Pruning Network Coding Traffic By Network Coding — A New Class of Max-Flow Algorithms

Chih-Chun Wang, *Member, IEEE,*

***Abstract*—This work explores new graph-theoretic and algebraic behaviors of network codes and provides a new class of coding-based, distributed max-flow algorithms. The proposed algorithm starts from broadcasting the coded packets, followed by continuously trimming the redundant traffic that does not constitute the *maximum flow* of the network information. The convergence speed of the proposed algorithms is no slower than that of the existing push-&-relabel max-flow algorithms. The algorithmic results in this work also possess several unique features that are especially suitable for practical network implementation with low control, communication, and complexity overhead.**

***Index Terms*—Coded-feedback, network coding, network multicast, max flow algorithms, multi-path routing, random linear coding.**

I. Introduction

It is shown in [21] that linear network coding within a single unicast or multicast session is capable of achieving the optimal min-cut/max-flow rate. This work focuses solely on coding within a single unicast or multicast session, termed *intrasession network coding*. The more general form of coding across multiple sessions [32] is beyond the scope of this paper. Many practical intrasession-coding schemes have since been developed to realize the promised throughput gain for both wireline [3] and wireless networks [2]. The main ideas behind many practical schemes consist of using random linear network coding for its distributedness [14] plus tagging the coded content with the associated coding coefficients. To minimize the network usage, the user generally has to run a distributed max-flow algorithm in order to search for the minimal subnetwork (the max-flow) that can support the desired transmission rate, along which the packets will be sent [3]. A low-complexity max-flow algorithm is thus critical to the random-coding-based network protocols.

A. Existing Max-Flow Search Algorithms

In addition to its network coding applications, finding the *maximum flow* of a given graph has been a classical optimization problem with many other practical applications since early 1960s [10]. There are at least two types of max-flow algorithms: one is the designated, graph-theoretic algorithms such as the Ford-Fulkerson algorithm, push-&-relabel (P&R) algorithm [11], and the references in [10]. The second type is the generic linear programming (LP) approaches [33], [36]. Both the P&R algorithm and the LP approaches admit distributed implementation and can be readily applied to computer networks. Nonetheless, each approach has its own disadvantages when applied to network communications, especially for network coded traffic.

Take the P&R algorithm as a representative of the graph-theoretic max-flow algorithms. The P&R algorithm monotonically "augments" the flow value by ingeniously pushing and pulling the flow (or pre-flow) in each edge while obeying the given capacity constraints [11]. As a designated max-flow algorithm, the P&R algorithm is generally much faster than the generic LP approaches. However, in practice, the control messages of the P&R algorithm, containing the *preflow* values and the *label* information, must be sent in parallel with the (network coded) data traffic, which incurs additional control and communication overhead. As a specialized algorithm finding the maximum flow between a single source s and a single destination d, it is also less flexible to take into account other objective functions such as energy or cost minimization and multicast traffic from a single source s to multiple destinations $\mathbf{d} = \{d_i\}$.

Based on generic optimization techniques, the LP approach is suitable for various objective functions other than maximizing the flow value. Common objective functions for the LP approaches include cost or energy minimization and utility maximization [27], [30], [33], [34], [36]. Many of the LP approaches solve the primal, dual, or primal-dual optimization problem by exchanging the subgradient information (often being a function of the queue-length) among intermediate network nodes. To ensure convergence to the optimal value, a small step size is often used in the sub-gradient update, which is at the price of increasing the time to convergence [24], [27]. Most dual solutions, e.g., the back-pressure algorithm in [30], also rely on building up the queues and using the queue-length as the control message. Since the queues at the intermediate network nodes induce delay and the dynamics between the queue-length and the network traffic are difficult to predict and control [31], packet delay is also a concern for dual LP solutions. It is worth noting that the concept of queues stems from the routing paradigm, in which packets are not mixed with each other and each packet takes an exclusive share of the memory (the queue). To adapt to the network coded traffic for which packets are mixed together, the concept of queues often need to be generalized [28], which may further increase the complexity of the LP algorithms.

This work was supported by the NSF CAREER Award CCF-0845968.

Part of this paper was presented in IEEE International Symposium on Information Theory 2008 and in the 46th Annual Allerton Conference on Communications, Control, and Computing, 2008.

C.-C. Wang is with the Center for Wireless Systems and Applications (CWSA), School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906, USA. E-mail: chihw@purdue.edu.

Manuscript received in June, 2008; revised September, 2009.

B. Taking Full Advantages of Network Coding

Mixing packets at the intermediate nodes, network coded traffic is fundamentally different from the routing-based traffic and possesses many new properties that have not been exploited by the existing max-flow algorithms. For example, one unique feature of (intrasession) network coding is its quick-start capability. Consider an s to d unicast session. With the help of random linear network coding [14], the network can randomly mix and forward packets and the destination can decode the information and achieve the optimal min-cut/max-flow rate with minimal time lapse[1] even before the maximum flow is identified. The optimal-rate transmission can thus be achieved in the beginning of the unicast session without identifying the best paths (see [2] for the application of this property to opportunistic routing). For comparison, multi-path routing can be made as throughput efficient as intrasession network coding when only unicast sessions are considered, conditioning that the uncoded packets are transmitted along the max (s,d)-flow. With the knowledge about the max flow being essential to the achievable performance, the optimal rate of multipath routing cannot be attained before the max flow is found.

This paper exploits the linear dependence of network coded traffic and designs a new class of max-flow algorithms based on the concept of *coded feedback* (CF). Being a designated max-flow algorithm, the results are in parallel with the P&R algorithm. For comparison, instead of gradually augmenting the existing flow as in the P&R algorithm, the proposed CF algorithms take advantage of the quick-start feature of network coding. Explicitly, we first use random linear network coding to achieve the min-cut/max-flow information rate with minimal time lapse. We then gradually trim the network-coded traffic without interrupting the forward data traffic until the remaining traffic becomes a max-flow. The CF algorithms also admit distributed implementation. The communication delay before the convergence of the distributed CF algorithms is proven to be $\mathcal{O}(|V|^2)$ where $|V|$ is the number of nodes in the graph, which is no slower than that of the classic P&R algorithm.

In addition to searching for the maximum flow for a unicast session, the CF algorithms handle the multicast scenario as well. Specifically, since the largest throughput gain of intrasession network coding exists in the multicast scenario, a natural graph-theoretic question of network coding is to identify the minimal subgraph that sustains the same min-cut/max-flow values (from s to each destination d_i respectively) as the original graph. When there is only one destination d, the minimal subgraph is simply the maximum flow, and the P&R algorithm solves the problem efficiently. However, when there are multiple destinations, the P&R algorithm does not apply as the minimal subgraph needs to take into account all destinations simultaneously.[2] Currently, there is no graph-theoretic solution that answers this question and one has to rely on the generic LP-based solution. With straightforward modification, the CF max-flow algorithms can simultaneously and efficiently search for the minimal subgraph that sustains the min-cut/max-flow values for all destinations.

The main contribution of this work is the novel algebraic study of the existing network coding framework [21] and the correctness and complexity analyses of the proposed graph-theoretic and coding-based algorithms. Our results illustrate that the subspace properties of the network coded traffic can be properly conveyed by using *feedback coding vectors*, which also facilitates decision making at the intermediate network nodes. Since our focus is on pure algorithmic coding analysis, several critical issues of implementation[3] are beyond the scope of this paper, e.g., network synchronization, construction of acyclic subnetworks, the generation flushing policy for practical network coding [3], and the interference or scheduling policy when implemented for wireless networks.

Taking further advantage of the network coded traffic, the proposed CF max-flow algorithms could have significant impact on resource allocation protocol design for network coded traffic. Specifically, the CF algorithms have several unique features that are suitable for practical network implementation, including (i) minimal control, communication, and complexity overhead, (ii) no interruption to the forward traffic, and (iii) enhanced flexibility for different design criteria and constraints. Detailed discussions about these features are presented in Sections IV to VI.

The remainder of this paper is organized as follows. Section II introduces the existing results on network code construction and network tomography based on network-coded traffic. Section III discusses the setting and some background on linear algebra and graph theory. A new class of max flow algorithms, termed the coding-cancelation algorithms, is introduced in Section IV and two specific constructions motivated by different philosophies will be discussed. Concrete examples are provided for illustration. The correctness and complexity results of the proposed algorithms will be stated in Section V while the proofs are provided in the appendices. Section VI discusses the low-complexity distributed implementations and generalizes the proposed algorithms for networks with multiple destinations, for sessions with delay constraints, and for low power consumptions. Numerical experiments are conducted and reported in Section VII. Section VIII concludes this paper.

II. Related Existing Works

A. Linear-Programming Solutions for Network Coded Traffic

Linear-programming-based max-flow algorithms cast the graph-theoretic problem of finding the maximum flow to a

[1]We are interested in when the network can start to sustain the largest rate of information exchange. Therefore, the time-lapse (before achieving the optimal rate) is defined as the lapse of time before the number of independent packets per second reaches the min-cut/max-flow rate. This time lapse is different from the decoding delay, which is due to a large *generation* size [3]. With this definition, the time lapse of network coding is minimal in the sense that it is the same as the time lapse of any oracle-assisted multi-path routing scheme for which the best paths (the max flow) are given by an oracle and the packets are directly “routed” along the given max flow.

[2]One obvious attempt is to take the union of the maximum flows for each (s,d_i) pair as suggested in [3]. Nonetheless, the union of max-flows only guarantees that the min-cut/max-flow values remain unchanged, but does not guarantee the minimality of the resulting subgraph.

[3]As an abstract algorithm, the results in this paper may be implemented either on the overlay networks or as a new network layer protocol for wireless networks.

linear optimization problem by imposing the flow-conservation constraints on each intermediate network node. Pioneered by the back-pressure algorithm in [30], the LP-based network solutions have been studied extensively for the routing-based traffic and become the corner stone of many cross-layer resource allocation algorithms [25]. As a generic network optimization scheme, the distributed LP solutions are fundamentally different from the designated max-flow algorithms, such as the P&R algorithm and the CF algorithm of this work. Both types of schemes identify the maximum flow, but in much different manners.

Take Fig. 1(a) for example. To find the max-flow between source s and destination d, a simple LP solution will solve the optimal values of the two rate variables $x_{s,v}$ and $x_{v,d}$ that satisfy the capacity and flow-conservation constraints. Initially, $(x_{s,v}, x_{v,d})$ is set to $(0, 0)$. Small increments $\beta\nabla F(x_{s,v}, x_{v,d})$ are then added to rate vector $(x_{s,v}, x_{v,d})$ throughout iterations where β is the step size and $\nabla F(\cdot,\cdot)$ is the gradient of the objective function. The convergence speed for this example is thus inversely proportional to the step size β (in order to reach the optimal $(x^*_{s,v}, x^*_{v,d}) = (25, 25)$). The larger the β, the faster the convergence. But using a large β has its own risk as the convergence of the LP is guaranteed only for a small β. In contrast, the P&R algorithm takes a different approach. In the first iteration (Fig. 1(b)), the P&R algorithm will exhaust the capacity of the (s, v) link by allocating 30 *preflow* for (s, v). In the second iteration (Fig. 1(c)), node v will push 25 more preflow along link (v, d). In the third iteration (Fig. 1(d)), the label of node v changes due to the saturation of the (v, d) link and 5 preflows are pushed back from v to s. The final result (Fig. 1(e)) is the maximum flow of value 25.

For comparison, the CF algorithm first lets nodes s and v to generate 30 and 25 randomly coded packets according to the maximum capacity. Based on the coding coefficients of the network coded traffic, coded feedback is computed at d and sent back to v, and then re-computed at v and sent back to s. Based on the coded feedback, the rate of link (s, v) will be reduced and trimmed from 30 to 25 in the next iteration and the max-flow is thus identified. This simple example demonstrates the algorithmic differences between the generic LP-based solution and the designated P&R and CF algorithms. The former has the flexibility of adapting to different objective functions $F(\cdot,\cdot)$ while the latter admits much faster convergence (only four information exchanges) with tractable convergence speed analysis that does not depend on the value of β [23]. The same example also illustrates the similarity between the two different designated max-flow algorithms. Throughout the paper, the P&R algorithm will thus be used as the main benchmark of the proposed CF algorithm while the LP solution will be compared in Section VII when examining numerically the efficiency of the CF algorithm.

It is worth pointing out that the LP solutions have been generalized for network-coded traffic in several existing works. One common foundation of these LP network coding solutions is the use of the min-cut/max-flow theorem [21] that abstracts the packet-by-packet behavior of network coding to a fractional-rate-based characterization, which is in contrast with the integer-based packet-by-packet operations of the CF algorithm in this work. Specifically, the distributed primal and dual LP solutions have been developed for various settings, including the minimum-cost broadcast for directed wireless networks [4], [15], [27], [28], [34], [35], [37], for directed wireline networks [15], [27], [33], [36], and the throughput optimization for undirected networks [22], [23].

Remark: The CF max-flow algorithms are not intended to replace the LP approach but to provide a new, efficient class of designated max-flow algorithms that complement and enrich the existing (network) optimization methodology. For example, it is shown in [22], [33] that using a designated max-flow algorithm as a subroutine can substantially enhance the efficiency of the general LP solutions. An explicit example of combining the CF-based algorithms and LP solutions for network resource allocation can be found in [31].

B. Network Code Construction & Network Tomography

The CF max-flow algorithms focus on the packet-by-packet coding operations and the corresponding subgraph construction problem for graphs with unit or with integer edge-capacity. Therefore, in a broad sense, it can also be viewed as a network code construction problem with the goal of minimizing the number of edges being used. Existing works on network code construction include the randomized, polynomial-time code construction algorithm [17], the binary vector network code construction [26], and the network code construction for cyclic networks [1]. For comparison, one unique feature of the CF-based code construction is to use *coded feedback* to convey the subspace information between network nodes, which was first exploited in [13]. The feedback coding vectors in [13] carry the subspace information of the destination, and the intermediate nodes simply relay the subspace information of the destination without any processing. The coded feedback of this paper further extends this concept and allows full processing capability at the intermediate node, which is essential to the novel fully distributed max-flow algorithms.

In addition to its application to network code construction, the subspace information of network coded traffic can also be used in network tomography and network monitoring. For example, by probing the network either actively or passively, the rank information of the packets at the destination node can be used to estimate the link loss rate or the network topology [6], [7], [9], [12]. Some common components of these works include the distinguishability analysis for different topological decompositions and the loss-rate or topology inference based on the maximum likelihood detector or other efficient inference methods, e.g., the belief propagation detector. [16] uses the rank growth curve (versus the progress of time) to distinguish different network topologies based on new sufficient conditions of distinguishability. For comparison, most existing works on network tomography and network monitoring exploit only the “one-way” subspace information and derive the network topology accordingly without any feedback. The CF max-flow algorithms, on the other hand, use coded feedback to carry the subspace information in the reverse direction. The subspace information (the coded feedback) is then used as

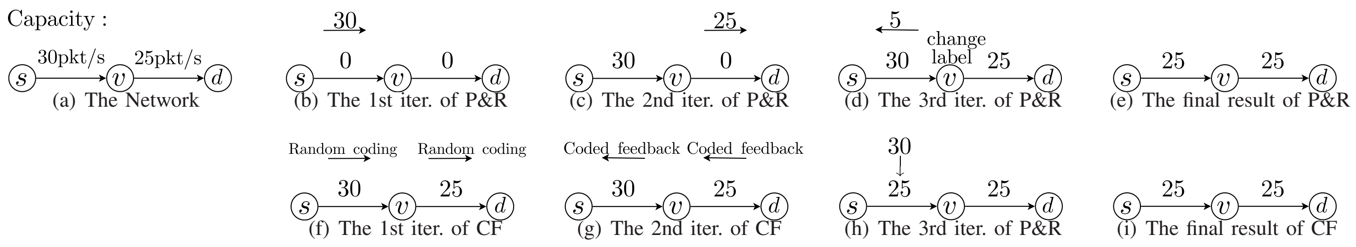


Fig. 1. A small example that illustrates the steps of the Push-&-Relabel (P&R) and the Coded Feedback (CF) algorithms.

a control message, which interactively modifies the network information flow and thus constantly changes the subspace of the network coded traffic throughout iterations. This network interaction is not exploited in the existing network tomography and monitoring literature.

The subspace property of network coding was also used for error correction in a non-coherent network environment [19], as in a non-coherent environment the network coded information is carried over the corresponding subspaces. Other examples that illustrate the benefits of feedback for network coded traffic can be found in [5]. [29] explores the reliability and throughput tradeoff by using the *training bits* to infer the global network coding matrix.

III. The Setting & Background

A. The Network

We consider directed acyclic graphs (DAGs) $G = (V, E)$ where each edge is able to carry one packet per unit time, say a second. Sending one packet along any edge also takes one second. High capacity links are modelled by parallel edges and long delay links are modelled by long paths.

We use $\mathrm{In}(v) \subseteq E$ and $\mathrm{Out}(v) \subseteq E$ to denote all edges entering and leaving node v, respectively. We consider either a single unicast session from source s to destination d in the network or a single multicast session from source s to destinations $\mathbf{d} = \{d_i\}$. Without loss of generality, we assume that $\mathrm{In}(s) = \emptyset$ and $\mathrm{Out}(d) = \emptyset$ (or $\mathrm{Out}(d_i) = \emptyset$ for all $d_i \in \mathbf{d}$). And we assume that $\mathrm{In}(v) \neq \emptyset$ and $\mathrm{Out}(v) \neq \emptyset$ for all $v \notin \{s, d\}$ (or $v \notin \{s\} \cup \{d_i\}$), otherwise node v has no impact on the network and can be removed from G.

Since the underlying network is acyclic, we can also assume the vertices are labelled from 1 to $|V|$ and a directed edge $(u, v) \in E$ exists only if $u < v$. Without loss of generality, we further assume the source node $s = 1$ and the destination node $d = |V|$ for the unicast scenario.

B. Graph-Theoretic Definitions

A *flow* is a subgraph of $F \subseteq G$ such that each intermediate node v other than s and d (or $d_i \in \mathbf{d}$) has the same numbers of incoming and outgoing edges $|\mathrm{In}_F(v)| = |\mathrm{Out}_F(v)|$. The value of a flow is $|\mathrm{Out}_F(s)|$. A max flow is a flow with the maximum flow value. Let $\mathsf{MFV}(s, d)$ denote the value of the max (s, d)-flow. A node v is reachable by u if there exists a path connecting u and v.

C. 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. When the matrix operation is considered, the vectors a, b, c are considered as row vectors. The upper case notations A, B, C are then used for matrices of size $m \times n$ for some m. The capital I_k is reserved for the $k \times k$ identity matrix. We use a^{T} and A^{T} to denote the transposes of vector a and matrix A. 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 $\mathsf{GF}(q)$. We use $\langle a, b \rangle$ to denote the linear space spanned by a and b.

IV. A New Class of Max-Flow Algorithms

Discussion in this section focuses on a single unicast session, and a running example is provided to illustrate the algorithms. Results for a single multicast session are relegated to Section VI.

A. The Basic Structure

We consider the following time-invariant network coding scheme. Source s sends continuously packets $\{X_{i,t}\}_{i=1,\cdots,|\mathrm{Out}(s)|}$ along edges in $\mathrm{Out}(s)$ to its neighbors $\{w_i\}_{i=1,\cdots,|\mathrm{Out}(s)|}$ in the t-th second. A practical network coding scheme can then be devised as follows for directed acyclic networks with delay.

In the practical network coding scheme by [3], the incoming packets are buffered and packet mixing is allowed among packets of the same time stamp (also known as "generations"). We use $n \triangleq |\mathrm{Out}(s)|$ as the *generation size*. The coding vector for each edge, denoted by $m(u, v)$, is then an n-dimensional vector in $\mathsf{GF}(q)$. The i-th component corresponds to the coefficient with respect to $X_{i,t}$. For example, $m(s, w_i)$ is a delta vector δ_i with all but the i-th component being zero and the i-th component being one. Before computing the messages $m(\cdot, \cdot)$ along the networks, we perform the following initialization.

```
§ INITIALIZATION
1: Each edge $(s, w_i) \in \mathrm{Out}(s)$ sets $m(s, w_i)$ to be the $\delta_i$ vector.
2: Each node $v \in V \backslash \{s, d\}$ chooses arbitrarily an $|\mathrm{Out}(v)| \times |\mathrm{In}(v)|$ transfer matrix (also known as a local kernel) $\Gamma(v) = [\gamma_{w,u}(v)]$ where each entry $\gamma_{w,u}(v) \in \mathsf{GF}(q)$ for all $(v, w) \in \mathrm{Out}(v)$ and $(u, v) \in \mathrm{In}(v)$.
```

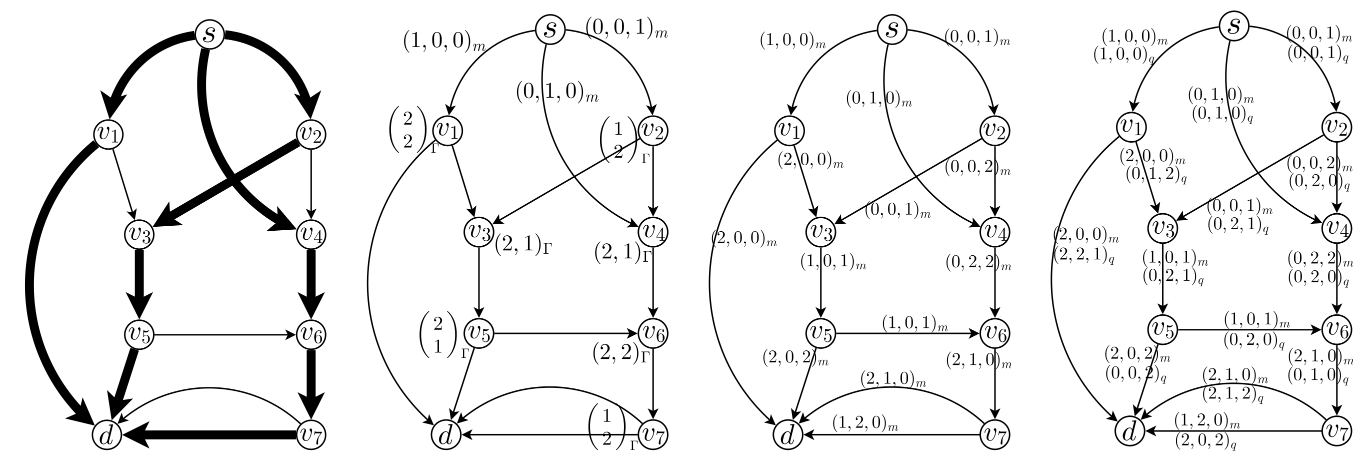


Fig. 2. An illustrative example of the BASIC CF ALGORITHM: (a) The underlying network, for which the max flow is illustrated by thick arrows, (b) After the INITIALIZATION step, (c) After the COMPUTE FORWARD MESSAGES step, and (d) After the COMPUTE CODED FEEDBACK step.

Consider a running example of a DAG as illustrated in Fig. 2(a). Source s would like to transmit (at the optimal rate) to destination d using a network with 7 other intermediate nodes. One max flow between s and d is illustrated by thick arrows, and the corresponding max-flow value is 3.

Consider a small finite field $\mathsf{GF}(3)$. The INITIALIZATION subroutine is illustrated in Fig. 2(b). Source s sends the coding vectors $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$ along edges (s, v_1), (s, v_4), and (s, v_2). We add subscripts m to emphasize that each row vector corresponds to a message vector $m(\cdot, \cdot)$. Each intermediate node also chooses a transfer matrix $\Gamma(v)$ of size $|\text{Out}(v)| \times |\text{In}(v)|$. Again we add the subscript Γ to emphasize it is a transfer matrix rather than a vector. For example, $\Gamma(v_2) = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$.

The same transfer matrix $\Gamma(v)$ will be used for packets of different generations, and the entire scheme is thus termed time-invariant network coding. The packets along $\text{In}(v)$ are mixed and sent over $\text{Out}(v)$ according to the transfer matrix $\Gamma(v)$. For any $E' \subseteq E$, since each message $m(\cdot, \cdot)$ is an n-dimensional row vector, we define the following bracket notation $[m(u, v) : (u, v) \in E']$ as an $|E'| \times n$ matrix vertically concatenating $|E'|$ row vectors. Based on this notation, we construct the following COMPUTE FORWARD MESSAGES routine that computes $m(\cdot, \cdot)$ based on the local kernels $\Gamma(\cdot)$.

---

§ COMPUTE FORWARD MESSAGES

Every second, each node $v \in V \backslash \{s, d\}$ performs the following tasks until all forward messages $m(\cdot, \cdot)$ in the network are stabilized and do not change anymore.

```
1: $[m(v, w) : (v, w) \in \text{Out}(v)] \leftarrow \Gamma(v) \cdot [m(u, v) : (u, v) \in \text{In}(v)]$.
```

The above routine computes the outgoing messages $m(v, w)$ from its incoming messages $m(u, v)$ based on the $\Gamma(v)$ chosen in INITIALIZATION.

Continue our running example in Fig. 2 (see especially Figs. 2(b) and 2(c)). Since $\Gamma(v_2) = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ and $\Gamma(v_3) = (2, 1)$, we have

$$\begin{aligned}
\begin{bmatrix} m(v_2, v_3) \\ m(v_2, v_4) \end{bmatrix} &= \Gamma(v_2) m(s, v_2) \\
&= \begin{pmatrix} 1 \\ 2 \end{pmatrix} (0, 0, 1) = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 2 \end{pmatrix} \\
m(v_3, v_5) &= \Gamma(v_3) \begin{bmatrix} m(v_1, v_3) \\ m(v_2, v_3) \end{bmatrix} \\
&= (2, 1) \begin{pmatrix} 2 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} = (1, 0, 1).
\end{aligned}$$

With the arbitrarily chosen $\Gamma(\cdot)$ in Fig. 2(b), the forward messages after the above routine are described in Fig. 2(c).

It is worth noting that the routine COMPUTE FORWARD MESSAGES models is simply an algorithmic way of describing packet mixing and broadcasting in a practical network coding scheme [3], which may take place either in the physical layer for wireless networks, or in the network layer for wireline networks, or even in the application layer for overlay networks.

The first novel component of the Coded Feedback (CF) max-flow algorithm is the introduction of *the coded feedback message* $q(u, v)$, which is an n-dimensional vector sent in the opposite direction of $m(u, v)$. Namely, for any $(u, v) \in \text{In}(v)$, its coded feedback $q(u, v)$ is a linear combination of $\{q(v, w) : (v, w) \in \text{Out}(v)\}$. We compute the feedback messages $q(\cdot, \cdot)$ by the following subroutine.

---

§ COMPUTE CODED FEEDBACK

In the beginning of this routine,

```
1: Use $\text{Rank}(d)$ to denote the rank of the matrix $[m(u, d) : (u, d) \in \text{In}(d)]$.
2: Destination $d$ arbitrarily constructs $(n - \text{Rank}(d))$ vectors $\{m^*_a : a \in \{\text{Rank}(d) + 1, \cdots, n\}\}$ such that jointly $\{m(e) : e \in \text{In}(d)\}$ and $\{m^*_a\}$ span the entire $n$-dimensional vector space $\Omega$.
```

```
3: Destination $d$ arbitrarily constructs two sets of vectors $\{q(e) : e \in \text{In}(d)\}$ and $\{q^*_a : a \in \{1, \cdots, n - \text{Rank}(d)\}\}$ such that the following matrices
```

$$\mathbf{q} \triangleq \begin{pmatrix} [q(e) : e \in \text{In}(d)] \\ [q^*_a : a \in \{\text{Rank}(d)+1, \cdots, n\}] \end{pmatrix}$$

$$\mathbf{m} \triangleq \begin{pmatrix} [m(e) : e \in \text{In}(d)] \\ [m^*_a : a \in \{\text{Rank}(d)+1, \cdots, n\}] \end{pmatrix}$$

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

After the above initialization, every second, each node $v \in V\backslash\{s, d\}$ performs the following task until all coded feedback $q(\cdot, \cdot)$ in the network are stabilized and do not change anymore.

```
1: $[q(u, v) : (u, v) \in \text{In}(v)] \leftarrow \Gamma(v)^{\mathrm{T}} \cdot [q(v, w) : (v, w) \in \text{Out}(v)]$.
```

In our running example, since $\text{Rank}(d) = 3 = n$, we do not need to add any additional coding vectors m^*_a as described in Line 2. In Line 3, we need to solve the following equation based on the forward coding vectors $m(\cdot, \cdot)$ in Fig. 2(c):

$$\begin{bmatrix} q(v_1, d) \\ q(v_5, d) \\ q_1(v_7, d) \\ q_2(v_7, d) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} m(v_1, d) \\ m(v_5, d) \\ m_1(v_7, d) \\ m_2(v_7, d) \end{bmatrix} = \begin{bmatrix} q(v_1, d) \\ q(v_5, d) \\ q_1(v_7, d) \\ q_2(v_7, d) \end{bmatrix}^{\mathrm{T}} \begin{pmatrix} 2 & 0 & 0 \\ 2 & 0 & 2 \\ 2 & 1 & 0 \\ 1 & 2 & 0 \end{pmatrix} = I_3. \tag{1}$$

The solution of (1) is not unique and we can choose arbitrarily one such solution. In this example, we choose

$$\begin{bmatrix} q(v_1, d) \\ q(v_5, d) \\ q_1(v_7, d) \\ q_2(v_7, d) \end{bmatrix}^{\mathrm{T}} = \begin{pmatrix} 2 & 0 & 2 & 2 \\ 2 & 0 & 1 & 0 \\ 1 & 2 & 2 & 2 \end{pmatrix},$$

where $q_1(v_7, d)$ and $q_2(v_7, d)$ are the upper and the lower of the parallel (v_7, d) edges, respectively.

In Fig. 2(d), we use the subscript q to denote the feedback coding vectors. The $q(\cdot, \cdot)$ feedback vectors are then computed and propagated from destination d back to source s based on the $\Gamma(\cdot)$ matrices (depicted in Fig. 2(b)). For example, since $\Gamma(v_7) = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ and $\Gamma(v_6) = (2, 2)$, we have

$$q(v_6, v_7) = \begin{pmatrix} 1 \\ 2 \end{pmatrix}^{\mathrm{T}} \begin{bmatrix} q_1(v_7, d) \\ q_2(v_7, d) \end{bmatrix} = (0, 1, 0)$$

$$\begin{bmatrix} q(v_4, v_6) \\ q(v_5, v_6) \end{bmatrix} = (2, 2)^{\mathrm{T}} q(v_6, v_7) = \begin{pmatrix} 0 & 2 & 0 \\ 0 & 2 & 0 \end{pmatrix}.$$

The $q(\cdot, \cdot)$ feedback vectors are depicted in Fig. 2(d). Note that both the forward and backward messages share the same transfer matrices $\Gamma(\cdot)$.

In order to decode the network coded (forward) traffic, destination d must be capable of solving linear equations. Therefore, no additional hardware is necessary to generate the $q(v, d)$ feedback vectors according to (1). The computation of the coded feedback $q(u, v)$ at the intermediate node is also as straightforward as the computation of $m(v, w)$, and thus no additional hardware is necessary for intermediate nodes either. Moreover, since the computation of $q(u, v)$ at the intermediate node is no different than the coding operation for the forward data traffic (except for using the same $\Gamma(v)$), we can use $q(u, v)$ as the coding vector of the reverse data traffic. If the $q(u, v)$ are indeed used to encode the reverse data traffic, there is zero computation and communication overhead at the intermediate node and only a small computation cost for the destination node d to compute and encode the reverse data traffic according to $q(v, d)$. If there is no reverse data traffic, we have to send explicit feedback packets containing $q(u, v)$. The communication overhead of sending $q(u, v)$ is identical to the coding overhead of sending $m(u, v)$, which is generally 3–6% of the normal traffic [3].

With this new component of coded feedback, the basic structure of a Coded Feedback (CF) max-flow algorithm can be described as follows.

```
§ BASIC CF MAX-FLOW ALGORITHM
1: INITIALIZATION
2: loop
3:   COMPUTE FORWARD MESSAGES
4:   COMPUTE CODED FEEDBACK
5:   Let $E_R(v)$ denote the output of IDENTIFY REDUNDANT EDGES, where $E_R(v) \subseteq \text{In}(v)$ for some node $v \in V\backslash\{s\}$.
6:   if $E_R(v) \neq \emptyset$ then
7:     Remove $E_R(v)$.
8:   else
9:     return the remaining graph $G$
10:  end if
11: end loop
```

The subroutine IDENTIFY REDUNDANT EDGES takes the stabilized forward and feedback messages as input, and outputs an edge set $E_R(v) \subseteq \text{In}(v)$ for some $v \in V\backslash\{s, d\}$. The goal is to design IDENTIFY REDUNDANT EDGES such that its output $E_R(v)$ *contains only edges that can be safely removed without affecting the rank of the space received by* d. Once the edges in $E_R(v)$ are removed, we repeat the computation of COMPUTE FORWARD MESSAGES and COMPUTE CODED FEEDBACK and the edge removal process. In the following subsection, we will show that one can design at least two different subroutines IDENTIFY REDUNDANT EDGES that compute the redundant edges in an efficient and distributed fashion. The same running example in Fig. 2 will be used to illustrate the steps of two different subroutines.

B. Two Specific Constructions

In this subsection, we provide two different constructions of IDENTIFY REDUNDANT EDGES. One is inspired by graph-theoretic observations and the other is inspired by pure algebraic properties between forward and backward messages.

B.1 — A Graph-Based Construction

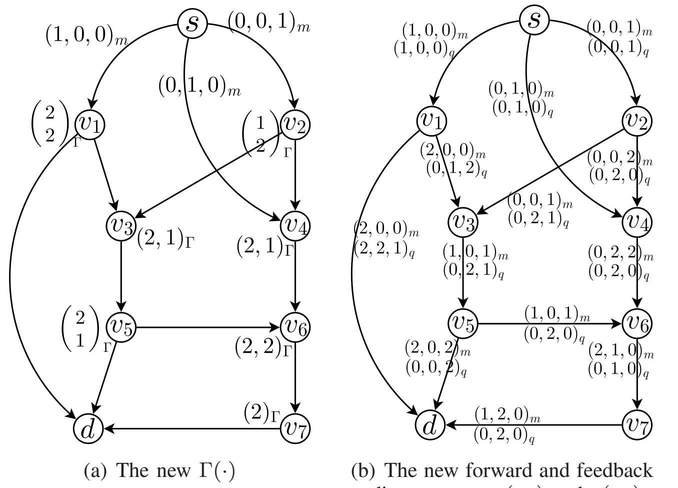
§ Graph-Based Identify Redundant Edges (GB-IRE)

This subroutine maintains a static counter cnt, which is initialize to cnt $\leftarrow |V|$ and decreases every time this subroutine is called.

```
1: while cnt $> 1$ do
2:     $v \leftarrow$ cnt
3:     if $v = |V|$ (i.e., $v$ is the destination node) then
4:         Find an $E_v \subseteq \mathrm{In}(d)$ such that the rank of $[m(v,d) : (v,d) \in E_v]$ is $\mathrm{Rank}(d)$.
5:         if $E_v \neq \mathrm{In}(v)$ then
6:             return $E_R(v) \leftarrow \mathrm{In}(v) \backslash E_v$.
7:         end if
8:     else
9:         Compute an $|\mathrm{Out}(v)| \times |\mathrm{In}(v)|$ matrix $\Phi = [q(v,w) : (v,w) \in \mathrm{Out}(v)][m(u,v) : (u,v) \in \mathrm{In}(v)]^{\mathrm{T}}$.
10:        Choose an $E_v \subseteq \mathrm{In}(v)$ such that $|E_v| = |\mathrm{Out}(v)|$ and the corresponding $|E_v|$ columns of $\Phi$ form a full rank square matrix.
11:        if $E_v \neq \mathrm{In}(v)$ then
12:            Consider the $|\mathrm{Out}(v)| \times |\mathrm{Out}(v)|$ square sub-matrix of $\Gamma(v)$ that corresponds to the transfer matrix from $[m(u,v) : (u,v) \in E_v]$ to $[m(v,w) : (v,w) \in \mathrm{Out}(v)]$.
13:            if the sub-matrix is not of full rank then
14:                Choose arbitrarily a new transfer matrix $\Gamma'(v)$ such that the new resulting $|\mathrm{Out}(v)| \times |\mathrm{Out}(v)|$ sub-matrix has full rank.
15:                Replace the old transfer matrix $\Gamma(v)$ with the new $\Gamma'(v)$.
16:            end if
17:            return $E_R(v) \leftarrow \mathrm{In}(v) \backslash E_v$.
18:        end if
19:    end if
20:    cnt $\leftarrow$ cnt $- 1$.
21: end while
22: return $E_R(v) \leftarrow \emptyset$.
```

In GB-IRE, we search for $E_R(v)$ from the most downstream node back to the most upstream node. Each node v computes $\Phi = [q(v,w) : (v,w) \in \mathrm{Out}(v)][m(u,v) : (u,v) \in \mathrm{In}(v)]^{\mathrm{T}}$. The edge set E_v identifies the "useful incoming messages" by ensuring the corresponding columns have full rank, see Line 10. Then one can safely remove those incoming edges not in E_v. Lines 13 to 15 ensure that the information of those useful incoming messages can be properly passed to its outgoing edges via a full-rank $\Gamma(v)$. If the existing $\Gamma(v)$ is not of full rank, then it is replaced by a new full-rank transfer matrix $\Gamma'(v)$.

Continue our running example. When the GB-IRE processes destination d ($v = |V|$), we select arbitrarily three linearly independent incoming messages and remove the rest. Suppose we select edges (v_1, d), (v_5, d), and the lower of (v_7, d) edges and remove the upper of (v_7, d) edges. After the edge removal, $\Gamma(v_7)$ changes from $(1,2)^{\mathrm{T}}$ to a scalar 2 (Fig. 3(a)). Based on Fig. 3(a), we will recompute the forward and feedback coding vectors $m(\cdot,\cdot)$ and $q(\cdot,\cdot)$. The new stabilized vectors



(a) The new $\Gamma(\cdot)$ (b) The new forward and feedback coding vectors $m(\cdot,\cdot)$ and $q(\cdot,\cdot)$.

Fig. 3. An illustration of GB-IRE after processing the destination d.

are depicted in Fig. 3(b).

In the second iteration, the GB-IRE searches for useful incoming edges of v_7. Since for v_7 the product Φ

$$q(v_7, d) \cdot m(v_6, v_7) = (0,2,0) \cdot (2,1,0) = 2$$

is of full rank, GB-IRE results in $E_{v_7} = \{(v_6, v_7)\} = \mathrm{In}(v_7)$ and no redundant edge will be identified. GB-IRE moves to the next node and searches for useful incoming edges of v_6. For v_6 the product Φ becomes

$$\Phi = q(v_6, v_7) \begin{bmatrix} m(v_5, v_6) \\ m(v_4, v_6) \end{bmatrix}^{\mathrm{T}} = (0,1,0) \begin{pmatrix} 1 & 0 \\ 0 & 2 \\ 1 & 2 \end{pmatrix} = (0,2),$$

and there is a unique choice of useful edges: $E_{v_6} = \{(v_4, v_6)\}$. The redundant edge is thus $E_R(v_6) = \mathrm{In}(v_6) \backslash E_{v_6} = \{(v_5, v_6)\}$. After removing (v_5, v_6), we need to rerun Compute Forward Messages and Compute Coded Feedback. The new network, the stabilized forward and feedback messages $m(\cdot,\cdot)$ and $q(\cdot,\cdot)$ are illustrated in Fig. 4. Note that the forward messages $m(\cdot,\cdot)$ change from Fig. 3(b) to Fig. 4 due to the removal of (v_5, v_6). The feedback messages $q(\cdot,\cdot)$ thus change accordingly. By repeatedly searching for useful edges in $\mathrm{In}(v_5), \mathrm{In}(v_4), \cdots, \mathrm{In}(v_1)$, the CF max-flow algorithm based on GB-IRE will return the maximum flow as described in Fig. 2(a).

B.2 — An Algebra-Based Construction

§ Algebra-Based Identify Redundant Edges (AB-IRE)

```
1: if there exists a $v \in V$ and a non-empty edge subset $\Xi \subseteq \mathrm{In}(v)$ satisfying the following properties: (i) let $\Pi$ denote the $|\Xi| \times |\Xi|$ square matrix $\Pi = [q(u,v) : (u,v) \in \Xi] \cdot [m(u,v) : (u,v) \in \Xi]^{\mathrm{T}}$, and (ii) $I_{|\Xi|} - \Pi$ is of full rank then
2:     $E_R(v)$ can be chosen arbitrarily as any of such $\Xi$ satisfying the above properties for some $v \in V$.
3: else
4:     $E_R(v) \leftarrow \emptyset$
```

From Fig. 3(b), choose $v = v_6$, then

$$\Phi = (0,1,0)\begin{pmatrix} 1 & 0 \\ 0 & 2 \\ 1 & 2 \end{pmatrix} = (0,2)$$

$E_{v_6} = \{(v_4, v_6)\}$
$E_R(v_6) = \{(v_5, v_6)\}$.

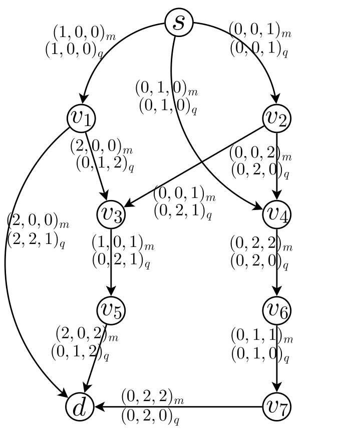


Fig. 4. GRAPH-BASED IDENTIFY REDUNDANT EDGES (GB-IRE) and the resulting graph after the removal of $E_R(v_6) = (v_5, v_6)$.

From Fig. 2(d), choose $v = v_3$ & $\Xi = \{(v_1, v_3)\}$, then

$$\Pi = (0,1,2)\begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} = 0$$

$1 - \Pi$ has full rank
$E_R(v_3) = \{(v_1, v_3)\}$.

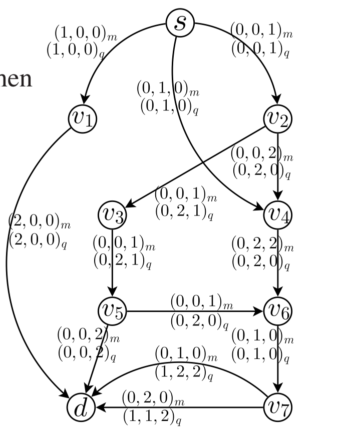


Fig. 5. ALGEBRA-BASED IDENTIFY REDUNDANT EDGES (AB-IRE) and the resulting graph after the removal of $E_R(v_3) = (v_1, v_3)$.

```
5: end if
```

In the following, we illustrate how AB-IRE finds the max flow in the network.

In AB-IRE, nodes can be searched in any order. Suppose the first node to search is v_3. There are three possible collections of redundant edges: $\Xi_1 = \{(v_1, v_3)\}$, $\Xi_2 = \{(v_2, v_3)\}$, and $\Xi_3 = \{(v_1, v_3), (v_2, v_3)\}$. Based on the stabilized coding vectors in Fig. 2(d), the $I - \Pi$ matrices are

$$\begin{aligned}
1 - \Pi_1 &= 1 - q(v_1, v_3)m(v_1, v_3)^{\mathrm{T}} \\
&= 1 - (0,1,2)(2,0,0)^{\mathrm{T}} = 1 \\
1 - \Pi_2 &= 1 - q(v_2, v_3)m(v_2, v_3)^{\mathrm{T}} \\
&= 1 - (0,2,1)(0,0,1)^{\mathrm{T}} = 0 \\
I_2 - \Pi_3 &= I_2 - \left[\begin{array}{c} q(v_1, v_3) \\ q(v_2, v_3) \end{array}\right]\left[\begin{array}{c} m(v_1, v_3) \\ m(v_2, v_3) \end{array}\right]^{\mathrm{T}} \\
&= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 2 \\ 0 & 2 & 1 \end{pmatrix}\begin{pmatrix} 2 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}.
\end{aligned}$$

The only full rank choice is $1 - \Pi_1$. Therefore, $\Xi_1 = \{(v_1, v_3)\}$ is a redundant edge set according to AB-IRE. By removing $E_R(v_3) = \{(v_1, v_3)\}$, we have a new network as in Fig. 5. Note that the forward messages $m(\cdot,\cdot)$ and the coded feedback $q(\cdot,\cdot)$ change from Fig. 2(d) to Fig. 5 due to the removal of (v_1, v_3). By repeatedly searching for redundant edges $E_R(v)$ in any order, the CF max-flow algorithm based on AB-IRE will return the max flow as described in Fig. 2(a).

C. *Comparison between* GB-IRE *and* AB-IRE

Both GB-IRE and AB-IRE are based on algebraic operations but they have the following major differences.

- When searching for the possible $E_R(v)$, GB-IRE starts from the most downstream nodes and moves upward, while the AB-IRE can start from arbitrary node v.
- GB-IRE focuses on the Φ matrix, which is the product of $q(\cdot,\cdot)$ on $\mathrm{Out}(v)$ and $m(\cdot,\cdot)$ on $\mathrm{In}(v)$. AB-IRE focuses on the Π matrix, which is the product of $q(\cdot,\cdot)$ and $m(\cdot,\cdot)$ on the same $\mathrm{In}(v)$.
- GB-IRE identifies useful edges E_v by searching for the independent columns, and then removes the complement $\mathrm{In}(v)\backslash E_v$. AB-IRE identifies directly redundant edges by computing the rank of $I_{|\Xi|} - \Pi$.
- GB-IRE will correct the inefficient choice of $\Gamma(v)$ by replacing it with a full rank matrix (Line 15). On the other hand, AB-IRE will not help update $\Gamma(v)$ even if the original choice of $\Gamma(v)$ is inefficient.[4] In almost all reasonable network coding schemes, each node v choose a transfer matrix $\Gamma(v)$ with the maximum possible rank during INITIALIZATION. In those cases, GB-IRE does not make any change to $\Gamma(v)$ either.

In summary, AB-IRE is more suitable for distributed implementation as it does not require any activation order of v. On the other hand, the complexity of finding $E_R(v)$ in AB-IRE is higher as one has to check the rank of $I_{|\Xi|} - \Pi$ for each Ξ separately. Detailed discussion on low-complexity implementations is relegated to Section VI-A.

V. THE CORRECTNESS OF THE PROPOSED ALGORITHMS

In this section, we state the correctness and complexity results of the CF max-flow algorithms with GB-IRE and AB-IRE, respectively. To streamline the discussion, the detailed proofs are provided in the appendices.

A. *When* GB-IRE *Is Used*

In this subsection, we assume GB-IRE is used in the CF max-flow algorithm. We state the main propositions first and then give several corollaries based on random network coding and the concept of generic Linear Code Multicast (LCM). The proofs are provided in Appendix A.

Proposition 1: For any finite field size $\mathsf{GF}(q)$, the following properties hold for the CF algorithm using the GB-IRE subroutine.

[4]See the discussion of generic LCM versus non-generic LCM in [21] for the efficiency of transfer matrices.

1) One can always find a valid E_v as described in Line 10 of GB-IRE;
2) The BASIC CF ALGORITHM stops in $2l|V|$ seconds[5] where l is the length of the longest path in the network. Before the algorithm stops, at most $|V||E|$ feedback messages $q(u,v)$ are generated and sent along the network;
3) Throughout the main loop in Line 2 of the BASIC CF ALGORITHM, the dimension of $\langle m(u,d) : (u,d) \in \text{In}(d)\rangle$ remains unchanged;
4) The BASIC CF ALGORITHM returns a *flow*.

With high probability, random linear network coding (for which the $\Gamma(v)$ transfer matrices are selected randomly in INITIALIZATION) will have the dimension of $\langle m(u,d) : (u,d) \in \text{In}(d)\rangle$ equal the max-flow value $\mathsf{MFV}(s,d)$ [14]. We therefore have the following corollary as a direct result of Properties 3 and 4 in Proposition 1.

Corollary 1: When random linear network coding is used, the probability that the CF algorithm with GB-IRE returns a max (s,d)-flow approaches one when a sufficiently large $\mathsf{GF}(q)$ field is used.

With random linear network coding, Property 3 ensures that the information exchange remains uninterrupted during the entire session and is, with high probability, at the optimal rate.

It is worth noting that even when the transfer matrix $\Gamma(v)$ is fixed, the choices of E_v and $\Gamma'(v)$ in COMPUTE CODED FEEDBACK and in Lines 10 and 15 of GB-IRE are not unique. Choosing different E_v and $\Gamma'(v)$ will return different flows as the final output. The flexibility of the CF algorithm with GB-IRE is then illustrated by the following proposition.

Proposition 2: Suppose the $\mathsf{GF}(q)$ size is sufficiently large and the initial random network coding is a generic linear-code multicast (LCM) as defined in [21]. For any max (s,d)-flow, we can make a sequence of choices of $\{m_a^*\}$, $\{q(v,d) : (v,d) \in \text{In}(d)\}$, E_v, and $\Gamma'(v)$ such that the output graph of the CF algorithm is the given max (s,d)-flow.

B. *When* AB-IRE *Is Used*

In this subsection, we assume the AB-IRE is used in the CF algorithm. The proofs of the properties are provided in Appendix B.

[5]In this result, we assume the computation time within each node is negligible to the propagation delay of sending packets along a given edge. This convergence time can also be strengthened even if the computation time is not negligible. Assume the propagation delay is 1 second and it takes δt second to finish a $\mathsf{GF}(q)$ addition or multiplication. The total time for finishing the CF algorithm are upper bounded by

$$|V|(2l + 2|V| \cdot 2\max_{v\in V}|\text{In}(v)||\text{Out}(v)|n\delta t + 2n^3\delta t + 2\max_{v\in V}|\text{In}(v)||\text{Out}(v)|n\delta t + \max_{v\in V}|\text{In}(v)||\text{Out}(v)|^2\delta t)$$

where the common multiplier $|V|$ is the number of rounds and the terms inside the parentheses are the delays for each round. The first term is the propagation delay, the second term is the amount of time for intermediate nodes to compute the $m(\cdot,\cdot)$ and $q(\cdot,\cdot)$ coding vectors, the third term is the amount of time to generate the initial feedback vectors $q(\cdot,\cdot)$, the fourth term is the time to compute Φ, and the last term is the time to select the E_v edge set (or the $E_R(v)$ edge set). For network implementation, the efficiency of the CF algorithm thus depends on the ratio of the processing delay and the propagation delay.

We first note that AB-IRE is a generic subroutine as the choices of which v to examine and which Ξ edge subset of $\text{In}(v)$ to remove are not unique. The following properties hold for the CF algorithm with any generic AB-IRE.

Proposition 3: For any finite field size $\mathsf{GF}(q)$, the following properties hold for the CF algorithm using the AB-IRE subroutine.

1) The BASIC CF ALGORITHM stops in $2l|E|$ seconds where l is the length of the longest path in the network. Before the algorithm stops, at most $|E|^2$ feedback messages $q(u,v)$ are generated and sent along the network;
2) Throughout the main loop in Line 2 of the BASIC CF ALGORITHM, the dimension of $\langle m(u,d) : (u,d) \in \text{In}(d)\rangle$ remains unchanged;
3) Suppose the space $\langle m(u,d) : (u,d) \in \text{In}(d)\rangle$ received by d is the entire vector space Ω and consider the final output graph of the CF algorithm.[6] If the coefficients $\Gamma(\cdot)$ cannot be changed, then removing any non-empty subset of $\text{In}(v)$ of any given v will decrease the dimension of the space received by d.

Property 1 discusses the complexity and Property 2 ensures no interruption to the forward traffic. Property 3 focuses on the optimality of the final output. We have to emphasize that Property 3 of Proposition 3 does not exclude the case in which removing simultaneously two non-empty subsets of $\text{In}(v_1)$ and $\text{In}(v_2)$ for a given pair of distinct nodes $v_1 \neq v_2$ might still keep the dimension of the received space unchanged. However, if each node v has to make its own decision locally whether to remove a subset of $\text{In}(v)$, then no further edge reduction can be made. Accordingly, we say the BASIC CF ALGORITHM with AB-IRE finds a network coding solution with locally minimal network usage.

Another major distinction between GB-IRE and AB-IRE can be seen when comparing Property 4 of Proposition 1 and Property 3 of Proposition 3. The former focuses on the graph-theoretic property that the output being a flow while the latter concentrates on a pure algebraic statement regarding locally minimal network usage. To further bridge the algebra-based construction with the graph-theoretic concepts, we again rely on the notions of the generic LCM in [21] and random linear coding in [14]:

Proposition 4: Suppose the $\mathsf{GF}(q)$ size is sufficiently large and the $\Gamma(v)$ chosen in INITIALIZATION corresponds to a generic LCM. Then there exists a choice of $\{m_a^* : a \in \{\text{Rank}(d)+1,\cdots,n\}\}$ such that the output of the CF algorithm with AB-IRE is a max (s,d)-flow.

[6]The additional constraint $\langle m(u,d) : (u,d) \in \text{In}(d)\rangle = \Omega$ is unique for Property 3 while Property 2 always holds regardless of the dimension of the received space $\langle m(u,d) : (u,d) \in \text{In}(d)\rangle$. Since the goal of any practical network coding solution is to recover all n symbols in a given generation, this additional condition imposes little restriction for practical applications. In practice, when the achievable max-flow-value rate is less than n, source s sends linearly coded (dependent) packets to facilitate decoding at d instead of sending linearly independent packets. The parity-check constraints of the coded packets are common knowledge shared by both s and d so that d can decode the original information. Therefore, sending coded packets can be modelled as sending independent packets plus allowing auxiliary "virtual pipes" to carry the parity-check information directly from s to d. After this conversion, the "augmented network coding solution" has $\langle m(u,d) : (u,d) \in \text{In}(d)\rangle = \Omega$. The additional constraint is satisfied.

Corollary 2: When random linear network coding is used and when $\{m_a^* : a \in \{\mathrm{Rank}(d)+1, \cdots, n\}\}$ is also chosen randomly, the probability that the CF algorithm with AB-IRE returns a max (s,d)-flow approaches one when a sufficiently large $\mathsf{GF}(q)$ field is used.

Proof of Corollary 2: This is a direct consequence of the proof of Proposition 4 and that for sufficiently large size of $\mathsf{GF}(q)$, with high probability, random linear coding results in a generic LCM. ■

Using the generic LCM during INITIALIZATION also improves the complexity and flexibility of the CF algorithm with AB-IRE.

Proposition 5: Suppose the $\mathsf{GF}(q)$ size is sufficiently large and the $\Gamma(v)$ chosen in INITIALIZATION corresponds to a generic LCM. Then there exists a choice of $\{m_a^* : a \in \{\mathrm{Rank}(d)+1, \cdots, n\}\}$ such that

1) The CF algorithm with AB-IRE stops in $2l|V|$ seconds if for any v, we choose the maximum-sized $\Xi \subseteq \mathrm{In}(v)$ in Line 2 of AB-IRE and output the maximum Ξ as $E_R(v)$;
2) For any given max (s,d)-flow, we can make a sequence of choices of v and Ξ in Line 2 of AB-IRE such that the output graph of the CF algorithm is the given max (s,d)-flow.

Proposition 5 ensures that when a generic LCM is used, the complexity of the CF algorithm using AB-IRE is no different than that of using GB-IRE and the CF algorithm is able to converge to any given max flow. As a result, when random linear network coding is used on a sufficiently large $\mathsf{GF}(q)$, with high probability, the CF algorithm using AB-IRE has almost identical computational complexity and flexibility as that of the GB-IRE while still enjoys the freedom of searching intermediate network nodes in any arbitrary order

VI. IMPLEMENTATIONS AND GENERALIZATIONS

A. Low-Complexity Distributed Implementation

The BASIC CF ALGORITHM can be made distributed in a straightforward manner. The subroutines COMPUTE FORWARD MESSAGES and COMPUTE CODED FEEDBACK can be easily implemented in a network as they are direct analogy of the network coding traffic. Deciding which node v and the associated $E_R(v) \subseteq \mathrm{In}(v)$ to remove by IDENTIFY REDUNDANT EDGES is the only subroutine for which some form of coordination is necessary. For both GB-IRE and AB-IRE, this coordination can be achieved by a token-based approach. Each intermediate node calculates its own Φ (or Π) matrices and checks whether it satisfies the criteria in GB-IRE (or in AB-IRE). Consider those v that have non-empty $E_R(v)$ subject to edge deletion. Each v requests a token from the destination d. Using a single token ensures that one and only one v is allowed to remove $E_R(v)$, or equivalently to stop its incoming traffic. Passing the token along the data and acknowledgement packets also enforces sufficiently long waiting time for $m(\cdot,\cdot)$ and $q(\cdot,\cdot)$ to stabilize.

An even more decoupled scheme is that each v that has a non-empty $E_R(v)$ just waits a random number of seconds before stopping the traffic on $E_R(v)$. A carefully designed random waiting time (with carefully designed reset mechanisms) could ensure that with high probability $(1-\epsilon)$, no two nodes trim the graph simultaneously and the waiting time for $m(\cdot,\cdot)$ and $q(\cdot,\cdot)$ to stabilize is sufficient. Since Property 2 of Proposition 3 guarantees that only when we trim the network before $m(\cdot,\cdot)$ and $q(\cdot,\cdot)$ are stabilized will the rank decrease, such a random waiting time scheme will decrease the achievable rate with a small probability ϵ, which would be a welcome compromise for a perfectly distributed token-free algorithm.[7]

B. Searching Max Flows with Constraints on Delay or Coding Complexity

The CF algorithm focuses on maintaining the same dimension of the received space while trimming the network usage. Therefore, even in the (relatively infrequent) cases in which random linear network coding achieves only a near-optimal $r < \mathsf{MFV}$ rate, one can still apply the CF algorithm to prune the unnecessary edges while sustaining the same 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,d) paths that use fewer than h hops. Then the user can use a controlled broadcast plus network coding scheme that explores only paths of length $< h$. The optimal rate MFV may not be attainable under this partial network exploration. The CF algorithm allows the user to still achieve the best possible rate r and prune the redundant traffic. If h is chosen as the minimal distance from s to d, the initial controlled broadcast is no different than the routing-based shortest-path discovery. Varying the h value thus serves as an additional degree of freedom in network design: How widely the (s,d) communication is going to spread on the network. The CF algorithm provides a way to harness the h-controlled broadcast by pruning the unnecessary traffic after the initial controlled broadcast.

Similarly, the complexity constraint may also limit how large the $\mathsf{GF}(q)$ size that can be employed in the network. In some cases, the allowed $\mathsf{GF}(q)$ size is too small and one cannot achieve the optimal MFV multicast rate for multiple destinations [8]. In the following subsection, we illustrate how to adapt the unicast CF max-flow algorithm for the multicast session. Since the CF algorithm applies to $\mathsf{GF}(q)$ of arbitrary size, one can still maintain the best achievable rate under a small $\mathsf{GF}(q)$ while trimming the network usage in the optimal way.

C. Solving Multiple Max Flows Simultaneously

The CF Algorithm with AB-IRE can be generalized for the multicast session problem so that the output is a multicast network coding solution with locally minimal network usage.

Consider a multicast session problem from source s to destinations $\mathbf{d} = \{d_i\}$. A generalized CF algorithm is described

[7]It is shown in [18] that for a general network most edges are not in an edge cut. Therefore, even if a node performs graph-trimming before the messages are stabilized from an earlier graph-trimming, it is still likely that the rank of $\langle m(u,d) : (u,d) \in \mathrm{In}(d)\rangle$ may not decrease.

as follows, in which each feedback message is $\mathbf{q}(u,v)$ that contains $|\mathbf{d}|$ vectors $\{q_1(u,v), \cdots q_{|\mathbf{d}|}(u,v)\}$.

```
§ GENERALIZED CF ALGORITHM FOR THE MULTICAST PROBLEM
 1: INITIALIZATION
 2: loop
 3:     COMPUTE FORWARD MESSAGES
 4:     Run COMPUTE CODED FEEDBACK for all destinations $d_i \in \mathbf{d}$ and use $\mathbf{q}(u,v)$ to denote the $|\mathbf{d}|$ vectors $\{q_1(u,v), \cdots q_{|\mathbf{d}|}(u,v)\}$
 5:     Let $E_R(v)$ denote the output of AB-MULTI-RE, where $E_R(v) \subseteq \mathrm{In}(v)$ for some node $v \in V \backslash \{s\}$.
 6:     if $E_R(v) \neq \emptyset$ then
 7:         Remove $E_R(v)$.
 8:     else
 9:         return the remaining graph $G$
10:     end if
11: end loop
```

In the above algorithm, there is only one new subroutine:

```
§ ALGEBRA-BASED IDENTIFY MULTI-REDUNDANT EDGES (AB-MULTI-RE)
 1: if there exists a $v \in V$ and a non-empty edge subset $\Xi \subseteq \mathrm{In}(v)$ satisfying the following properties for all $d_i \in \mathbf{d}$: (i) let $\Pi_i$ denote the $|\Xi| \times |\Xi|$ square matrix $\Pi_i = [q_i(u,v) : (u,v) \in \Xi] \cdot [m(u,v) : (u,v) \in \Xi]^{\mathrm{T}}$, and (ii) $I_{|\Xi|} - \Pi_i$ is of full rank then
 2:     $E_R(v)$ can be chosen arbitrarily as any of such $\Xi$ satisfying the above properties for some $v \in V$.
 3: else
 4:     $E_R(v) \leftarrow \emptyset$
 5: end if
```

The above algorithm and the corresponding subroutines are straightforward generalizations for BASIC CF ALGORITHM, COMPUTE CODED FEEDBACK, and AB-IRE. Many existing properties for the basic CF algorithm with AB-IRE hold for the generalized CF algorithm as well.

Proposition 6: For any finite field size $\mathsf{GF}(q)$, the following properties hold for the generalized CF algorithm using the AB-MULTI-RE subroutine.

1) The GENERALIZED CF ALGORITHM stops in $2l|E|$ seconds where l is the length of the longest path in the network. Before the algorithm stops, at most $|E|^2$ feedback messages $\mathbf{q}(u,v)$ are generated and sent along the network;
2) Throughout the main loop in Line 2 of the GENERALIZED CF ALGORITHM, the dimension of $\langle m(u,d_i) : (u,d_i) \in \mathrm{In}(d_i)\rangle$ remains unchanged for all destinations $d_i \in \mathbf{d}$;
3) Suppose the received spaces $\langle m(u,d_i) : (u,d_i) \in \mathrm{In}(d_i)\rangle$ for all $d_i \in \mathbf{d}$ are the entire vector space Ω. Consider the final output graph of the generalized CF algorithm. If the coefficients $\Gamma(\cdot)$ cannot be changed, then removing any non-empty subset of $\mathrm{In}(v)$ of any given v will decrease the dimension of the received space for at least one $d_i \in \mathbf{d}$.

The first property ensures that with the assumption[8] that each acknowledgement packet is capable of carrying multiple coded feedback vectors $\{q_i(u,v) : \forall d_i \in \mathbf{d}\}$, one can identify a locally minimal union of multiple max flows within the same time as that of identifying a single max flow.

Property 2 guarantees the correctness of the algorithm with no interruption to the forward traffic. Property 3 focuses on the local optimality of the resulting graph. The proof of Proposition 6 is provided in Appendix C.

When the underlying network coding is a generic LCM with sufficiently large $\mathsf{GF}(q)$, all the properties for the basic CF algorithm regarding the complexity and the flexibility can be extended straightforward to the generalized CF algorithm. We state the corresponding properties for the generalized CF algorithm with their proofs omitted.

Proposition 7: Suppose the $\mathsf{GF}(q)$ size is sufficiently large and the $\Gamma(v)$ chosen in INITIALIZATION corresponds to a generic LCM. Then there exists a choice of $\{m^*_{i,a} : a \in \{\mathrm{Rank}(d_i)+1, \cdots, n\}\}$ for all destinations d_i such that

1) The output of the generalized CF algorithm is a union of max (s,d_i)-flows that is of locally minimal network usage. Namely, removing any edge will decrease the max-flow values for some destination $d_i \in \mathbf{d}$;
2) The generalized CF algorithm stops in $2l|V|$ seconds if for each $E_R(v)$, we choose the maximum-sized $\Xi \subseteq \mathrm{In}(v)$ in Line 2 of AB-MULTI-RE;
3) For any given union of max (s,d_i)-flows that is of locally minimal network usage, we can make a sequence of choices of v and Ξ in Line 2 of AB-MULTI-RE such that the output graph of the generalized CF algorithm is the given locally minimal union of max (s,d_i)-flows.

D. A Greedy Search for Max Flows With Minimal Cost

All the algorithms discussed previously are generic as they allow flexible designs for the performance and complexity tradeoff. For example, in AB-IRE and in AB-MULTI-RE, one has the freedom of designing an arbitrary search order of v for which the redundant edge set $E_R(v)$ will be returned. Even for the same given v, there might be more than two choices of Ξ. Searching for those Ξ containing more edges accelerates the convergence speed as one could remove more edges in a single round. However, searching for large Ξ requires more computational resources. One can thus strike a balance of the computation resources within a single node and the convergence speed of the entire network.

In the following, we demonstrate the flexibility of the proposed algorithm by using the generalized CF algorithm as a simple greedy algorithm searching for the union of max flows with small total cost. Based on a designated max-flow algorithm, the proposed scheme returns a locally optimal

[8]This assumption is not very restrictive. In practice, the number of bits for carrying the coding coefficients is much smaller than the packet size as each forward packet needs to carry both the coding coefficients and the coded payload. Since each ACK packet does not need to carry the coded payload, multiple coded feedback $q_i(u,v)$ can be carried within a single ACK packet.

solution rather than a globally minimal cost solution that is generally obtained by an LP solver. In Section VII, we compare the deficiency of using this fast but sub-optimal solution based on CF algorithms versus the optimal but more complicated LP solutions such as [27], [33], [34], [36]

Consider the single source multicast problem from s to $\mathbf{d}$. In addition to the setting discussed previously, each edge e charges a non-uniform price $c(e)$ for carrying one packet. Our goal is to find a subgraph G' such that the max-flow values between all (s, d_i) in G' are identical to those in the original graph G, while at the same time the total cost $\sum_{e\in G'} c(e)$ is kept small. The cost function $c(e)$ represents generally the power consumption for a given link. A reasonable choice is

$$c(e) = \frac{1}{\text{multiplicity of edge } e} \tag{2}$$

where the multiplicity of edge $e = (u, v)$ is the number of parallel edges connecting nodes u and v. To be more specific, in a given wireless network, each node can use certain power level to send packets along a given link. Depending on the noise level of the given link, different link data rates can be achieved for different edges (even when the same power level is used). Since links with high capacity are modelled by multiple parallel edges, the power (or time) consumption for each edge in a given link is thus (2). Minimizing the total power thus corresponds to minimizing $\sum_{e\in G'} c(e)$.

The generalized algorithm can be easily modified as a greedy search algorithm for the above problem. To that end, random linear network coding is used in INITIALIZATION, which ensures a generic LCM is generated with a high probability. Since the output $E_R(v)$ of AB-MULTI-RE can be chosen arbitrarily as any valid Ξ, the greedy search simply returns an edge set $E_R(v)$ that is the valid Ξ having highest cost per edge. By sequentially removing the redundant edge sets with the highest average cost, the generalized CF algorithm acts as a greedy algorithm and reaches a locally optimal solution.

VII. Numerical Experiments

In this section, numerical experiments are conducted for the CF algorithms with GB-IRE and with AB-IRE. We use the routing-based, distributed push-&-relabel (P&R) algorithm as benchmark for performance comparison. We also implement the generalized CF algorithm in Section VI-D, which finds locally minimal network coding solutions for any given multicast session with total cost $\sum_{e\in G'} c(e)$ defined in (2). Some inherent differences between the existing routing-based and the proposed network-coding-based algorithms are discussed in this section as well.

A. Numerical Experiment 1 — Unicast

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.[9] 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 Propositions 1, 3, and 5. In terms of the total number of feedback messages exchanged in the network, the complexity is $\mathcal{O}\left(|V||E|\right)$. We also note that 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 may be piggybacked to the reverse data traffic as discussed in Section IV-A.

We use the generic push-&-relabel (P&R) algorithm [11] 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 CF 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 CF algorithm.

For numeric comparisons, we consider a randomly generated, sparse, 30-node DAG, of which each node is connected to 5.2 links in average. The capacity of each link (or equivalently the multiplicity of each edge) is randomly chosen from 1 to 10. To describe explicitly the underlying network, we consider its 30×30 incidence matrix $A = \begin{bmatrix} A_{1-15} \\ A_{16-30} \end{bmatrix}$, where each entry $A_{i,j}$ of A is the number of parallel directed edges connecting nodes i and j. If we use "a" as the hexadecimal digit for 10 and use "." for zeros, the two 15×30 sub-matrices A_{1-15} and A_{16-30} can be expressed as follows.

```
........a8....................   .................6...3a.a.....
...a.....3....................   ...................a8a........
....44..121...................   ....................6..68.....
.....a....3...................   .....................6..395...
.....59...a.8.................   ....................65..48..9.
........3.....8...............   .....................1...3.9.a
........16.5...8..............   .......................6....1.
........4a....82..............   .......................6381...
.........8.6.1................   .........................1..8.
..........9..7................   ............................4.
...........5..2a3..6..........   ..............................
................6.2...........   .............................8
..............a..a.7..........   ..............................
...............5a.............   ..............................
................4....56.......,  ..............................
```

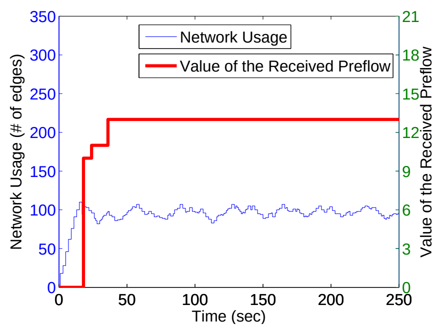
All the numerical experiments are conducted based on the above graph.

Fig. 6(b) contains the time evolution of the network usage and the dimension of the received space when the CF algorithm with GB-IRE is applied to the above network. The source $s = 1$ and the destination $d = 30$. The max flow value between s and d is $\mathsf{MFV}(1, 30) = 13$ and the largest rate of a single-path route is 7. Therefore network coding achieves $13/7 \approx 185.7\%$ throughput when compared to the best single-path routing. From the time-lapse perspective, broadcast plus network coding achieves the optimal $\mathsf{MFV}(1, 30)$ rate in 7 seconds while the CF max-flow algorithm with GB-IRE converges in 116 seconds. In the 109-second interval between the 7-th and the 116-th seconds, source s and destination d sustain the optimal-rate transmission even before the convergence of the algorithm.

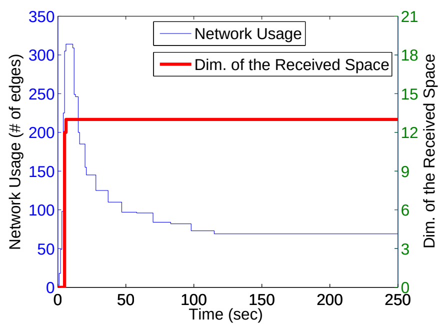
If we run the distributed P&R[10] on the same graph, it requires 18 seconds before a non-zero transmission rate can be

[9]See footnote 5 for discussion when the computation time is non-negligible.

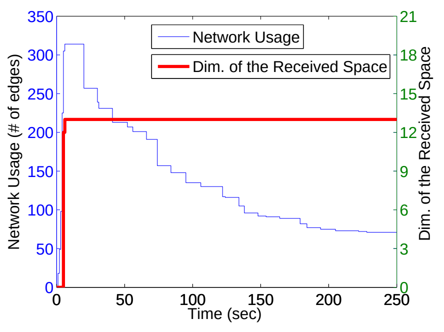
[10]Greedy non-interfering scheduling is assumed, i.e. two neighboring nodes cannot perform the push or the relabel operations simultaneously.



(a) The P&R Algorithm



(b) The CF Algorithm w. GB-IRE



(c) The CF Algorithm w. AB-IRE

Fig. 6. The time evolution graphs of different max (s,d)-flow algorithms including (a) the benchmark push-&-relabel algorithm, (b) the Coded Feedback (CF) algorithm with GB-IRE, and (c) the CF algorithm with AB-IRE. The incidence matrix of the underlying directed acyclic network is described in Section VII-A.

established. The optimal rate can be reached in 36 seconds,[11] but the max flow will not converge until the 487-th second. The push-and-pull approach of the P&R algorithm results in the oscillating nature of the network usage, as seen in Fig. 6(a). 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 might imposes significant control overhead for the network. For comparison, the traffic trimmed by the CF algorithm will never be added back to the network as can be seen from the monotonic traffic reduction after the initial broadcast stage. Both the P&R and the CF algorithm with GB-IRE output max flows that use roughly 1/4 of the total 314 edges.

Fig. 6(c) contains similar time-evolution data for the CF algorithm with AB-IRE. A randomly-chosen search order of intermediate nodes v is employed by AB-IRE when searching for $E_R(v)$. Again, the CF algorithm with AB-IRE achieves the optimal rate transmission in 7 seconds and sustains the rate throughout iterations. Traffic is trimmed monotonically from the initial broadcast. The CF algorithm with AB-IRE converges after 228 seconds. The slower convergence speed of AB-IRE is caused by the randomly chosen search order. Since the redundant edge set $E_R(v)$ is not searched from the nodes closest to d but in an arbitrary order, it takes a longer time for the forward and feedback messages to stabilize during each round. So each iteration takes longer time even when the total number of iterations are the same for both the GB-IRE- and the AB-IRE-based algorithms.

As can be seen in the evolution of network usage in Figs. 6(b) and 6(c), there is an initial spike of network usage due to the aggressive packet broadcast in the beginning stage. One can use controlled broadcast to mitigate this initial network usage spike. The tradeoff is that the CF algorithm may not be able to identify the optimal max flow due to the more self-restrained initial broadcast. As discussed previously, the CF algorithm can still identify the best rate under this suboptimal scenario.

[11]Unlike the CF 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 interim flow values before convergence.

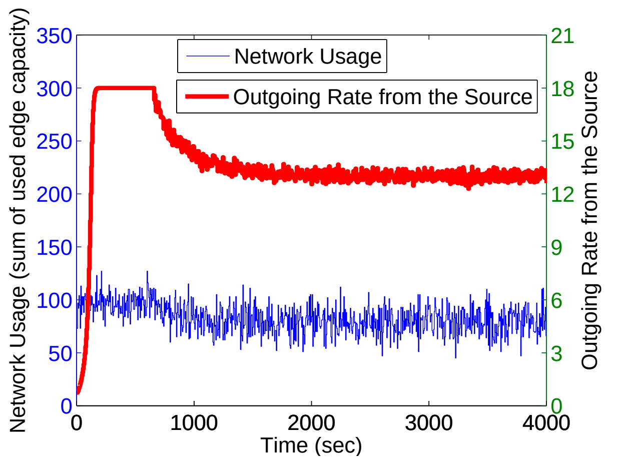


Fig. 7. The time evolution graphs of the back-pressure algorithm with the largest step size that ensure that the oscillation is within 5% of the max-flow value $\mathsf{MFV}(1,30)=13$.

B. Numerical Comparison with The Back Pressure Algorithm

To compare the designated max-flow algorithms such as P&R and the new CF algorithms with the generic LP-based ones, we have also applied the back-pressure (BP) algorithm [30] to the same acyclic network as in Section VII-A. For the BP algorithm, we set the utility function to be $U(x)=\log(1+x)$ where x is the achievable rate. We also choose the largest step-size $\beta=3\times10^{-4}$ such that the achievable is still within 5% of the max-flow value $\mathsf{MFV}(1,30)=13$. Every second, all edges update their rate allocation based on the difference of queue lengths between its neighbors. We also assume no interference so that exchanging the new rate-allocation and the queue-length information can be finished within the same second.

Fig. 7 depicts the time evolution curves of the network usage and the outgoing rate of the source for the BP algorithm. As a dual LP algorithm, the primal rate-allocation variables oscillate constantly, which explains the oscillating network usage in Fig. 7. The average network usage is 80, which is comparable to that of the P&R, and CF algorithms in Fig. 6. To obtain a stable plot for the flow rate, for each second we start from the dual queue-length variable q_S at the source q_S, solve the following maximization problem

$$\max_{x\geq 0} U(x) - \beta x q_S,$$

and plot the maximizing primal variable x^* in Fig. 7. As seen in Fig. 7, the x^* first reaches 18, which is the total outgoing capacity of Node 1. We need roughly 1500 seconds before BP converges to within 5% of the optimal max-flow value $\mathsf{MFV}(1,30) = 13$. If one would like to converge to 1% of the optimal value, a smaller step-size $\beta = 6 \times 10^{-5}$ needs to be used, and the BP algorithm takes roughly 12000 seconds before convergence. This demonstrates the efficiency of the designated max-flow algorithms in Fig. 6.

C. Numerical Experiment 2 — Multicast

The generalized CF algorithm with AB-MULTI-RE is able to find the union of several max-flows simultaneously for a multicast sessions from source s to destinations $\mathbf{d} = \{d_i\}$. We have implemented the proposed greedy search method in Section VI-D that finds a network coding solution with locally minimal total cost. The cost for each edge is assigned according to (2), which corresponds to the total power or time consumption of the network coding solution. In each round, AB-MULTI-RE selects the $E_R(v) \leftarrow \Xi \subseteq \mathrm{In}(v)$ for some v that has the largest average cost per edge, which will be removed from the current graph. We consider the same 30-node graph as discussed in Section VII-A. The source $s = 1$ and the destination set $\mathbf{d}$ can be one of the following three sets $\{30\}$, $\{29, 30\}$, and $\{28, 29, 30\}$. Again, we use the P&R algorithm as benchmark.

1) $\mathbf{d} = \{30\}$*:* In this unicast scenario, let $G_{\mathrm{P\&R},30}$ denote the output of the P&R algorithm with source $s = 1$ and destination $d = 30$. It takes 487 seconds for the P&R algorithm to converge to a max flow of value 13. There are 78 edges in $G_{\mathrm{P\&R},30}$ and the total cost is:

$$\sum_{e\in G_{\mathrm{P\&R},30}} c(e) \approx 11.7496.$$

Let $G_{\mathrm{Grdy},30}$ denote the output of the generalized CF algorithm with a greedy AB-IRE. It takes 395 seconds to converge. There are 77 edges in $G_{\mathrm{Grdy},30}$ and the total cost is:

$$\sum_{e\in G_{\mathrm{Grdy},30}} c(e) \approx 11.0028.$$

The generalized CF algorithm with greedy AB-MULTI-RE reduces the total power by 6.3%. For comparison, we use linear programming to find the globally minimal-cost solution, which has total cost 10.0226. The penalty of using a greedy solution solution over the optimal LP one is 8.9%.

The relatively small improvement is attributed to the fact that the P&R algorithm prefers sending traffic along one edge to sending it along several different edges as in each step the maximum amount of traffic is pushed toward a given neighbor. Therefore, the output graph will not use too many edges, which results in a reasonable total cost. As will be seen shortly, the advantages of the CF algorithm over the P&R algorithm are more substantial in the multicast scenario.

2) $\mathbf{d} = \{29, 30\}$*:* In this multicast scenario, the P&R algorithm needs to find the max flow between $(1, 29)$ in addition to finding the max flow between $(1, 30)$. Let $G_{\mathrm{P\&R},29}$ denote the output of the P&R algorithm with source $s = 1$ and destination $d = 29$. It takes 43 seconds for the P&R algorithm to converge to a max $(1, 29)$-flow with the corresponding max-flow value 18. The total convergence time for finding two max flows is $487 + 43 = 530$ seconds. Let $A_{\mathrm{P\&R},29}$, $A_{\mathrm{P\&R},30}$ denote the incidence matrices of $G_{\mathrm{P\&R},29}$ and $G_{\mathrm{P\&R},30}$ respectively. The final output will have an incidence matrix

$$A_{\mathrm{P\&R},\{29,30\}} = \max(A_{\mathrm{P\&R},29}, A_{\mathrm{P\&R},30}),$$

where the max operator takes the component-wise maximum of the two matrices.[12] Let $G_{\mathrm{P\&R},\{29,30\}}$ denote the corresponding final graph. There are 144 edges in $G_{\mathrm{P\&R},\{29,30\}}$ and the total cost is

$$\sum_{e\in G_{\mathrm{P\&R},\{29,30\}}} c(e) \approx 24.6210.$$

On the other hand, the generalized CF algorithm with a greedy AB-IRE finds the final output $G_{\mathrm{Grdy},\{29,30\}}$ directly without resorting to the two-staged approach of the existing P&R algorithm. Moreover, the generalized CF algorithm now only takes 355 seconds to converge,[13] which is faster than the unicast scenario. The reason is that edges now have to carry information for both destinations and thus there are less redundant edges to prune in the network. A locally minimal network coding solution can thus be reached sooner by the generalized CF algorithm. There are 133 edges in $G_{\mathrm{Grdy},\{29,30\}}$ and the total cost is

$$\sum_{e\in G_{\mathrm{Grdy},\{29,30\}}} c(e) \approx 18.8500.$$

The power saving of the generalized CF algorithm with greedy AB-MULTI-RE is 23.4% and the convergence time is only 67.0% of that of the two-staged P&R algorithm. A substantial improvement when compared to the unicast scenario. For comparison, the optimal LP solution has total cost 17.2036, which is 8.7% better than that of the greedy search.

3) $\mathbf{d} = \{28, 29, 30\}$*:* The last example is when there are three destinations. The P&R algorithm needs to find the max $(1, 28)$-flow in addition to finding the other two max flows. It takes 374 seconds to converge to a max $(1, 28)$-flow with max-flow value 9. The total convergence time for finding three max flows is $487 + 43 + 374 = 904$ seconds. Let $G_{\mathrm{P\&R},\{28,29,30\}}$ denote the combined final graph as discussed in the previous

[12]When network coding is allowed, we take the maximum between two max flows. For comparison, when network coding is prohibited, one has to take the summation of two max flows.

[13]We again emphasize that all destinations sustain the optimal rate even before the convergence of the final output, a unique feature of the CF algorithm.

case. There are 163 edges in $G_{\mathrm{P\&R},\{28,29,30\}}$ and the total cost is

$$\sum_{e\in G_{\mathrm{P\&R},\{28,29,30\}}} c(e) \approx 27.3294.$$

The generalized CF algorithm now takes only 344 seconds to converge. There are 142 edges in the output graph $G_{\mathrm{Grdy},\{28,29,30\}}$ and the total cost is

$$\sum_{e\in G_{\mathrm{Grdy},\{28,29,30\}}} c(e) \approx 19.8500.$$

The power saving of the generalized CF algorithm with greedy AB-MULTI-RE is 27.4% and the convergence time is only 38.1% of that of the three-staged P&R algorithm. For comparison, the optimal LP solution has total cost 18.2036, which is 8.3% better than that of the greedy search.

The power saving follows from that the generalized CF algorithm is able to consider all destinations simultaneously while the P&R algorithm searches for max flows in a sequential, one by one fashion without holistic consideration.

VIII. CONCLUSION

A new coded feedback (CF) max (s,d)-flow algorithm is provided for directed acyclic networks, which trims the network coding traffic by network coding. As designated max-flow algorithms, the new CF algorithms are asymptotically no slower than the existing ones, admit straightforward distributed network implementation, impose no additional hardware requirement on intermediate nodes, and sustain the optimal transmission rate even before the algorithms converge. The proposed CF algorithms can also search for the minimal union of multiple max flows for the multicast scenario within the same time of finding a single max flow. This work serves as a precursor to a new algorithmic study of network coding. We conclude this paper by a non-comprehensive list of future directions that we are actively investigating.

1) The coded feedback scheme provides a counter-measure to harness the power of broadcast. We are interested in the dynamics between this pair of opposite mechanisms. In a multiple unicast setup when several users are competing for network resources, a new rate-control scheme will be devised accordingly. We will also exploit this pair of mechanisms in non-stationary, time-varying networks. Jointly, the new rate-control algorithm and the results on time-varying networks will provide a new cross-layer framework of scheduling, route-finding, and coded feedback for wireless networks.
2) Improved pipelining: Waiting for all incoming messages to stabilize before starting trimming the graph is a bit conservative. In simulations, many graph-pruning opportunities can be identified even before all messages are stabilized. The properly designed pipelining will accelerate the convergence speed, which mitigates further the negative impact of initial broadcast.
3) For a practical network with delay, any directed cyclic graph is decoupled naturally into its acyclic counterpart along the time axis [21] and the CF algorithm can be applied directly in a distributed fashion. For network coding applications with generation-based structure, there is no advantage in considering cyclic networks. On the other hand, it is, in theory, an interesting and open question whether this network-coding-based approach can be generalized to cyclic networks as well, especially when the P&R algorithm is applicable to both acyclic and cyclic networks. To solve this question, new methods of constructing network coding for cyclic networks have to be investigated. The results will complement our understanding of conveying subspace information by coded feedback in a cyclic network.

APPENDIX A
PROOFS OF PROPOSITIONS 1 AND 2

Without loss of generality, we assume that all vertices are reachable by s and we assume that d is reachable by all vertices. Define $R \triangleq \langle m(u,d) : (u,d) \in \mathrm{In}(d)\rangle$, $\kappa \triangleq \dim(R)$. After GB-IRE processes the destination d, it is guaranteed that $|\mathrm{In}(d)| = \kappa$. Let $\left\{\overset{\leadsto}{v}\right\}$ 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 $\forall z \in Z, \left\{\overset{\leadsto}{z}\right\} \subseteq Z$. Define the edge collection C_Z for a separating Z:

$$C_Z \triangleq \{(u,v) \in E : u \notin Z, v \in Z\}.$$

We say C_Z is a *parallel edge cut.*[14] If C_Z contains i edges, we say C_Z is a parallel i-edge cut.

Lemma 1: After GB-IRE processes d, we have

$$[q(v,d) : (v,d) \in \mathrm{In}(d)]\,[m(v,d) : (v,d) \in \mathrm{In}(d)]^{\mathrm{T}} = I_\kappa.$$

Proof: If $\kappa = n$, then $|\mathrm{In}(d)| = \kappa = n$. By the construction of $q(v,d)$, we have

$$[q(v,d) : (v,d) \in \mathrm{In}(d)]^{\mathrm{T}}\,[m(v,d) : (v,d) \in \mathrm{In}(d)] = I_n = I_\kappa.$$

Since $[q(v,d) : (v,d) \in \mathrm{In}(d)]$ and $[m(v,d) : (v,d) \in \mathrm{In}(d)]$ are $n \times n$ square matrices, we have

$$[q(v,d) : (v,d) \in \mathrm{In}(d)]\,[m(v,d) : (v,d) \in \mathrm{In}(d)]^{\mathrm{T}} = I_\kappa. \tag{3}$$

For the case in which $|\mathrm{In}(d)| = \kappa < n$, since in Line 2 of COMPUTE CODED FEEDBACK, we choose additional m_a^* to complement the incoming coding vectors. By the same argument as before, we have

$$\begin{bmatrix} [q(v,d) : (v,d) \in \mathrm{In}(d)] \\ [q_a^* : a = \kappa+1, \cdots, n] \end{bmatrix} \begin{bmatrix} [m(v,d) : (v,d) \in \mathrm{In}(d)] \\ [m_a^* : a = \kappa+1, \cdots, n] \end{bmatrix}^{\mathrm{T}} = I_n. \tag{4}$$

[14]The definitions of an "edge cut" and a "parallel edge cut" are not equivalent. Any parallel edge cut C_Z constructed as above is a valid "edge cut," namely, removing all edges in C_Z will "disconnect source s and destination d." However, not all edge cuts are parallel edge cuts. For example, consider a graph $G = (V,E)$ where $V = \{s, v_1, v_2, d\}$ and $E = \{(s,v_1), (s,v_2), (v_1,v_2), (v_1,d), (v_2,d)\}$. Then $\{(s,v_1), (v_2,d)\}$ is an edge cut but not a parallel edge cut as one cannot find the corresponding separating vertex set Z. The parallel edge cut is equivalent to the *regular edge cut* defined in the context of network error correction as in [38].

Therefore (3) holds for the case $\kappa < n$ as well. The proof is thus complete. ■

We can then prove the following central lemma.

Lemma 2: Suppose $|\mathrm{In}(d)| = \kappa = \mathrm{Rank}([m(v,d) : (v,d) \in \mathrm{In}(d)])$ after executing subroutines COMPUTE FORWARD MESSAGES and COMPUTE CODED FEEDBACK in Lines 3 and 4 of the BASIC CF ALGORITHM with GB-IRE. Then for any separating vertex set $Z \subseteq V$ with $|C_Z| = \kappa$, the following equation must hold:

$$[q(u,v) : (u,v) \in C_Z]\,[m(u,v) : (u,v) \in C_Z]^{\mathrm{T}} = I_\kappa.$$

Proof: We first note that $[q(u,v) : (u,v) \in C_Z]$ and $[m(u,v) : (u,v) \in C_Z]$ are matrices of size $\kappa \times n$ since each vector $q(u,v)$ and $m(u,v)$ are n-dimensional row vectors. By the algebraic framework of network coding in [20], we have

$$[m(w,d) : (w,d) \in \mathrm{In}(d)] = A\,[m(u,v) : (u,v) \in C_Z]\,, \quad (5)$$

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

COMPUTE CODED FEEDBACK ensures that the transpose of the same mixing coefficients $\Gamma(v)$ used for constructing $m(\cdot,\cdot)$ are also used to construct the feedback messages $q(\cdot,\cdot)$. Therefore

$$[q(u,v) : (u,v) \in C_Z] = A^{\mathrm{T}}\,[q(w,d) : (w,d) \in \mathrm{In}(d)]\,. \quad (6)$$

Jointly, (5), (6), Lemma 1, and the invertibility of A imply that

$$\begin{aligned}
&[q(u,v) : (u,v) \in C_Z]\,[m(u,v) : (u,v) \in C_Z]^{\mathrm{T}} \\
&= A^{\mathrm{T}}\,[q(w,d) : (w,d) \in \mathrm{In}(d)] \\
&\qquad \cdot [m(w,d) : (w,d) \in \mathrm{In}(d)]^{\mathrm{T}}\,(A^{-1})^{\mathrm{T}} \\
&= A^{\mathrm{T}} I_\kappa (A^{-1})^{\mathrm{T}} = I_\kappa.
\end{aligned}$$

The proof is complete. ■

Based on the above result, we have the following lemma.

Lemma 3: For the CF algorithm with GB-IRE, suppose v_0 is the node that has just been searched (for useful E_{v_0}) during GB-IRE. After the removal of the redundant edges $E_R(v_0)$ in Line 7 of the BASIC CF ALGORITHM, there are exactly κ edges in C_Z for any separating vertex set $Z \subseteq \left\{\overset{\rightsquigarrow}{v_0}\right\}$.

Proof: We prove this lemma by induction. The basic case is when $v_0 = d$. In this case, Lemma 3 holds since the only separating Z satisfying $Z \subseteq \left\{\overset{\rightsquigarrow}{d}\right\}$ is $\{d\}$ and $|C_{\{d\}}| = |\mathrm{In}(d)| = \kappa$.

We note that the purpose of checking whether $E_v \neq \mathrm{In}(v)$ in Line 11 of GB-IRE is simply to accelerate the convergence. If $E_v = \mathrm{In}(v)$, then $E_R(v) = \emptyset$ and no edge will be removed. The subroutines COMPUTE FORWARD MESSAGES and COMPUTE CODED FEEDBACK in Lines 3 and 4 of the BASIC CF ALGORITHM thus do not change the forward and feedback messages. In this case, one can directly search for the next $v \leftarrow v - 1$ as in the WHILE-LOOP of GB-IRE. As a result, we can assume node $v_0 + 1, v_0 + 2, \cdots, d$ was searched in the previous $(d - v_0)$ rounds as if no checking of $E_v \neq \mathrm{In}(v)$ is performed, while v_0 has just been searched in the current round.

Any $Z \subseteq \left\{\overset{\rightsquigarrow}{v_0}\right\}$ can be categorized into two cases: $Z \subsetneq \left\{\overset{\rightsquigarrow}{v_0}\right\}$ or $Z = \left\{\overset{\rightsquigarrow}{v_0}\right\}$.

Case 1: $Z \subsetneq \left\{\overset{\rightsquigarrow}{v_0}\right\}$. Let $w_1, \cdots, w_{|\mathrm{Out}(v_0)|}$ denote the $|\mathrm{Out}(v_0)|$ distinct children of v_0. If $Z \subseteq \left\{\overset{\rightsquigarrow}{w_1}\right\}$, then the statement holds by induction since $w_1 > v_0$.

Suppose that the statement holds for all $Z \subseteq \bigcup_{i=1}^{a} \left\{\overset{\rightsquigarrow}{w_i}\right\}$ with some $a \le |\mathrm{Out}(v_0)| - 1$. For those $Z \subseteq \bigcup_{i=1}^{a+1} \left\{\overset{\rightsquigarrow}{w_i}\right\}$ but not a subset of $\bigcup_{i=1}^{a} \left\{\overset{\rightsquigarrow}{w_i}\right\}$, define a partition $\{Z_1, Z_2, Z_3\}$ of Z as follows.

$$Z_1 = Z \cap \left(\bigcup_{i=1}^{a} \left\{\overset{\rightsquigarrow}{w_i}\right\} \setminus \left\{\overset{\rightsquigarrow}{w_{a+1}}\right\} \right) \quad (7)$$

$$Z_2 = Z \cap \left(\left\{\overset{\rightsquigarrow}{w_{a+1}}\right\} \setminus \bigcup_{i=1}^{a} \left\{\overset{\rightsquigarrow}{w_i}\right\} \right) \quad (8)$$

$$Z_3 = Z \cap \left(\left\{\overset{\rightsquigarrow}{w_{a+1}}\right\} \cap \bigcup_{i=1}^{a} \left\{\overset{\rightsquigarrow}{w_i}\right\} \right). \quad (9)$$

We also define the following five disjoint sets of edges

$$C_1 = \{(u,v) \in E : u \notin Z, v \in Z_1\} \quad (10)$$
$$C_2 = \{(u,v) \in E : u \notin Z, v \in Z_2\} \quad (11)$$
$$C_3 = \{(u,v) \in E : u \notin Z, v \in Z_3\} \quad (12)$$
$$C_4 = \{(u,v) \in E : u \in Z_1, v \in Z_3\} \quad (13)$$
$$C_5 = \{(u,v) \in E : u \in Z_2, v \in Z_3\}. \quad (14)$$

Since $C_Z = C_1 \cup C_2 \cup C_3$, we would like to show that $|C_1| + |C_2| + |C_3| = \kappa$. We first notice that the three vertex sets $Z_3 \subseteq \left\{\overset{\rightsquigarrow}{w_{a+1}}\right\}$, $(Z_1 \cup Z_3) \subseteq \bigcup_{i=1}^{a} \left\{\overset{\rightsquigarrow}{w_i}\right\}$, and $(Z_2 \cup Z_3) \subseteq \left\{\overset{\rightsquigarrow}{w_{a+1}}\right\}$ are all separating vertex sets. By induction, each of the three edge sets $C_{Z_3} = C_3 \cup C_4 \cup C_5$, $C_{Z_1 \cup Z_3} = C_1 \cup C_3 \cup C_5$, and $C_{Z_2 \cup Z_3} = C_2 \cup C_3 \cup C_4$ contains exactly κ edges. As a result,

$$\begin{aligned}
|C_3| + |C_4| + |C_5| &= |C_1| + |C_3| + |C_5| \\
&= |C_2| + |C_3| + |C_4| = \kappa,
\end{aligned}$$

which implies $|C_1| + |C_2| + |C_3| = \kappa$. By induction, Lemma 3 holds for $Z \subseteq \bigcup_{i=1}^{|\mathrm{Out}(v_0)|} \left\{\overset{\rightsquigarrow}{w_i}\right\}$ and thus for $Z \subsetneq \left\{\overset{\rightsquigarrow}{v_0}\right\}$. The proof of Case 1 is complete.

Case 2: $Z = \left\{\overset{\rightsquigarrow}{v_0}\right\}$. Let $Z_{\setminus v_0} = \bigcup_{i=1}^{|\mathrm{Out}(v_0)|} \left\{\overset{\rightsquigarrow}{w_i}\right\}$, for which the corresponding $C_{Z_{\setminus v_0}}$ contains κ edges as just proven in Case 1. Consider the subroutine GB-IRE before the actual removal of $E_R(v_0)$. By applying Lemma 2 to $C_{Z_{\setminus v_0}}$ and focusing only on the $q(\cdot,\cdot)$ and $m(\cdot,\cdot)$ corresponding to edges in $\mathrm{Out}(v_0)$, we have

$$\begin{bmatrix} q(v_0, w_1) \\ \vdots \\ q(v_0, w_{|\mathrm{Out}(v_0)|}) \end{bmatrix} \begin{bmatrix} m(v_0, w_1) \\ \vdots \\ m(v_0, w_{|\mathrm{Out}(v_0)|}) \end{bmatrix}^{\mathrm{T}} = I_{|\mathrm{Out}(v_0)|}.$$

By the definition of the transfer matrix $\Gamma(v_0)$, we have the following equation

$$\begin{bmatrix} m(v_0, w_1) \\ \vdots \\ m(v_0, w_{|\mathrm{Out}(v_0)|}) \end{bmatrix} = \Gamma(v_0) \begin{bmatrix} m(u_1, v_0) \\ \vdots \\ m(u_{|\mathrm{In}(v_0)|}, v_0) \end{bmatrix}.$$

From the above two matrix equations, the product of two matrices Φ and $\Gamma(v_0)^{\mathrm{T}}$ becomes

$$\begin{aligned}&\Phi\Gamma(v_0)^{\mathrm{T}}\\ &=\begin{bmatrix} q(v_0,w_1)\\ \vdots\\ q(v_0,w_{|\mathrm{Out}(v_0)|})\end{bmatrix}\begin{bmatrix} m(u_1,v_0)\\ \vdots\\ m(u_{|\mathrm{In}(v_0)|},v_0)\end{bmatrix}^{\mathrm{T}}\Gamma(v_0)^{\mathrm{T}}\\ &=\begin{bmatrix} q(v_0,w_1)\\ \vdots\\ q(v_0,w_{|\mathrm{Out}(v_0)|})\end{bmatrix}\begin{bmatrix} m(v_0,w_1)\\ \vdots\\ m(v_0,w_{|\mathrm{Out}(v_0)|})\end{bmatrix}^{\mathrm{T}}\\ &=I_{|\mathrm{Out}(v_0)|}.\end{aligned}$$

Therefore, the $|\mathrm{Out}(v_0)|\times|\mathrm{In}(v_0)|$ matrix Φ must have column rank at least $|\mathrm{Out}(v_0)|$. As a result, $|\mathrm{In}(v_0)|\geq|\mathrm{Out}(v_0)|$. Moreover, a valid choice of E_{v_0} in Line 10 of GB-IRE always exists and the final output E_{v_0} has $|E_{v_0}|=|\mathrm{Out}(v_0)|$. There are two subcases depending on whether $E_{v_0}=\mathrm{In}(v_0)$.

Case 2.1: $E_{v_0}=\mathrm{In}(v_0)$. In this case, since

$$|C_Z|=|C_{Z_{\backslash v_0}}|-|\mathrm{Out}(v_0)|+|\mathrm{In}(v_0)|=\kappa,$$

Lemma 3 is proven for Case 2.1.

Case 2.2: $E_{v_0}\subsetneq\mathrm{In}(v_0)$. In this case, we still has $|C_Z|=|C_{Z_{\backslash v_0}}|$ since after the removal of $E_R(v_0)=\mathrm{In}(v_0)\backslash E_{v_0}$, the new graph has $|\mathrm{In}(v_0)|=|\mathrm{Out}(v_0)|$. However, removing edges in $E_R(v_0)$ may change the $|\mathrm{In}(d)|$ messages received by d and they may not be independent anymore. Hence, it remains to show that after the removal of $E_R(v_0)$ edges (in Lines 7 of the BASIC CF ALGORITHM), $\dim(R')$, the dimension of the new received space remains unchanged and equals $\kappa=\dim(R)$.

In the following, the primed notation, such as R', m', etc., refers to the new quantities after the edge removal. Without loss of generality, assume $E_{v_0}=\{(u_1,v_0),\cdots,(u_{|\mathrm{Out}(v_0)|},v_0)\}$ being the first $|\mathrm{Out}(v_0)|$ edges of $\mathrm{In}(v_0)$. Let $\{m(x_i,d)\}$ denote the κ received messages before the graph change, and use $m^*_{\kappa+1},\cdots,m^*_n$ to denote the complementing independent vectors during the construction of coded feedback in Line 2 of COMPUTE CODED FEEDBACK. Note that by (4) in the proof of Lemma 1 we have $\forall i\leq\kappa,\forall a>\kappa,q(x_i,d)\cdot m^*_a=0$.

Consider an edge set $C'_Z\triangleq\left(C_{Z_{\backslash v_0}}\backslash\mathrm{Out}(v_0)\right)\cup E_{v_0}$ of κ edges. By expressing the original κ messages in the edge set C'_Z using the basis $\{m(x_i,d)\}_{i\leq\kappa}$ and $\{m^*_a\}_{a>\kappa}$, we have

$$\mathbf{m}\triangleq\begin{bmatrix} m(y_1,z_1)\\ \vdots\\ m(y_{\kappa-|\mathrm{Out}(v_0)|},z_{\kappa-|\mathrm{Out}(v_0)|})\\ m(u_1,v_0)\\ \vdots\\ m(u_{|\mathrm{Out}(v_0)|},v_0)\end{bmatrix}=B\begin{bmatrix} m(x_i,d)\\ \vdots\\ m(x_\kappa,d)\\ m^*_{\kappa+1}\\ \vdots\\ m^*_n\end{bmatrix}$$

for some $\kappa\times n$ matrix B, where (y_i,z_i) are edges in $C'_Z\backslash E_{v_0}=C_{Z_{\backslash v_0}}\backslash\mathrm{Out}(v_0)$.

After the removal of $E_R(v_0)=\mathrm{In}(v_0)\backslash E_{v_0}$, the new received messages are

$$\begin{aligned}\begin{bmatrix} m'(x_1,d)\\ \vdots\\ m'(x_\kappa,d)\end{bmatrix}&=A\begin{bmatrix} I_{\kappa-|\mathrm{Out}(v_0)|} & 0\\ 0 & \Gamma'(v_0)\end{bmatrix}\mathbf{m}\\ &=A\begin{bmatrix} I_{\kappa-|\mathrm{Out}(v_0)|} & 0\\ 0 & \Gamma'(v_0)\end{bmatrix}B\begin{bmatrix} m(x_i,d)\\ \vdots\\ m(x_\kappa,d)\\ m^*_{\kappa+1}\\ \vdots\\ m^*_n\end{bmatrix},\end{aligned}$$

where A is the $\kappa\times\kappa$ transfer matrix from $C_{Z_{\backslash v_0}}$ to $\mathrm{In}(d)$ and the matrix in the middle is the transfer matrix from C'_Z to $C_{Z_{\backslash v_0}}$. As a result, the necessary and sufficient condition that all κ messages $m'(x_1,d),\cdots,m'(x_\kappa,d)$ received by d are independent is that the following matrix being of full row rank.

$$A\begin{bmatrix} I_{\kappa-|\mathrm{Out}(v_0)|} & 0\\ 0 & \Gamma'(v_0)\end{bmatrix}B. \tag{15}$$

We first note that the A matrix is of full rank since before the edge removal, the κ received messages are independent.

Following the above argument, a sufficient condition that (15) is of full rank is when the corresponding right-multiplied matrix

$$A\begin{bmatrix} I_{\kappa-|\mathrm{Out}(v_0)|} & 0\\ 0 & \Gamma'(v_0)\end{bmatrix}B\begin{bmatrix} A\\ 0_{(n-\kappa)\times\kappa}\end{bmatrix} \tag{16}$$

is of full row rank. In the following, we focus on proving (16) being of full row rank.

We first notice that

$$\begin{aligned}&\mathbf{m}\begin{bmatrix} q(y_1,z_1)\\ \vdots\\ q(y_{\kappa-|\mathrm{Out}(v_0)|},z_{\kappa-|\mathrm{Out}(v_0)|})\\ q(v_0,w_1)\\ \vdots\\ q(v_0,w_{|\mathrm{Out}(v_0)|})\end{bmatrix}^{\mathrm{T}} \qquad (17)\\ &=B\begin{bmatrix} m(x_i,d)\\ \vdots\\ m(x_\kappa,d)\\ m^*_{\kappa+1}\\ \vdots\\ m^*_n\end{bmatrix}\begin{bmatrix} q(x_1,d)\\ \vdots\\ q(x_\kappa,d)\end{bmatrix}^{\mathrm{T}}\left(A^{\mathrm{T}}\right)^{\mathrm{T}}\\ &=B\begin{bmatrix} I_\kappa\\ 0_{(n-\kappa)\times n}\end{bmatrix}A=B\begin{bmatrix} A\\ 0_{(n-\kappa)\times\kappa}\end{bmatrix}. \qquad (18)\end{aligned}$$

On the other hand, since the $\kappa-|\mathrm{Out}(v_0)|$ edges $(y_1,z_1),\cdots,(y_{\kappa-|\mathrm{Out}(v_0)|},z_{\kappa-|\mathrm{Out}(v_0)|})$ are a subset of the parallel edge cut $C_{Z_{\backslash v_0}}$ and $|C_{Z_{\backslash v_0}}|=\kappa$, we can rewrite (17) by invoking Lemma 2:

$$\mathbf{m}\begin{bmatrix} q(y_1, z_1) \\ \vdots \\ q(y_{\kappa-|\mathrm{Out}(v_0)|}, z_{\kappa-|\mathrm{Out}(v_0))|}) \\ q(v_0, w_1) \\ \vdots \\ q(v_0, w_{|\mathrm{Out}(v_0)|}) \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} I_{\kappa-|\mathrm{Out}(v)|} & 0 \\ Y & \left(\Phi|_{E_{v_0}}\right)^{\mathrm{T}} \end{bmatrix}$$

for some $|\mathrm{Out}(v)| \times (\kappa - |\mathrm{Out}(v)|)$ matrix Y, where $\Phi|_{E_{v_0}}$ is the sub-matrix of Φ corresponding to the E_{v_0} columns chosen in GB-IRE. By the construction of E_{v_0} in Line 10 of GB-IRE, $\Phi|_{E_{v_0}}$ is of full rank. Therefore, the right-hand side of (18) is a $\kappa \times \kappa$ matrix of full rank. Since A is a full rank square matrix and by construction $\Gamma'(v_0)$ is also a full rank square matrix, (16) is the product of three full-rank square $\kappa \times \kappa$ matrices. As a result, (16) must be of full rank and we have $\dim(R') = \kappa = \dim(R)$. This completes the proof. ■

Properties 1 and 3 in Proposition 1 are direct byproducts of the proof of Lemma 3. Property 2 in Proposition 1 is straightforward since it takes at most $2l$ seconds for the forward and feedback messages to stabilize, and the construction of GB-IRE guarantees that there are at most $|V|$ iterations in the main loop of BASIC CF ALGORITHM. Since in the proof of Lemma 3 we have already shown that once the CF algorithm stops, all $v \in V\backslash\{s, d\}$ must have $|\mathrm{Out}(v)| = |\mathrm{In}(v)|$, the output of the CF algorithm with GB-IRE must be a flow. Property 4 of Proposition 1 is thus proven.

Before proving Proposition 2, we provide the following lemma as a direct corollary of [Theorem 3.3 in [21]] regarding the generic LCM. The proof is a straightforward restatement of [Theorem 3.3 in [21]] and is thus omitted

Lemma 4 (From Theorem 3.3 in [21]): Consider any set of edges $C \subseteq E$. Construct G' from G by replacing each edge $(u, v) \in C$ by two serially concatenated edges (u, w) and (w, v). For any generic LCM, the dimension of the space spanned by messages in C equals the max flow value from s to all those new w's in G'.

Proof of Proposition 2: Consider any given max flow $F \subseteq E$ and any given generic LCM on G. Again, let $\kappa = \dim(R) = \mathrm{Out}(d)$ denote the dimension of the space received by destination d. We first modify the original graph by constructing $n - \kappa$ additional interior-vertex-disjoint paths connecting s and d. We carefully choose the coding vectors along the new paths and reuse the given generic LCM so that the augmented network coding solution on the new graph remains a generic LCM. This construction is always possible when the $\mathsf{GF}(q)$ size is sufficiently large. Let $m^*_{\kappa+1}, \cdots, m^*_n$ denote the coding vectors along the new paths such that the augmented network code is an LCM. Once the construction of $\{m^*_a\}$ is finished, choose arbitrarily the $q(v, d)$ vectors satisfying Line 3 in COMPUTE CODED FEEDBACK.

Suppose node v_0 is being searched in GB-IRE. We set E_{v_0} as $E_{v_0} = \mathrm{In}(v_0) \cap F$ and will show that such E_{v_0} is a valid choice according to Line 10 of GB-IRE. Without loss of generality, we assume that E_{v_0} contains the first $|\mathrm{Out}(v_0)|$ edges of $\mathrm{In}(v_0)$ since Lemma 3 guarantees that $\mathrm{Out}(v_0)$ is a subset of the given flow F. Following the same definitions in Lemma 3, we define

$$Z\backslash v_0 \triangleq \bigcup_{(v_0,w)\in \mathrm{Out}(v_0)} \left\{\overset{\leadsto}{w}\right\}$$

$$C'_Z \triangleq \left(C_{Z\backslash v_0}\backslash \mathrm{Out}(v_0)\right) \cup E_{v_0}$$

$$\Phi|_{E_{v_0}} \triangleq \text{the submatrix of } \Phi \text{ corresponding to the columns of } E_{v_0}.$$

Let (y_i, z_i) denote the edges in $C'_Z\backslash E_{v_0}$ and let $\kappa' = \kappa - |\mathrm{Out}(v_0)|$. By noticing that $\{q(y_i, z_i) : i = 1, \cdots, \kappa'\}$ and $\{q(v_0, w) : (v_0, w) \in \mathrm{Out}(v_0)\}$ are the linear combinations of $\{q(v, d) : (v, d) \in \mathrm{In}(d)\}$ and by (4), we have for all $q \in \{q(y_i, z_i) : i = 1, \cdots, \kappa'\} \cup \{q(v_0, w) : (v_0, w) \in \mathrm{Out}(v_0)\}$ and for all $a \in \{\kappa + 1, \cdots, n\}$,

$$q \cdot m^*_a = 0. \tag{19}$$

By Lemma 2, we thus have

$$\begin{bmatrix} q(y_1, z_1) \\ \vdots \\ q(y_{\kappa'}, z_{\kappa'}) \\ q(v_0, w_1) \\ \vdots \\ q(v_0, w_{|\mathrm{Out}(v)|}) \end{bmatrix} \begin{bmatrix} m(y_1, z_1) \\ \vdots \\ m(y_{\kappa'}, z_{\kappa'}) \\ m(u_1, v_0) \\ \vdots \\ m(u_{|\mathrm{Out}(v_0)|}, v_0) \\ m^*_{\kappa+1} \\ \vdots \\ m^*_n \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} I_{\kappa'} & Y_1 & 0 \\ 0 & \Phi|_{E_{v_0}} & 0 \end{bmatrix} \tag{20}$$

since the first κ' edges are part of the parallel κ-edge cut $C_{Z_{\backslash v_0}}$. Y_1 is a matrix of size $\kappa' \times |\mathrm{Out}(v_0)|$. By (19), the $(\kappa + 1)$-th to the n-th columns of (20) are all zero.

Since the feedback messages $q(\cdot, \cdot)$ in the left-hand side of (20) correspond to the parallel κ-edge cut $C_{Z_{\backslash v_0}}$, those feedback messages must form an $\kappa \times n$ matrix of full row rank. Otherwise, the product of $q(\cdot, \cdot)$ and $m(\cdot, \cdot)$ on the parallel κ-edge cut cannot be an identity matrix, which contradicts Lemma 2. On the other hand, the forward coding vectors $m(\cdot, \cdot)$ in the left-hand side of (20) corresponds to edges on the given max flow F since we choose $E_{v_0} \leftarrow \mathrm{In}(v_0) \cap F$. Lemma 4 guarantees that those $m(\cdot, \cdot)$ are also independent and form an $n \times n$ invertible matrix since the auxiliary messages $m^*_{\kappa+1}, \cdots, m^*_n$ and the existing messages form jointly a generic LCM.

By the above reasoning, (20) must be of full row rank, which implies $\Phi|_{E_{v_0}}$ being of full rank. Therefore $E_{v_0} \leftarrow \mathrm{In}(v_0) \cap F$ is a valid choice satisfying Line 10 of GB-IRE.

Since $\Gamma(v_0)$ is associated with a generic LCM, the sub-matrix of $\Gamma(v_0)$ corresponding to the E_{v_0} columns must be of full rank. Line 15 is never executed and there is no need to find a new $\Gamma'(v_0)$.

The above choices of $\{m^*_a\}$, $\{q(v, d)\}$, E_{v_0}, and $\Gamma'(v_0)$ remove edges in $\mathrm{In}(v_0)\backslash F$. By repeatedly applying the above

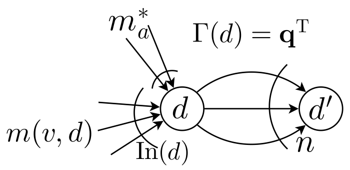


Fig. 8. Illustration for the case $\mathrm{In}(d) \neq \kappa$.

procedure and constructing a sequence of choices of $\{m_a^*\}$, $\{q(v,d)\}$, E_{v_0}, and $\Gamma'(v)$, any edges outside the given max flow F will be removed and Proposition 2 is proven. ■

APPENDIX B PROOFS OF PROPOSITIONS 3 TO 5

We need the following central lemma for proving the correctness of the CF algorithm when AB-IRE is used, which a corollary of the Sylvester's determinant theorem for the finite field.

Consider two $m \times n$ matrices A and B in $\mathsf{GF}(q)$ where $m \leq n$.

Lemma 5: The following two statements imply each other.

- The square matrix $I_m - AB^{\mathrm{T}}$ is of full rank.
- The square matrix $I_n - A^{\mathrm{T}}B$ is of full rank.

Proof: We first observe that

$$\begin{pmatrix} I_m & A \\ B^{\mathrm{T}} & I_n \end{pmatrix} = \begin{pmatrix} I_m & \mathbf{0} \\ B^{\mathrm{T}} & I_n \end{pmatrix} \begin{pmatrix} I_m & A \\ \mathbf{0} & I_n - B^{\mathrm{T}}A \end{pmatrix} = \begin{pmatrix} I_m & A \\ \mathbf{0} & I_n \end{pmatrix} \begin{pmatrix} I_m - AB^{\mathrm{T}} & \mathbf{0} \\ B^{\mathrm{T}} & I_n \end{pmatrix}. \tag{21}$$

Since both statements in Lemma 5 are equivalent to that (21) has full rank, the proof is complete. ■

Proof of Proposition 3: The computation time analysis of the CF algorithm with AB-IRE is straightforward. Since each iteration of the main loop in the BASIC CF ALGORITHM will remove at least one edge, there are at most $|E|$ iterations before termination. Each iteration takes at most $2l$ seconds for both the forward and feedback messages to stabilize. The entire algorithm thus stops within $2l|E|$ seconds. In each iteration of the main loop, at most $|E|$ feedback messages $q(\cdot,\cdot)$ are generated. The total number of feedback messages generated in the network is thus no larger than $|E|^2$.

Similar to the proof of Proposition 1, we let κ denote the dimension of the space received by d. Without loss of generality, we can assume that $|\mathrm{In}(d)| = \mathrm{Rank}(d) = \kappa$.[15] Let $\{m(x_i,d)\}$ denote the κ received messages before the graph change. If $\kappa < n$, then we use $m_{\kappa+1}^*, \cdots, m_n^*$ to denote the complementing independent vectors such that jointly $\{m(x_i,d)\}_{i\leq\kappa}$ and $\{m_a^*\}_{a>\kappa}$ form a basis of the global vector space Ω.

[15]If not, we simply need to introduce an auxiliary node d'. Add n parallel edges connecting d and d' and let the transfer matrix of d be $\Gamma(d) = \mathbf{q}^{\mathrm{T}}$. See Fig. 8 for illustration where d has $|\mathrm{In}(d)|$ incoming edges and $(n-\kappa)$ virtual pipes carrying $\{m_a^*\}$ as described in Line 3 of COMPUTE CODED FEEDBACK. All our following analysis then holds verbatim when focusing on the new destination d'.

Since $\{m(x_i,d)\}_{i\leq\kappa}$ and $\{m_a^*\}_{a>\kappa}$ form a basis of Ω, let B be the conversion matrix for $[m(u_i,v) : (u_i,v) \in E_R(v)]$ such that

$$\begin{bmatrix} m(u_1,v) \\ \vdots \\ m(u_{\kappa''},v) \end{bmatrix} = B \begin{bmatrix} m(x_1,d) \\ \vdots \\ m(x_\kappa,d) \\ m_{\kappa+1}^* \\ \vdots \\ m_n^* \end{bmatrix},$$

where $\kappa'' \triangleq |E_R(v)|$. Let A denote the transfer matrix from $E_R(v)$ to $\mathrm{In}(d)$. Then removing edges in $E_R(v)$ will results in new messages

$$\begin{bmatrix} m'(x_1,d) \\ \vdots \\ m'(x_\kappa,d) \end{bmatrix} = \begin{bmatrix} m(x_1,d) \\ \vdots \\ m(x_\kappa,d) \end{bmatrix} - A \begin{bmatrix} m(u_1,v) \\ \vdots \\ m(u_{\kappa''},v) \end{bmatrix}. \tag{22}$$

For the above κ new messages to be linearly independent, a sufficient condition is that the following square matrix

$$\begin{bmatrix} m'(x_1,d) \\ \vdots \\ m'(x_\kappa,d) \\ m_{\kappa+1}^* \\ \vdots \\ m_n^* \end{bmatrix} = \begin{bmatrix} m(x_1,d) \\ \vdots \\ m(x_\kappa,d) \\ m_{\kappa+1}^* \\ \vdots \\ m_n^* \end{bmatrix} - \begin{bmatrix} A \\ 0_{(n-\kappa)\times\kappa''} \end{bmatrix} \begin{bmatrix} m(u_1,v) \\ \vdots \\ m(u_{\kappa''},v) \end{bmatrix} = \left(I_n - \begin{bmatrix} A \\ 0_{(n-\kappa)\times\kappa''} \end{bmatrix} B\right) \begin{bmatrix} m(x_1,d) \\ \vdots \\ m(x_\kappa,d) \\ m_{\kappa+1}^* \\ \vdots \\ m_n^* \end{bmatrix} \tag{23}$$

is of full rank. This is equivalent to

$$I_n - \begin{bmatrix} A \\ 0_{(n-\kappa)\times\kappa''} \end{bmatrix} B \tag{24}$$

being of full rank. By Lemma 5, (24) being of full rank is equivalent to

$$I_{\kappa''} - \begin{bmatrix} A \\ 0_{(n-\kappa)\times\kappa''} \end{bmatrix}^{\mathrm{T}} B^{\mathrm{T}} \tag{25}$$

being of full rank. We then notice that by definition and by

Lemma 1,

$$\Pi = \begin{bmatrix} q(u_1,d) \\ \vdots \\ q(u_{\kappa''},d) \end{bmatrix} \begin{bmatrix} m(u_1,d) \\ \vdots \\ m(u_{\kappa''},d) \end{bmatrix}^{\mathrm{T}}$$

$$= A^{\mathrm{T}} \begin{bmatrix} q(x_1,d) \\ \vdots \\ q(x_\kappa,d) \end{bmatrix} \begin{bmatrix} m(x_1,d) \\ \vdots \\ m(x_\kappa,d) \\ m^*_{\kappa+1} \\ \vdots \\ m^*_n \end{bmatrix}^{\mathrm{T}} B^{\mathrm{T}}$$

$$= A^{\mathrm{T}} \left[I_\kappa \; 0_{\kappa\times(n-\kappa)} \right] B^{\mathrm{T}} = \begin{bmatrix} A \\ 0_{(n-\kappa)\times\kappa''} \end{bmatrix}^{\mathrm{T}} B^{\mathrm{T}}.$$

Since the construction of $E_R(v)$ guarantees $I_{\kappa''} - \Pi$ being of full rank, (25) must be of full rank. Therefore, removing edges in $E_R(v)$ will not change the dimension of the received space.

To prove the third property of Proposition 3, we notice that when $\kappa = n$, (23) collapses back to (22). Therefore, the condition

$$I_{\kappa''} - \Pi$$

being of full rank is not only sufficient but also necessary for (22) being of full rank. As a result, if there exists any non-empty subset of $\mathrm{In}(v)$ that can be removed locally without reducing the dimension of the received space, then the same non-empty subset is a valid choice of $E_R(v)$ such that $I_{\kappa''} - \Pi$ being of full rank. Therefore, when the CF algorithm stops, there is no valid $E_R(v)$ and the output network coding solution has locally minimal network usage. The proof of Proposition 3 is complete. ∎

Proof of Proposition 4: We first consider the case in which the dimension of $\langle m(x_i,d) : (x_i,d) \in \mathrm{In}(d)\rangle$ is n. In this case, the condition $I_{|E_R(v)|} - \Pi$ being of full rank is a necessary and sufficient condition that $E_R(v) \subseteq \mathrm{In}(v)$ can be safely removed without changing the dimension of the received space (see the proof of Proposition 3). Consider any edge (u,v) such that the removal of (u,v) will not change the max-flow value $\mathsf{MFV}(s,d)$. Since the underlying network coding solution is a generic LCM, removing the single edge (u,v) will not change the dimension of the received space. As a result, the 1-edge set $E_R(v) \leftarrow \{(u,v)\}$ is a valid redundant edge set. From the above reasoning, when the CF algorithm stops, removing any edge in the final graph will decrease the max-flow value $\mathsf{MFV}(s,d)$. The final output graph must be a max (s,d)-flow.

When the dimension of $\langle m(x_i,d) : (x_i,d) \in \mathrm{In}(d)\rangle$, denoted by κ, is strictly less than n, we need to consider the augmented network coding solution as first discussed in the proof of Proposition 2. We modify the original graph by constructing $n-\kappa$ additional *pipes* connecting s and d. Choose carefully the coding vectors $m^*_{\kappa+1}, \cdots, m^*_n$ along the new pipes and reuse the given generic LCM so that the augmented network coding solution on the new graph remains a generic LCM. This construction is always possible when the $\mathsf{GF}(q)$ size is sufficiently large.

In the augmented network coding solution, the dimension of the received space is n. By the same argument of the first case, the CF algorithm with AB-IRE will return a max (s,d)-flow of the new graph. We notice that when applying the CF algorithm to the original graph, the output is simply the projection of the result of the case in which we apply the CF algorithm to the new graph. Since projecting the max (s,d)-flow of the new graph to the original graph results in a max (s,d)-flow of the original graph, the output of the CF algorithm for the original graph must be a max (s,d)-flow. ∎

Proof of Proposition 5: As in the proof of Proposition 4, we first consider the case in which the dimension of $\langle m(x_i,d) : (x_i,d) \in \mathrm{In}(d)\rangle$ is n. The assumption that the underlying network coding solution is a generic LCM implies that the removal of any redundant edges can be performed locally. Namely, if removing an edge set E' does not change the dimension of the received space, then sequentially removing (arbitrary) partitions of E' in any arbitrary order does not change the dimension of the received space throughout the sequential edge removal process. As a result, when each vertex removes the largest $E_R(v)$ during each iteration, the CF algorithm does not have to revisit any v for the second time. The CF algorithm will thus stop within at most $|V|$ iterations of the main loop. The complexity statement in Proposition 5 follows naturally.

The use of the generic LCM also bridges the gap between graph-theoretic statements regarding max flows and the algebraic statements regarding the dimension of the received space. Since removing any edge outside the given max flow F will not change the dimension of the received space, we can choose $E_R(v) \leftarrow \mathrm{In}(v)\backslash F$. The output of the CF algorithm is thus the given max flow F. Note that the complexity and flexibility statements complement each other. Namely, with the underlying network coding solution being a generic LCM, the CF algorithm can output any given max flow while still stops within $2l|V|$ seconds.

For the case in which the dimension of the received space is strictly less than n, we can follow the same approach of considering the augmented network coding solution as introduced in the proof of Proposition 4. The proof is thus complete.

∎

Appendix C
Proof of Proposition 6

Proof of Proposition 6: This proof is a simple extension of the proof of Proposition 3.

By noting that each $E_R(v)$ contains at least one edge, the complexity statement follows verbatim that of the proof of Proposition 3.

By noting that the new output $E_R(v)$ in AB-MULTI-RE is the intersection of all redundant edge sets for each destination d_i respectively, removing $E_R(v)$ will not change the dimensions of the received spaces for any $d_i \in \mathbf{d}$.

When the received spaces $\langle m(u,d_i) : (u,d_i) \in \mathrm{In}(d_i)\rangle$ are the entire vector space Ω for all $d_i \in \mathbf{d}$, the sufficient condition for choosing redundant edges becomes a necessary condition.

Hence, the output of the generalized CF algorithm must be of locally minimal network usage as there exists no non-empty redundant edge set $E_R(v)$. ∎

References

[1] A. Barbero and O. Ytrehus, "Cycle-logical treatment for cyclopathic networks," *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2795–2804, June 2006.

[2] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *Proc. ACM Special Interest Group on Data Commun. (SIGCOMM).* Kyoto, Japan, August 2007.

[3] 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.

[4] T. Cui, L. Chen, and T. Ho, "Distributed optimization in wireless networks using broadcast advantage," in *Proc. 46th IEEE Conf. Decision and Contr.* New Orleans, December 2007.

[5] C. Fragouli, D. Lun, M. Médard, and P. Pakzad, "On feedback for network coding," in *Proc. 41st Conf. Inform. Sciences and Systems*, 2007.

[6] C. Fragouli and A. Markopoulou, "A network coding approach to network monitoring," in *Proc. 43rd Annual Allerton Conf. on Comm., Contr., and Computing.* Monticello, IL, USA, October 2005.

[7] C. Fragouli, A. Markopoulou, and S. Diggavi, "Topology inference using network coding," in *Proc. 44th Annual Allerton Conf. on Comm., Contr., and Computing.* Monticello, IL, USA, October 2006.

[8] C. Fragouli and E. Soljanin, "Information flow decomposition for network coding," *IEEE Trans. Inform. Theory*, vol. 52, no. 3, pp. 829–848, March 2006.

[9] M. Gjoka, C. Fragouli, P. Sattri, and A. Markopoulou, "Loss tomography in general topologies with network coding," in *Proc. 2007 Global Telecommunications Conference (GLOBECOM)*, November 2007, pp. 381–386.

[10] A. Goldberg, "Recent developments in maximum flow algorithms," NEC Research Institute, Inc., Technical Report 98-045, 1998.

[11] 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.

[12] T. Ho, B. Leong, Y.-H. Chang, Y. Wen, and R. Koetter, "Network monitoring in multicast networks using network coding," in *Proc. IEEE Int'l Symp. Inform. Theory.* Adelaide, Australia, September 2005, pp. 1977–1981.

[13] T. Ho, B. Leong, R. Koetter, and M. Médard, "Distributed asynchronous algorithms for multicast network coding," in *Proc. 1st Workshop on Network Coding, Theory, and Applications.* Riva del Garda, Italy, April 2005.

[14] 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.

[15] T. Ho and H. Viswanathan, "Dynamic algorithms for multicast with intra-session network coding," *IEEE Trans. Inform. Theory*, vol. 55, no. 2, pp. 797–815, February 2009.

[16] M. Jafarisiavoshani, C. Fragouli, and S. Diggavi, "Subspace properties of randomized network coding," in *Proc. IEEE Inform. Theory Workshop*, July 2007, pp. 1–5.

[17] S. Jaggi, P. Sanders, P. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen., "Polynomial time algorithms for multicast network code construction," *IEEE Trans. Inform. Theory*, vol. 51, 2005.

[18] 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.

[19] R. Koetter and F. Kschischang, "Coding for errors and erasures in random network coding," *IEEE Trans. Inform. Theory*, vol. 54, no. 8, pp. 3579–3591, August 2008.

[20] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, pp. 782–795, October 2003.

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

[22] Z. Li and B. Li, "Efficient and distributed computation of maximum multicast rates," in *Proc. 24th IEEE Conference on Computer Communications (INFOCOM).* Miami, Florida, USA, March 2005.

[23] Z. Li, B. Li, D. Jiang, and L. Lau, "On achieving optimal throughput with network coding," in *Proc. 24th IEEE Conference on Computer Communications (INFOCOM).* Miami, Florida, USA, March 2005.

[24] X. Lin and N. Shroff, "Utility maximization for communication networks with multi-path routing," *IEEE Trans. Automat. Contr.*, vol. 51, no. 5, pp. 766–781, May 2006.

[25] X. Lin, N. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE J. Selected Areas of Commun.*, vol. 24, no. 8, pp. 1452–1463, August 2006.

[26] H.-F. Lu, "Binary linear network codes," in *Proc. 2007 IEEE Information Theory Workshop on Information Theory for Wireless Networks*, July 2007, pp. 1–5.

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

[28] B. Radunović, C. Gkantsidis, P. Key, and P. Rodriguez, "An optimization framework for opportunistic multipath routing in wireless mesh networks," in *Proc. 27th IEEE Conference on Computer Communications (INFOCOM).* Pheonix, USA, 2008.

[29] M. Riemensberger, Y. Sagduyu, M. Honig, and W. Utschick, "Training overhead for decoding random linear network codes in wireless networks," *IEEE J. Selected Areas of Commun.*, vol. 27, no. 5, pp. 729–737, June 2009.

[30] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Automat. Contr.*, vol. 37, no. 12, pp. 1936–1948, December 1992.

[31] C.-C. Wang and X. Lin, "Fast resource allocation for network-coded traffic — a coded-feedback approach," in *Proc. 28th IEEE Conference on Computer Communications (INFOCOM).* Rio de Janeiro, Brazil, April 2009, mini conference.

[32] C.-C. Wang and N. Shroff, "Beyond the butterfly — a graph-theoretic characterization of the feasibility of network coding with two simple unicast sessions," in *Proc. IEEE Int'l Symp. Inform. Theory.* Nice, France, June 2007.

[33] 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.

[34] Y. Wu, P. Chou, and S.-Y. Kung, "Minimum-energy multicast in mobile ad hoc networks using network coding," *IEEE Trans. Commun.*, vol. 53, no. 11, pp. 1906–1918, November 2005.

[35] Y. Wu, P. Chou, Q. Zhang, K. Jain, W. Zhu, and S.-Y. Kung, "Network planning in wireless ad hoc networks: a cross-layer approach," *IEEE J. Selected Areas of Commun.*, vol. 23, no. 1, pp. 136–150, January 2005.

[36] Y. Wu and S.-Y. Kung, "Distributed utility maximization for network coding based multicasting: A shortest path approach," *IEEE J. Selected Areas of Commun.*, vol. 24, pp. 1475–1488, August 2006.

[37] Y. Xi and E. Yeh, "Distributed algorithms for minimum cost multicast with network coding in wireless networks," in *Proc. 4th Int'l Symp. Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, April 2006.

[38] R. Yeung and N. Cai, "Network error correction, part I: Basic concepts and upper bounds," *COMMUNICATIONS IN INFORMATION AND SYSTEMS*, vol. 6, no. 1, pp. 19–36, 2006.

PLACE
PHOTO
HERE

Chih-Chun Wang (M'06) joined the School of Electrical and Computer Engineering in January 2006 as an Assistant Professor. He received the B.E. degree in E.E. from National Taiwan University, Taipei, Taiwan in 1999, the M.S. degree in E.E., the Ph.D. degree in E.E. from Princeton University in 2002 and 2005, respectively. He worked in Comtrend Corporation, Taipei, Taiwan, as a design engineer during in 2000 and spent the summer of 2004 with Flarion Technologies, New Jersey. In 2005, he held a post-doc position in the Electrical Engineering Department of Princeton University. He received the National Science Foundation Faculty Early Career Development (CAREER) Award in 2009. His current research interests are in the graph-theoretic and algorithmic analysis of iterative decoding and of network coding. Other research interests of his fall in the general areas of optimal control, information theory, detection theory, coding theory, iterative decoding algorithms, and network coding.