In}(v) \backslash E_v$.

16: If the $\Gamma(v)$ of the new graph is not of full rank, replace $\Gamma(v)$ by a full rank matrix Γ'.
17: Wait until all new coding vectors $m'(\cdot,\cdot)$ resulted from the edge removal and from the new Γ' are stabilized.
18: Set $v \leftarrow t = |V|$.
19: **else**
20: $\forall (u_j, v) \in \mathrm{In}(v)$, set

$$q(u_j, v) \leftarrow \sum_{i=1}^{|\mathrm{Out}(v)|} \gamma_{i,j} q(v, w_i).$$

21: Set $v \leftarrow v - 1$.
22: **end if**
23: **end if**
24: **end while**
25: **return** the remaining graph.

The correctness and complexity analysis of the CC algorithm rely on the following theorems.

Theorem 1: The following properties hold for the CC algorithm.

1) Such E_v in Line 13 always exists although the choice is not unique.
2) The algorithm stops in no longer than $2l|V|$ seconds where l is the length of the longest s-t path. Before the algorithm stops, at most $|V||E|$ feedback messages $q(y,z)$ are generated and sent.
3) Throughout the while-loop in Line 5, the dimension of $\langle m(y,t) : (y,t) \in \mathrm{In}(t) \rangle$ remains identical.
4) The remaining graph in Line 25 is a flow.

As proven in [4], with high probability, the random network coding in Line 1 will have the dimension of $\langle m(y,t) : (y,t) \in \mathrm{In}(t) \rangle$ equal MFV. The Properties 3 and 4 in Theorem 1 then guarantees that a max (s,t)-flow is found. Property 3 ensures further that the optimal transmission rate remains uninterrupted during the entire session.

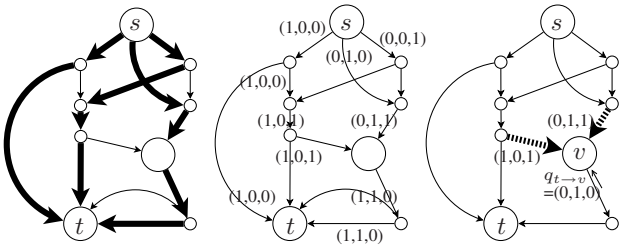
We also note that the choices of $q(u_j, t)$, E_v, and $\{\gamma'_{w_i,u_j}\}$ in Lines 9, 13, and 16 are not unique. The flexibility of the proposed algorithm is then stated in the following theorem.

Theorem 2: Suppose the initial random network coding is a *generic linear-code multicast* as defined in [1]. For any max (s,t)-flow, a sequence of choices of $q(u_i, t)$, E_v, and Γ' can be made such that the output graph is the given max flow.

A. An Illustrative Example

We use Fig. 1 to demonstrate the basic principle behind the CC algorithm. Fig. 1(a) represents the underlying network for which the max flow is illustrated by thick arrows and the corresponding max-flow value is three. Fig. 1(b) shows a possible network coding solution in GF(2) where packets are broadcast over the entire network and each 3-dimensional vector is the corresponding coding coefficients along a given edge. As can be seen in Fig. 1(b), destination t receives two identical messages from its right neighbor and one of them can be safely trimmed as in Line 8 of the CC algorithm.

The real challenge arises when node v wants to trim one of its two incoming edges (see Fig. 1(c)). As v has no global information of the graph topology, its two incoming



(a) A directed acyclic graph & its max flow. (b) An intrasession network coding solution (c) Trimming Network Coding Traffic

Fig. 1. Finding good paths/flows by network coding.

messages $(1,0,1)$ and $(0,1,1)$ are indistinguishable. To help v decide which edge to trim, destination t sends a coded feedback $q_{t\to v} = (0,1,0)$ which is orthogonal to the other two messages $(1,0,0)$ and $(1,0,1)$ received by t. This construction of orthogonal $q_{t\to v}$ is defined in Line 9 of the CC algorithm.

By comparing the inner products of $q_{t\to v}$ and the two messages on the edges of $\mathrm{In}(v)$, node v knows that its left message is orthogonal to $q_{t\to v}$. Therefore, the left incoming message of v must be a linear combination of the two other messages received by t, which is thus redundant from the destination's perspective. Node v can then safely remove the left incoming edge, which is indeed a redundant edge as can be seen as a thin arrow in Fig. 1(a).

In the CC algorithm, Lines 12 and 13 formalize the idea of comparing the inner products of the forward and feedback messages for the general scenarios. Line 20 focuses on how to propagate the feedback messages $q(\cdot,\cdot)$ along the network so that the "orthogonality" can be preserved after many hops of transmissions. Line 16 deals with the exception that the original coefficient matrix $\Gamma(v)$ was not chosen properly. The old $\Gamma(v)$ is then replaced by a new Γ' that has full rank.

B. Variant 1: Controlled Broadcast

We first notice that even in the (relatively infrequent) cases that the initial random network coding achieves only a near-optimal $r <$ MFV rate, one can still apply the CC algorithm to prune the unnecessary edges while sustaining the near-optimal rate r. This is a highly desired feature for practical systems. For example, with the delay and complexity requirements, a user is likely to be interested only in (s,t) paths that use fewer than h hops. He/she can use a controlled broadcast + network coding that explores only paths of length $< h$. The optimal rate MFV may not be attainable under this partial exploration. The CC algorithm allows the user to still achieve the best possible rate r and prune the redundant traffic. Varying the h value thus serves as an additional degree of freedom in network design. Namely, how widely the (s,t) communication is going to spread on the network.[3] The CC algorithm provides a way to harness the h-controlled broadcast by pruning the unnecessary traffic.

[3]If h is chosen as the minimal distance from s to t, the controlled broadcast is no different than the routing-based shortest-path discovering.

C. Variant 2: Distributed Implementation

The CC algorithm can be made distributed in a straightforward manner. The feedback messages $q(\cdot,\cdot)$ can be easily carried in a network. Deciding which v to perform graph-trimming (as in Line 15) can be implemented by a token-based system. A single token ensures that one and only one v is allowed to stop its incoming traffic,[4] a key ingredient for the correctness of the CC algorithm. Passing the token along with the data / acknowledgement packets enforces sufficient waiting time for $m(\cdot,\cdot)$ and $q(\cdot,\cdot)$ to stabilize.

An even more decoupled scheme is that each v that is eligible to graph-trimming (those with $E_v \subsetneq \text{In}(v)$) just waits a random number of seconds before trimming the graph. A carefully designed random waiting time (with proper reset mechanisms) ensures that in high probability, no two nodes trim the graph simultaneously and the waiting time for $m(\cdot,\cdot)$ and $q(\cdot,\cdot)$ to stabilize is sufficient. Even when two nodes perform graph-trimming in too short a time period, the rank of $\langle m(u,t) : (u,t) \in \text{In}(t)\rangle$ decreases only if the pruned edges happen to be in a minimum cut, a rare event in a random network [7]. In summary, a very small probability that the dimension of the received space may decrease slightly is a welcome compromise for a perfectly distributed algorithm.

D. Comparison to Existing Work & Simulation

The following discussion and simulation assumes that exchanging coding vectors or feedback messages along a given edge takes one second and the matrix operations take a negligible amount of time. Since the length of any path is upper bounded by $|V|$, the complexity in terms of the convergence time is $\mathcal{O}\left(|V|^2\right)$ based on Theorem 1. In terms of the total number of feedback messages exchanged in the network, the complexity is $\mathcal{O}\left(|V||E|\right)$.

Remark: Sending the coding vectors in the forward direction does not incur any communication cost since $m(u,v)$ is embedded in the forward traffic. Only the feedback message may have caused additional traffic, which in many cases can be piggybacked naturally to the acknowledgement packet and has much smaller impact than the $\mathcal{O}\left(|V||E|\right)$ upper bound.

We use the generic push-&-relabel (P&R) algorithm [5] as benchmark, which also facilitates distributed network implementation. The distributed P&R algorithm converges in $\mathcal{O}\left(|V|^2\right)$ seconds and is asymptotically not faster than the proposed CC algorithm. In terms of the amount of network usage, a generic P&R requires $\mathcal{O}(|V|^2|E|)$ push operations plus $\mathcal{O}(|V||E|)$ packets to disseminate the "label" information, both of which are no less than the $\mathcal{O}(|V||E|)$ feedback messages in the CC algorithm.

Fig. 2(a) contains the time evolution of the network usage and the dimension of the received space with respect to t when the CC algorithm is applied to a randomly generated sparse 30-node sparse DAG, of which each node is connected to 5.2 links in average. The capacity of each link is randomly chosen from 1 to 10. The max flow value is $\mathsf{MFV} = 13$ and the largest rate of a single-path route is 7. Broadcast + network coding achieves the MFV rate in 7 seconds while the max flow converges in 116 seconds. If we run the distributed P&R[5] on the same graph, it requires 18 seconds before a non-zero transmission rate can be established. The optimal rate can be reached in 36 seconds,[6] but the max flow will not converge until the 487-th second. The push/pull approach of the P&R algorithm results in the oscillating nature of the network usage, as seen in Fig. 2(b). Not only do we have a 30% network usage oscillation throughout iterations, the P&R algorithm *continuously redirects the traffic along different edges as an effort to find the max flow of the network.* In order to implement this traffic redirection, each node has to keep establishing and releasing connections between its neighbors, which imposes significant control overhead for the network. For comparison, the traffic trimmed by the CC algorithm will never be added back to the network. Both the CC and P&R algorithms output max flows that use roughly 1/4 of the total 314 edges.

As can be seen in the evolution of network usage in Fig. 2(a), there is an initial spike of network usage due to the aggressive broadcast in the beginning stage. One can use *controlled broadcast* to mitigate this initial network usage spike. *The tradeoff is that the CC algorithm may not be able to identify the optimal max flow due to the more self-restrained initial broadcast.* However, as discussed previously, the CC algorithm can still identify the best rate under this suboptimal scenario.

IV. A Central Lemma for the CC Algorithm

Due to the limit of space, we provide a central lemma that demonstrates the special relationship between the forward and feedback messages. This lemma will be used extensively when proving Theorems 1 and 2.

Define $R \triangleq \langle m(x,t) : (x,t) \in \text{In}(t)\rangle$, $\kappa \triangleq \dim(R)$, and without loss of generality, we assume $|\text{In}(t)| = \kappa$ as also suggested/enforced in Line 8. Let $\left\{\overset{\rightsquigarrow}{v}\right\}$ denote the vertices reachable from v including v itself. A non-empty vertex set $Z \subseteq V$ is defined as a *separating* vertex set if $\forall z \in Z, \left\{\overset{\rightsquigarrow}{z}\right\} \subseteq Z$. Define the edge collection C_Z for a separating Z:

$$C_Z \triangleq \{(y_i, z_i) \in E : y_i \notin Z, z_i \in Z\}.$$

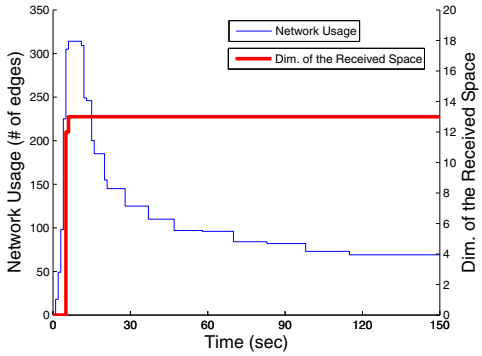
We first note that Line 20 is always executable even when $E_v \subsetneq \text{In}(v)$. We can then prove the following central lemma.

Lemma 2: Consider a modified version of the CC algorithm in which no graph-trimming will be performed. The modified CC algorithm only generates coded feedback $q(\cdot,\cdot)$ messages and propagates them based on Lines 20 and 21 without

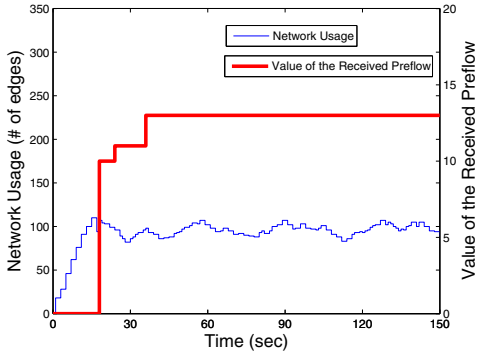
[4]Removing edges in the CC algorithm is equivalent to stopping the traffic in a real network.

[5]Greedy non-interfering scheduling is assumed, i.e. two neighboring nodes cannot perform the push/relabel operations simultaneously.

[6]Unlike the CC algorithm, it is still an open problem how to integrate the P&R algorithm with the forward data traffic so that data transmission is possible at rates equal to the tentative flow values before convergence.



(a) The coding-cancellation algorithm



(b) The distributed push & relabel algorithm

Fig. 2. The time evolution of the network usage & the dimension of the received space of the CC algorithm in a 30-node random graph.

performing any graph trimming. For any separating vertex set Z with $|C_Z| = \kappa$, the following equation must hold:

$$\begin{bmatrix} m(y_1, z_1) \\ \vdots \\ m(y_\kappa, z_\kappa) \end{bmatrix} \begin{bmatrix} q(y_1, z_1) \\ \vdots \\ q(y_\kappa, z_\kappa) \end{bmatrix}^{\mathrm{T}} = I_\kappa,$$

where I_κ is an identity matrix in $\mathsf{GF}(q)$ with dimension κ, assuming all $m(\cdot,\cdot)$ and $q(\cdot,\cdot)$ in the network are stabilized.

This lemma shows that the orthogonality between $m(\cdot,\cdot)$ and $q(\cdot,\cdot)$ can be preserved by the propagation rules of $q(\cdot,\cdot)$ in Lines 20 and 21. This orthogonality serves an important role when deciding the redundant messages by comparing the products of the $m(\cdot,\cdot)$ and $q(\cdot,\cdot)$.

Proof: By the algebraic framework of network coding in [8], we have

$$\begin{bmatrix} m(x_1, t) \\ \vdots \\ m(x_\kappa, t) \end{bmatrix} = A \begin{bmatrix} m(y_1, z_1) \\ \vdots \\ m(y_\kappa, z_\kappa) \end{bmatrix}, \tag{2}$$

where A is a $\kappa \times \kappa$ transfer matrix from the cut C_Z to $\mathrm{In}(t)$. Since the messages in $\mathrm{In}(t)$ are independent, the matrix A must be invertible.

Line 20 ensures that the same mixing coefficients $\Gamma(v)$ that are used to construct $m(\cdot,\cdot)$ are also used to construct the feedback messages $q(\cdot,\cdot)$ with the input/output roles swapped. Therefore

$$\begin{bmatrix} q(y_1, z_1) \\ \vdots \\ q(y_\kappa, z_\kappa) \end{bmatrix} = A^{\mathrm{T}} \begin{bmatrix} q(x_1, t) \\ \vdots \\ q(x_\kappa, t) \end{bmatrix}. \tag{3}$$

The construction of the feedback messages $q(x_1, t), \cdots, q(x_\kappa, t)$ in Line 9 ensures that

$$\begin{bmatrix} m(x_1, t) \\ \vdots \\ m(x_\kappa, t) \end{bmatrix} \begin{bmatrix} q(x_1, t) \\ \vdots \\ q(x_\kappa, t) \end{bmatrix}^{\mathrm{T}} = I_\kappa. \tag{4}$$

Jointly, (2–4) and the invertibility of A prove this lemma. ■

V. Conclusion & Future Directions

A new coding-cancellation max (s,t)-flow algorithm is provided for directed acyclic networks, which trims the network coding traffic by network coding. The algorithm is asymptotically no slower than the existing algorithms, admits straightforward distributed network implementations, imposes no additional requirement on intermediate nodes, and sustains the optimal transmission rate even before the algorithm converges. Two major future directions are discussed as follows.

1) [9] shows that the min-cut values for several $\{(s, t_i)\}$ can be found simultaneously without running the max-flow algorithm for each t_i separately. Similarly, we have generalized the CC algorithm for when multiple destinations are considered. The detailed work will be presented in a companying paper.
2) The coding-cancellation scheme provides a counter-measure to harness the power of broadcast. We are interested in the dynamics between this pair of opposite mechanisms. Based on the concept of broadcast versus coded feedback, a new rate-control/scheduling framework will be developed for wireless networks in the setting of multiple unicast sessions and non-stationary, time-varying networks.

References

[1] S.-Y. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inform. Theory*, vol. 49, no. 2, pp. 371–381, Feb. 2003.
[2] P. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. 41th Annual Allerton Conf. on Comm., Contr., and Computing*. Monticello, IL, Oct. 2003.
[3] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *Proc. SIGCOMM 2007*. Kyoto, Japan, Aug. 2007.
[4] 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, April 2006.
[5] A. Goldberg and R. Tarjan, "A new approach to the maximum-flow problem," *Journal of the ACM*, vol. 35, no. 4, pp. 921–940, Oct. 1988.
[6] A. Goldberg, "Recent developments in maximum flow algorithms," NEC Research Institute, Inc., Technical Report 98-045, 1998.
[7] D. Karger and C. Stein, "A new approach to the minimum cut problem," *Journal of the ACM*, vol. 43, no. 4, pp. 601–640, July 1996.
[8] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, pp. 782–795, October 2003.
[9] X. Hao and J. Orlin, "A faster algorithm for finding the minimum cut in a graph," in *Proc. 3rd ACM–SIAM Symp. on Discrete Algorithms (STOC)*, 1992, pp. 165–174.