Distributed Utility Maximization for Network Coding Based Multicasting: A Critical Cut Approach

Yunnan Wu, *Student Member, IEEE,* Mung Chiang, *Member, IEEE,* Sun-Yuan Kung, *Fellow, IEEE.*

***Abstract*— Recent advances in network coding have greatly facilitated information multicasting in communication networks. One central issue in practically deploying network coding in a shared network is the adaptive and economic allocation of network resources. Such an issue can be formulated as a globally coupled optimization problem, where the *net-utility* – the difference between a utility derived from the attainable multicast throughput and the total cost of resource provisioning – is maximized. This formulation presents new challenges due to the unique characterization of the multicast throughput attainable via network coding.**

We develop a primal-subgradient type distributed algorithm for solving this utility maximization problem. The effectiveness of the algorithm hinges upon two key properties we discovered: (1) the set of subgradients of the multicast throughput is the convex hull of the indicator vectors for the *critical cuts*, and (2) the complexity of finding such critical cuts can be substantially reduced by exploiting the algebraic properties of linear network coding. Extensions to multiple multicast sessions and rate-constrained utility maximization are also carried out. The effectiveness of the proposed algorithm is confirmed by simulations on an ISP (Internet Service Provider) topology.

***Index Terms*— Multicast, Network coding, Max-flow min-cut, Network utility maximization, Mathematical programming/optimization.**

I. Introduction and Problem Formulation

Consider a network formed by a collection of lossless links, which can naturally be represented by a directed graph $G = (V, E)$, where the vertex set V and the edge set E denote the nodes and links, respectively.

A. Network coding based multicasting

Information multicasting in a network involves a source node s that transmits common information to a set of destination nodes T. Let the bit-rate constraints on the links are specified by a length-$|E|$ vector $\boldsymbol{c}$; the capacity for link $vw \in E$ is denoted by c_{vw}. Given V, E, $\boldsymbol{c}$, s, and T, the *multicast capacity* refers to the maximum multicast throughput.

An upper bound of the multicast capacity can be established by examining the *cuts* that separate s from any destination $t \in T$. For $t \in T$, an s-t-cut $(U, \overline{U})$ refers to a partition of the nodes $V = U + \overline{U}$ with $s \in U$, $t \in \overline{U}$. The *capacity* of the cut refers to the sum of the edge capacities for edges going from U to $\overline{U}$; see Fig. 1. An s-t-cut with minimum capacity is called a *minimum s-t-cut*. Let $\rho_t(\boldsymbol{c})$ denote the capacity of a minimum s-t-cut for graph (V, E) with link capacities $\boldsymbol{c}$. Then

$$\min_{t \in T} \rho_t(\boldsymbol{c}) \tag{1}$$

is an upper bound of the multicast capacity since the capacity of any s-t-cut is an upper bound on the rate at which information can be transmitted from s to t.

In today's practical networks such as the Internet, end-to-end information delivery is done by *routing*, i.e., having intermediate nodes store and forward packets. For multicasting, Ahlswede *et al.* showed in [1] that the upper bound (1) cannot be achieved by routing in general, but it *can* be achieved, via more powerful techniques called *network coding*. Hence, (1) is the multicast capacity. Network coding generalizes routing by allowing a node to mix information, i.e., produce output data by computing certain functions of the data it received. Ahlswede *et al.*'s result was established via information theoretic arguments. Shortly afterwards, Li, Yeung, and Cai [2] showed that the multicast capacity can be achieved by linear network coding, i.e., using linear encoding functions at each node. Following this and other constructive theoretical results about network coding, in [3] a prototype system for practical network coding in real packet networks, using distributed random linear network coding with buffering, was presented. The system achieves throughput close to the capacity with low delay, and is robust to network dynamics.

B. Net-utility maximization formulation

An essential element needed in the practical network coding system is a distributed scheme for "properly" allocating bit-rate resources at each link for each multicast session in a shared network. Generally speaking, from a system perspective, there are two interacting considerations. On one hand, it is desirable to maximize the utilities of end users derived from the supported end-to-end multicast throughput. On the other hand, there is incentive to economize the consumption of network resources. Following a popular approach in economics theory, we can cast this problem as the maximization of a *net-utility* function

$$U(r) - \sum_{vw \in E} p_{vw}(g_{vw}). \tag{2}$$

Here $U(r)$ represents the raw utility when an end-to-end throughput r is provided. The cost function p_{vw} associated with link vw maps the consumed bit-rate g_{vw} to the charge.

The authors are with the Dept. of Electrical Engineering, Princeton University, Princeton, NJ 08544. {yunnanwu, chiangm, kung}@princeton.edu.

As a standard assumption, p_{vw} are nondecreasing and convex functions, and U is a nondecreasing and concave function.

The critical constraint of such a maximization is that throughput r must be attainable using the resources g_{vw}. Let $\boldsymbol{g}$ be a length-$|E|$ vector collectively representing g_{vw}. For network coding based multicasting, this relation is characterized by

$$r \leq R(\boldsymbol{g}) \equiv \min_{t \in T} \rho_t(\boldsymbol{g}). \tag{3}$$

Since by assumption $U(r)$ is nondecreasing, given $\boldsymbol{g}$, we can always set $r = R(\boldsymbol{g})$. With this observation, we can turn the problem into a maximization over $\boldsymbol{g}$ only. Let

$$U_{\text{net}}(\boldsymbol{g}) \equiv U(R(\boldsymbol{g})) - \sum_{vw \in E} p_{vw}(g_{vw}). \tag{4}$$

We consider the following optimization formulation[1]

$$\begin{aligned} U^*_{\text{net}} \equiv \max \quad & U_{\text{net}}(\boldsymbol{g}) \\ \text{subject to:} \quad & \mathbf{0} \leq \boldsymbol{g} \leq \boldsymbol{c}, \end{aligned} \tag{5}$$

The objective of this paper is to design efficient distributed algorithms for finding an optimal solution $\boldsymbol{g}^*$ of (5), which will be used as the allocated bit-rate resources at each link. The distributed algorithms should, hopefully, incur low extra communication overhead and be adaptive to network dynamics.

C. Proposed approach

We will show that $R(\boldsymbol{g})$ is concave and thus $U_{\text{net}}(\boldsymbol{g})$ is concave. Our formulation of the problem, (5), motivates us to apply an iterative algorithm that starts with an initial assignment $\boldsymbol{g}$ and incrementally updates it along certain directions. However, one of the difficulties is that the objective function is non-differentiable, due to the non-differentiability of the function $R(\boldsymbol{g})$. To cope with this issue, we resort to *subgradient methods*. A subgradient is a generalization of the gradient to non-differentiable functions; for a concave function f, each subgradient at $\boldsymbol{x}$ corresponds to a linear over-estimator of f that touches f at $\boldsymbol{x}$.

The subgradient method has been applied to various formulations of network utility maximization, e.g., in [4]–[10]. In addition, previous work by Lun *et al.* [11] proposed a subgradient-based distributed algorithm for minimum cost multicasting using network coding. However, in all these previous works, the subgradient method is applied to the Lagrangian dual problem. In contrast, our approach in this paper represents a primal subgradient method. The unique challenges here lie in deriving an analytic expression of a subgradient and developing a low-complexity implementation. As shown in sections II and III, respectively, both challenges can be met by using peculiar properties of network coding and following a primal, rather than dual-based, approach. The resulting distributed algorithm is presented in Section IV, followed by several extensions in Section V and simulation results in Section VI.

[1]In this paper, $\boldsymbol{a} \leq \boldsymbol{b}$ is in the element-wise sense.

When particularizing the subgradient methods to (5), we need to find a subgradient for the concave function $R(\boldsymbol{g})$. In Sections II.B and II.C, we provide a characteriztion of the set of subgradients of $R(\boldsymbol{g})$. We show the set of subgradients of $R(\boldsymbol{g})$ is the set of convex combinations of the indicator vectors for the s-T *critical cuts*. A cut $(U^*, \overline{U^*})$ is said to be s-T critical if it is a minimum s-t^*-cut for a *critical destination* t^*; a destination t^* is said to be critical if $\rho_{t^*}(\boldsymbol{g}) = \min_{t \in T} \rho_t(\boldsymbol{g})$.

Thus we can implement the subgradient iterations by seeking an s-T critical cut in each iteration. Finding a minimum s-t-cut is a classical combinatorial optimization problem that has been well understood. In particular, the preflow-push algorithm [12] for finding a minimum s-t-cut is suitable for distributed implementation. We can certaintly find an s-T critical cut by finding a minimum s-t-cut for each destination $t \in T$. However, as shown in Sections III.B and III.C, the algebraic properties of linear network coding are helpful in reducing that complexity. Previous works have shown the multicast capacity in an acyclic graph with unit edge capacities can be achieved with high probability by performing random linear coding over a sufficiently large finite field. Accordingly, if random mixing is done in a linear space with a dimension h_0 slightly less than the capacity C, then the destinations can recover the data with high probability. We observe that if mixing is done in a linear space with a dimension h slightly higher than the capacity C, then the critical destinations will have lower ranks than the noncritical destinations. To ensure that the destinations can still recover the data, we can linearly precode the original data, e.g., by letting some original data be zero. In essence, by perform random mixing in a space with dimention $h > C$ while using a signal space with dimension $h_0 < C$, the critical destinations are identified and the data recovery is feasible. As a result, we can find an s-T critical cut by first finding a critical destination t^* and then finding a minimum s-t^*-cut.

As discussed in Section V, the proposed distributed algorithm for network coding based utility maximization can be readily extended to the scenario where multiple multicast destinations are (additively) sharing the available resources $\boldsymbol{c}$, implementation is asynchronous, and lower bound on multicast throughput is provisioned.

The proposed algorithm has been tested on a large scale problem where the graph is the backbone topology of a commercial ISP (Internet Service Provider). The simulation results confirm the the convergence of the proposed algorithm.

II. Optimization via Subgradient Iterations

In this section, we present an iterative subgradient algorithm for solving (5). First we review some preliminaries on subgradient methods. Next we examine the function $R(\boldsymbol{g})$ and characterize the set of all subgradients of $R(\boldsymbol{g})$. The subgradient iterations is then presented in Section II-C.

A. Review: preliminaries on subgradient methods

Let the *domain* of a function f be denoted by

$$\text{dom } f \equiv \{\boldsymbol{x} \in \mathbb{R}^n : \ |f(\boldsymbol{x})| < \infty\}. \tag{6}$$

Definition 1 (Subgradient, subdifferential):
Given a convex function f, a vector $\boldsymbol{\xi}$ is said to be a *subgradient* of f at $\boldsymbol{x} \in \text{dom } f$ if

$$f(\boldsymbol{x}') \geq f(\boldsymbol{x}) + \boldsymbol{\xi}^T(\boldsymbol{x}' - \boldsymbol{x}), \quad \forall \boldsymbol{x}' \in \text{dom } f. \tag{7}$$

For a concave function f, a vector $\boldsymbol{\xi}$ is said to be a subgradient of f at $\boldsymbol{x}$ if $-\boldsymbol{\xi}$ is a subgradient of $-f$.

The set of all subgradients of f at $\boldsymbol{x}$ is called the *subdifferential* of f at $\boldsymbol{x}$ and denoted by $\partial f(\boldsymbol{x})$.

Lemma 1 (Subgradient calculus):
For a concave function $f: \mathbb{R}^+ \mapsto \mathbb{R}$, the following properties hold.

- f is differentiable at $\boldsymbol{x}$ if and only if $\partial f(\boldsymbol{x}) = \{\nabla f(\boldsymbol{x})\}$.
- $\partial(\alpha f) = \alpha \partial f$, if $\alpha > 0$.
- $\partial(f_1 + f_2) = \{\boldsymbol{\xi}_1 + \boldsymbol{\xi}_2 | \boldsymbol{\xi}_1 \in \partial f_1,\ \boldsymbol{\xi}_2 \in \partial f_2\}$
- **pointwise minimum**: If $f = \min_{i=1,\ldots,m} f_i$ where f_i, $i = 1, \ldots, m$ are concave functions, then

$$\partial f(\boldsymbol{x}) = \text{conv}\left\{\partial f_i(\boldsymbol{x}) | i \in I(\boldsymbol{x})\right\}, \tag{8}$$

where $I(\boldsymbol{x}) \equiv \{i | f_i(\boldsymbol{x}) = f(\boldsymbol{x})\}$. In other words, the subdifferential of f at $\boldsymbol{x}$ is the convex hull of the subdifferentials of the "active" functions at $\boldsymbol{x}$.
In particular, if $f_i(\boldsymbol{x}) = \boldsymbol{a}_i^T \boldsymbol{x} + \boldsymbol{b}_i, i = 1, \ldots, m$, then

$$\partial f(\boldsymbol{x}) = \text{conv}\left\{\boldsymbol{a}_i | f_i(\boldsymbol{x}) = f(\boldsymbol{x})\right\}. \tag{9}$$

■

The subgradient method [13] maximizes a non-differentiable concave function in a way similar to gradient methods for differentiable functions – in each step, the variables are updated in the direction of a subgradient. However, such a direction may not be an ascent direction; instead, the subgradient method relies on a different property. If the variable moves a sufficiently small step along the direction of a subgradient, then the new point is closer to any optimal solution.

Consider a generic, constrained concave maximization problem

$$\begin{aligned} \text{maximize} \quad & f(\boldsymbol{x}) \\ \text{subject to:} \quad & \boldsymbol{x} \in \mathcal{C}, \end{aligned} \tag{10}$$

where $f: \mathbb{R}^n \mapsto \mathbb{R}$ is concave, and $\mathcal{C}$ is a closed and nonempty convex set. The subgradient method uses the iteration

$$\boldsymbol{x}^{(k+1)} = P\left[\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{\xi}^{(k)}\right], \tag{11}$$

where $\boldsymbol{x}^{(k)}$ is the k-th iterate, $\boldsymbol{\xi}^{(k)}$ is any subgradient of f at $\boldsymbol{x}^{(k)}$, $\alpha_k > 0$ is the k-th step size, and P is the projection on $\mathcal{C}$:

$$P[\boldsymbol{x}] \equiv \arg\min_{\boldsymbol{x}' \in \mathcal{C}} \|\boldsymbol{x}' - \boldsymbol{x}\|^2. \tag{12}$$

Lemma 2 (Convergence of subgradient methods [14] [15]):
Assume $\boldsymbol{x}^*$ is a maximizer of (10) and there exists a G such that $\|\boldsymbol{\xi}^k\| \leq G$, $\forall k$. Then

$$f^* - \max_{i=1,\ldots,k} f(\boldsymbol{x}^{(i)}) \leq \frac{\|\boldsymbol{x}^{(1)} - \boldsymbol{x}^*\| + G^2 \sum_{i=1}^k \alpha_i^2}{2\sum_{i=1}^k \alpha_i}. \tag{13}$$

In particular,

- if constant step size is used, i.e., $\alpha_k = h$, then the right hand side of (13) converges to $G^2 h/2$ as $k \to \infty$.
- if the step sizes satisfy

$$\lim_{k\to\infty} \alpha_k = 0, \quad \sum_{k=1}^{\infty} \alpha_k = \infty, \tag{14}$$

then right hand side of (13) converges to 0 as $k \to \infty$. Step sizes that satisfy this condition are called *diminishing step size rules*.

B. Subgradients of $R(\boldsymbol{g})$

As mentioned in the introduction, a partition of the vertex set, $V = U + \overline{U}$ with $s \in U$, $t \in \overline{U}$, determines an s-t-cut. Define

$$\delta(U) \equiv \{vw \in E | v \in U, w \in \overline{U}\}. \tag{15}$$

Then $\rho_t(\boldsymbol{g})$ is the minimum capacity of an s-t-cut in graph (V, E) with link capacities $\boldsymbol{g}$, i.e.,[2]

$$\rho_t(\boldsymbol{g}) \equiv \min_{U:\ s\in U,\ t\in\overline{U}} \sum_{vw\in\delta(U)} g_{vw}. \tag{16}$$

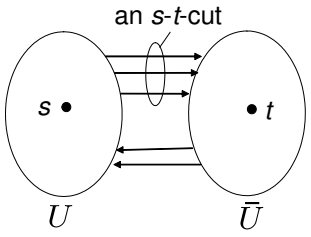

Fig. 1. Illustration of an s-t-cut $(U, \overline{U})$. A partition of the vertex set, $V = U + \overline{U}$ with $s \in U$, $t \in \overline{U}$, determines an s-t-cut. The capacity of the cut refers to the sum capacity of the links going from U to $\overline{U}$. The significance of an s-t-cut comes from the fact that it exhibits a bottleneck for communication from s to t.

An s-t-cut $(U^*, \overline{U^*})$ with minimum capacity is called a *minimum s-t-cut*. For a minimum s-t-cut $(U^*, \overline{U^*})$, the edges $\delta(U^*)$ can be interpreted as "critical" to the end-to-end capacity $\rho_t(\boldsymbol{g})$. If we decrease g_{vw} by any $\epsilon > 0$ for $vw \in \delta(U^*)$, then $\rho_t(\boldsymbol{g})$ will be reduced by ϵ. However, if we increase the capacity of some edges in $\delta(U^*)$, then it is possible that $\rho_t(\boldsymbol{g})$ will remain the same since there may exist other critical places. This asymmetry is a result of the piecewise linear nature of the function (16). As we shall show, a minimum s-t-cut $(U^*, \overline{U^*})$ determines a subgradient of $\rho_t(\boldsymbol{g})$. Indeed, from the pointwise minimum rule of subgradients and the definition of $\rho_t(\boldsymbol{g})$ (16), the subdifferential of $\rho_t(\boldsymbol{g})$ can be characterized as follows.

Let a length-$|E|$ binary vector $\boldsymbol{I}_X$ be the *indicator vector* for edge set $X \subseteq E$; its e-th entry is 1 if $e \in X$, and 0 if $e \notin X$.

Proposition 1 (Subgradients of min-cut function):
The subdifferential of $\rho_t(\boldsymbol{g})$ at $\boldsymbol{g}$ is

$$\partial\rho = \text{conv}\left\{\boldsymbol{I}_{\delta(U^*)} | (U^*, \overline{U^*}) \text{ is a minimum } s\text{-}t\text{-cut in } (V, E, \boldsymbol{g})\right\} \tag{17}$$

[2]In network communication applications, only nonnegative link capacities are physically meaningful. Nonetheless, the definition (16) allows us to treat the domain of $\rho_t(\boldsymbol{g})$ as $\mathbb{R}^{|E|}$; this turns out to be convenient as we discuss subgradients of $\rho_t(\boldsymbol{g})$.

■

Definition 2 (Critical destination, Critical cut): A destination $t^* \in T$ is said to be a *critical destination* if $\rho_{t^*}(\boldsymbol{g}) = \min_{t\in T} \rho_t(\boldsymbol{g})$. The set of critical destinations is

$$T^*(\boldsymbol{g}) \equiv \left\{ t^* \in T \,\middle|\, \rho_{t^*}(\boldsymbol{g}) = \min_{t\in T} \rho_t(\boldsymbol{g}) \right\}. \tag{18}$$

A cut $(U^*, \overline{U^*})$ is said to be an s-T *critical cut* if it is a minimum s-t^*-cut for some critical destination $t^* \in T^*(\boldsymbol{g})$.

The name "s-T critical" comes from the observation that reducing the capacity of the cut $(U^*, \overline{U^*})$ in Definition 2 by any positive amount reduces the multicast throughput $R(\boldsymbol{g})$ from s to T by the same amount.

Applying the pointwise minimum rule of subgradient calculus to (3), we can characterize the subgradients of $R(\boldsymbol{g})$ in Proposition 2.

Proposition 2 (Subgradients of multicast capacity): The subdifferential of $R(\boldsymbol{g})$ at $\boldsymbol{g}$ is

$$\partial R = \text{conv}\left\{ \boldsymbol{I}_{\delta(U^*)} | (U^*, \overline{U^*}) \text{ is an } s\text{-}T \text{ critical cut in } (V, E, \boldsymbol{g}) \right\} \tag{19}$$

C. Subgradient iterations

Let $\mathbb{R}^+$ denote the set of nonnegative real numbers. We assume that $p_{vw} : \mathbb{R}^+ \mapsto \mathbb{R}$ is nondecreasing and convex and $U : \mathbb{R}^+ \mapsto \mathbb{R}$ is nondecreasing and concave. As a result of these assumptions, we have the following

Lemma 3: The objective function of (5)

$$U\left(R(\boldsymbol{g})\right) - \sum_{vw\in E} p_{vw}(g_{vw}) \tag{20}$$

is concave in $\boldsymbol{g}$.

Proof: The function $\rho_t(\boldsymbol{g})$ is concave, since according to the definition (16), for $\lambda_1, \lambda_2 \geq 0$

$$\begin{aligned} &\rho_t(\lambda_1 \boldsymbol{g}_1 + \lambda_2 \boldsymbol{g}_2) && (21)\\ \geq &\rho_t(\lambda_1 \boldsymbol{g}_1) + \rho_t(\lambda_2 \boldsymbol{g}_2) && (22)\\ = &\lambda_1 \rho_t(\boldsymbol{g}_1) + \lambda_2 \rho_t(\boldsymbol{g}_2). && (23) \end{aligned}$$

The pointwise minimum of a family of concave function is also concave. Thus the lemma follows. ■

We now look for a subgradient of (20). Let $\dot{U}(x)$ denote a subgradient of $U(x)$ at x. Let $\dot{\boldsymbol{p}}(\boldsymbol{g})$ denote a subgradient of $\sum_{vw\in E} p_{vw}(g_{vw})$ at point $\boldsymbol{g}$. This vector can be obtained by finding a subgradient for each scalar function $p_{vw}(g_{vw})$.

Proposition 3 (A Subgradient of Objective Function): A subgradient of (20) at any $\boldsymbol{g} \geq \boldsymbol{0}$ is given as follows:

$$\boldsymbol{\xi} \equiv \dot{U}\left(R(\boldsymbol{g})\right) \boldsymbol{I}_{\delta(U^*)} - \dot{\boldsymbol{p}}(\boldsymbol{g}), \tag{24}$$

where $(U^*, \overline{U^*})$ is an s-T critical cut for $(V, E, \boldsymbol{g})$. This leads to the following subgradient updating rule:

$$\boldsymbol{g}^{(k+1)} = P\left[\boldsymbol{g}^{(k)} + \alpha_k \boldsymbol{\xi}^{(k)}\right]. \tag{25}$$

where $\boldsymbol{\xi}^{(k)}$ is a subgradient of (20) at the current solution $\boldsymbol{g}^{(k)}$ formed according to (24).

Proof: We just need to show that $\dot{U}\left(R(\boldsymbol{g})\right) \boldsymbol{I}_{\delta(U^*)}$ is a subgradient of $U\left(R(\boldsymbol{g})\right)$. The following proof essentially verifies that a subgradient chain rule holds.

From the definition of subgradients

$$U(x') - U(x) \leq \dot{U}(x)(x' - x), \quad \forall x', x \geq 0 \tag{26}$$

$$R(\boldsymbol{g}') - R(\boldsymbol{g}) \leq \boldsymbol{I}_{\delta(U^*)}(\boldsymbol{g}' - \boldsymbol{g}), \quad \forall \boldsymbol{g}', \boldsymbol{g} \geq \boldsymbol{0}. \tag{27}$$

Since U is nondecreasing, we have $\dot{U}(x) \geq 0$. Substitute x and x' in (26) with $x = R(\boldsymbol{g})$ and $x' = R(\boldsymbol{g}')$, respectively. Then

$$\begin{aligned} &U\left(R(\boldsymbol{g}')\right) - U\left(R(\boldsymbol{g})\right) && (28)\\ \leq &\dot{U}\left(R(\boldsymbol{g})\right)\left(R(\boldsymbol{g}') - R(\boldsymbol{g})\right) && (29)\\ \leq &\dot{U}\left(R(\boldsymbol{g})\right) \boldsymbol{I}_{\delta(U^*)}(\boldsymbol{g}' - \boldsymbol{g}), \quad \forall \boldsymbol{g}', \boldsymbol{g} \geq \boldsymbol{0}. && (30) \end{aligned}$$

Now the above chain rule together with Propositions 1 and 2 proves Proposition 3. ■

Note that in (25), the projection is onto the Cartesian set $\{\boldsymbol{g} | \boldsymbol{0} \leq \boldsymbol{g} \leq \boldsymbol{c}\}$ and thus it decouples into finding $\min\{\max\{0, g_{vw}\}\}$ for each entry g_{vw}.

To gain some intuition with the subgradient iterations above, it might be helpful to consider how the procedure applies to the degenerate case where the links are costless and $U(r) = r$. In this case, the problem is just to find the multicast capacity $R(\boldsymbol{c})$, which can be done straightforwardly by computing $\rho_t(\boldsymbol{g})$ for each $t \in T$. We now see how an initial solution $\boldsymbol{g}^{(1)} = \boldsymbol{0}$ "grows" to some vector with close to maximum throughput in the subgradient method. In this case, in each iteration we increase $\boldsymbol{g}^{(k)}$ along a certain s-T critical cut. This does not necessarily increase $R(\boldsymbol{g})$ since there could be multiple s-T critical cuts. However, as we allocate more resources to one critical cut at a time, gradually all of them are allocated more resources and then the end-to-end throughput will increase. In the general case with nonzero costs and arbitary utility U, the balance between the resource cost and the throughput comes into play in (24).

III. Finding an s-T critical cut efficiently

After developing the structure of the desired distributed algorithm and deriving a subgradient to the objective function, we turn to the issue of finding an $s-T$ critical cut distributively and efficiently.

We begin by discussing in subsection III-A how to identify a minimum s-t-cut in graph (V, E) with edge capacities $\boldsymbol{g}$ for a given t. We review a specific minimum s-t-cut algorithm, the preflow-push algorithm [12], [16], whose nature is suitable for distributed implementation.

It is certainly possible to find a minimum s-t^*-cut for some $t^* \in T^*$ by running the minimum s-t-cut algorithm for all destinations. However, it is possible to fulfill this task with a lower complexity, by exploiting some algebraic properties of (random) linear network coding. In order to explain this, we first review linear network coding for acyclic graphs with unit capacity edges [2] in Section III-B.

In Section III-C, we present a way of identifying the critical destinations. The basic observation is as follows. Previous

works have shown that random linear coding approaches the multicast capacity C. Specifically, by performing random linear coding over a C-dimensional space, the *rank of* each destination (defined in Section III-B) will be close to C. If we perform random linear coding over a space with a slightly larger dimension $h > C$, then the rank of each critical destination will still be close to C but the rank of a non-critical destination will be higher. This allows us to distinguish the critical destinations from the noncritical destinations. The incurred overhead in implementing this method is a few extra symbols in each packet.

A. Review: finding a minimum s-t-cut via Preflow-Push algorithm

Most minimum cut algorithms hinge upon the Max-Flow-Min-Cut Theorem, which states that for graph (V, E) with edge capacities $\boldsymbol{g}$, the minimum cut capacity $\rho_t(\boldsymbol{g})$ is equal to the maximum value of an s-t-*flow*. An s-t-*flow* is a length-$|E|$ nonnegative vector $\boldsymbol{f}$ satisfying the *flow conservation constraint*:

$$\text{excess}_v(\boldsymbol{f}) = 0, \quad \forall v \in V - \{s, t\}. \tag{31}$$

where

$$\text{excess}_v(\boldsymbol{f}) \equiv \sum_{u:\ uv \in E} f_{uv} - \sum_{w:\ vw \in E} f_{vw}, \tag{32}$$

is the *flow excess of* v, i.e., the amount of incoming traffic less the amount of outgoing traffic for node v. The flow excess is not required to be zero at s and t. An s-t-flow $\boldsymbol{f}$ essentially prescribes several parallel paths, along which information can be routed from s to t. The flow excess at t is called the *value of the flow*, which corresponds to the communication rate that can be achieved by routing along the paths associated with $\boldsymbol{f}$.

The Preflow-Push Algorithm [12] [16] makes use of a structure called *preflow*. A preflow is similar to a flow except that the amount flowing into a node may exceed the amount flowing out of the node; that is, a preflow $\boldsymbol{f}$ satisfies $\boldsymbol{0} \le \boldsymbol{f} \le \boldsymbol{g}$ and

$$\text{excess}_v(\boldsymbol{f}) \ge 0, \quad \forall v \in V - \{s\}. \tag{33}$$

We can define a *residual graph* $G_{\boldsymbol{f}}$ associated with a preflow $\boldsymbol{f}$. Let its vertex set be $V(G_{\boldsymbol{f}}) = V(G)$. Draw an edge vw in $G_{\boldsymbol{f}}$ if

$$c_{vw} - f_{vw} + f_{wv} > 0. \tag{34}$$

(if $wv \notin E$, then f_{wv} is treated as 0). Such an edge $vw \in E(G_{\boldsymbol{f}})$ indicates that v can "push" to w some additional traffic, by increasing the load on vw and/or decreasing the load on wv.

The algorithm works by pushing excess flows to nodes estimated to be closer to t. The distance estimation is done by maintaining a *distance label*. A distance label is a length-$|V|$ vector $\boldsymbol{d}$ satisfying

P1) $d_s = |V|$, $d_t = 0$, $d_v \in \mathbb{Z}^+$, $\forall v \in V$;
P2) $d_v \le d_w + 1$ for every edge $vw \in E(G_{\boldsymbol{f}})$.

The distance label $\boldsymbol{d}$ plays the role of an upper bound of the hop distance to t in the residual graph $G_{\boldsymbol{f}}$.

Specifically, the algorithm proceeds by applying two primitive operations, *Push*(v, w) and *Relabel*(v), at *active nodes*. A node v is active if $v \in V - \{s, t\}$, $d_v < |V|$, and $\text{excess}_v(\boldsymbol{f}) > 0$. The operation *Push*$(v, w)$ pushes traffic from an active node v to w when $vw \in G_{\boldsymbol{f}}$ and $d_v = d_w + 1$. When an active node v has possive excess but cannot push to any of its neighbors, its label d_v is increased while still satisfying P1) and P2).

The algorithm ends when, for each node $v \in V - \{s, t\}$ with $d_v < |V|$, $\text{excess}_v(\boldsymbol{f}) = 0$. At this time a backward breadth first search is used to find the set of nodes $\overline{U^*}$ that can reach t in $G_{\boldsymbol{f}}$. This identifies a minimum s-t-cut $(U^*, \overline{U^*})$.

Lemma 4 (Minimum cut via Preflow-Push Alg. [12]): Let $\boldsymbol{f}$ be an s-t-preflow and $\boldsymbol{d}$ be a distance labeling. Suppose for each node $v \in V - \{s, t\}$ with $d_v < |V|$, $\text{excess}_v(\boldsymbol{f}) = 0$. Let U^* be the set of nodes that cannot reach t in $G_{\boldsymbol{f}}$. Then $(U^*, \overline{U^*})$ is a minimum s-t-cut.

An asynchronous protocol for distribution implementation of the preflow-push algorithm is also proposed in [16]. The protocol implements the primitive operations *Push*(v, w) and *Relabel*(v) via local message exchanges.

B. Review: linear network coding for acyclic graphs with unit capacity edges

Before showing how practical linear network coding facilitate a low-complexity implementation of our distributed utility maximization algorithm, we first review the relevant recent results from [2] [17].

Consider information multicast from s to T in a directed acyclic graph $G = (V, E)$ with all edges of unit capacity. Assume each edge $e \in E$ can carry one symbol from a certain finite field $\mathbb{F}$. Let y_e denote the symbol carried by edge e. Let $x_1, \ldots, x_h$, denote the *source symbols* available at the source node s. For notational consistency, we introduce h *source edges*, $s_1, \ldots, s_h$, which all end in s; the source edges $s_1, \ldots, s_h$ carry the h source symbols $x_1, \ldots, x_h$, respectively. See Fig. 2 for an example.

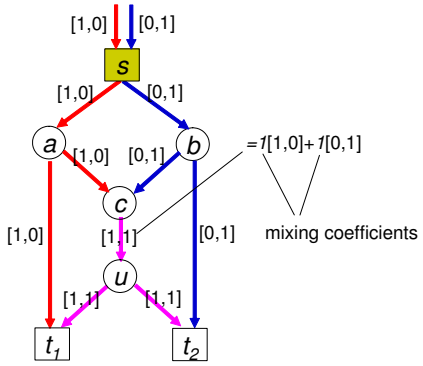

Fig. 2. Illustration of linear network coding. The global coding vectors are shown next to the edges.

In a linear network coding assignment, the symbol on edge e is a linear combination of the symbols on the edges entering

$\text{tail}(e)$[3], namely

$$y_e = \sum_{e':\ \text{head}(e')=\text{tail}(e)} w_{e,e'} y_{e'}. \tag{35}$$

We call the coefficients $\{w_{e,e'}\}$ the "mixing coefficients". By induction, y_e on any edge e is a linear combination of the source symbols, namely

$$y_e = \sum_{i=1}^{h} q_{e,i} x_i. \tag{36}$$

The vector $\boldsymbol{q}_e \equiv [q_{e,1}, \ldots, q_{e,h}]$ is known as the *global coding vector* at edge e. It can be determined recursively as

$$\boldsymbol{q}_e = \sum_{e':\ \text{head}(e')=\text{tail}(e)} w_{e,e'} \boldsymbol{q}_{e'}, \tag{37}$$

where $\boldsymbol{q}_{s_i}$ is the ith unit vector $\boldsymbol{\epsilon}_i$. Since each y_e is a linear combination of the source symbols, any destination t receiving h symbols with linearly independent global coding vectors can recover the source symbols. Therefore, an important metric is the rank of the span of global vectors for the incoming edges of each destination. The above discussions are summarized in the following definition. Fig. 2 shows a linear network coding assignment that allows the two destinations t_1 and t_2 to recover the two source symbols.

Definition 3 (Linear Network Coding Assignment): Given an acyclic graph $G = (V, E)$, a source node s, a finite field $\mathbb{F}$, and a code dimension h, a *linear network coding assignment* W refers to an assignment of mixing coefficients $w_{e,e'} \in \mathbb{F}$, one for each pair of edges (e, e') with $e \in E$, $e' \in E \cup \{s_1, \ldots, s_h\}$, and $\text{head}(e') = \text{tail}(e)$.

The *global coding vectors* resulting from a linear network coding assignment W, $\boldsymbol{q}_e(W)$, are the set of $|E|$ vectors $\boldsymbol{q}_e$ determined by W according to (37).

In a linear network coding assignment W, the *rank of a node* v, $\text{rank}_v(W)$, refers to the rank of the span of the global coding vectors for incoming edges of v, i.e.,

$$\text{rank}_v(W) \equiv \text{rank}\{\boldsymbol{q}_e(W),\ \text{head}(e) = v\}. \tag{38}$$

■

Lemma 5 is a straightforward extension of linear network coding result by Ahlswede *et al.* [1].

Lemma 5 (Mincut Upper Bound on Rank): Given an acyclic graph $G = (V, E)$, s, any linear network coding assignment W with dimension h must satisfy

$$\text{rank}_v(W) \le \min\{\rho_v, h\}, \quad \forall v \in V - \{s\}, \tag{39}$$

where ρ_v is the minimum capacity (cardinality) of an s-v-cut in G.

Proof: Let $(U, \overline{U})$ be a minimum s-v-cut. Since the global coding vector for any incoming edge of v must be spanned by the global coding vectors for $\delta(U)$, we have

$$\text{rank}_v(W) \le \text{rank}\{\boldsymbol{q}_e(W),\ e \in \delta(U)\} = \rho_v. \tag{40}$$

The relation $\text{rank}_v(W) \le h$ is trivial since $\boldsymbol{q}_e$ is a vector of length h. ■

[3]An edge e from v to w is said to have $\text{tail}(e) = v$ and $\text{head}(e) = w$.

Now consider multicasting information from s to T using linear network coding. Note from Lemma 5 that

$$\min_{t \in T} \text{rank}_t(W) \le \min\{\min_{t \in T} \rho_t, h\}. \tag{41}$$

Therefore the number of distinct information symbols that can be multicast to T is at most

$$C \equiv \min_{t \in T} \rho_t. \tag{42}$$

Previous works have shown that random linear coding can achieve the rate C with high probability, for a sufficiently large field size; see, e.g., Ho *et al.* [17], Jaggi *et al.* [18]. The following particular result is from [17].

Lemma 6 (Optimality of Random Linear Coding [17]): Consider an acyclic graph $G = (V, E)$, a source node s, and a set of destinations T. For a code dimension $h \le C$ and a finite field size $|\mathbb{F}|$,

$$Pr\left[\text{rank}_t(W) = h, \forall t \in T\right] \ge \left(1 - \frac{|T|}{|\mathbb{F}|}\right)^{|E|}, \tag{43}$$

where W is a random vector with each mixing coefficient $w_{e,e'}$ chosen independently and uniformly from $\mathbb{F}$.

C. Algebraically identifying critical destinations

Previous works about random linear coding have focused on achieving the multicast capacity and hence considered $h \le C$. However, we will make use of a code dimension $h > C$ for the purpose of identifying the critical destinations. First, note the following easy corollary of Lemma 6.

Corollary 1: Consider an acyclic graph $G = (V, E)$ and a source node s. For any dimension $h > 0$ and finite field $\mathbb{F}$,

$$Pr\left[\text{rank}_v(W) = \min\{\rho_v, h\}\right] \ge \left(1 - \frac{1}{|\mathbb{F}|}\right)^{|E|}, \quad \forall v \in V. \tag{44}$$

where W is a random vector with each mixing coefficient $w_{e,e'}$ chosen independently and uniformly from $\mathbb{F}$. In particular, by using a large enough $\mathbb{F}$, the probability in (44) can be made arbitrarily small.

Proof: Consider each node $v \in V$. If $\rho_v \ge h$, then (44) is established upon applying Lemma 6 with $T = \{v\}$. If $\rho_v < h$, let $\boldsymbol{q}'_e$ be the subvector of $\boldsymbol{q}_e$ consisting of the first ρ_v entries. Applying Lemma 6 with $T = \{v\}$ and a code dimension ρ_v, we see that

$$Pr\left[\text{rank}\{\boldsymbol{q}'_e : \text{head}(e) = v\} = \rho_v, \right] \ge \left(1 - \frac{1}{|\mathbb{F}|}\right)^{|E|}.$$

Since $\text{rank}_v(W) \ge \text{rank}\{\boldsymbol{q}'_e : \text{head}(e) = v\}$, the result follows.

■

Recall that a destination t is said to be critical if $\rho_t = C$ and non-critical if $\rho_t > C$. Suppose we use random linear network coding with $h > C$. Then for a sufficiently large finite field $\mathbb{F}$, the rank of a critical user $t^* \in T^*$ will be close to C, whereas the rank of a non-critical user $t \in T - T^*$ will be close to $\min\{\rho_t, h\} > C$. This gives a method to identify the critical destinations.

There is a note on precoding worth mentioning here. With random linear coding over a sufficiently large finite field, a critical destination t^* will receive approximately C symbols with linearly independent global coding vectors. Each symbol corresponds to a linear equation in terms of the h unknowns, $x_1, \ldots, x_h$. If $h > C$, there are more unknowns than the equations. How would t^* be able to recover the source information? This issue can be solved by performing *precoding*; this technique was used in [3] for robustness in a dynamic network. If the source symbols $x_1, \ldots, x_h$ are linearly coded versions of $h_0 \leq C$ underlying variables, then it becomes possible to recover the underlying variables. The simplest form of precoding is to set $x_{h_0+1}, \ldots, x_h$ to zero. This is sufficient for our purpose. The parameter h_0 is called the *signal space dimension* and h is called the *mixing dimension*.

We now come to implications of the above results to the implementation of our distributed utility maximization algorithm in the practical network coding system. The proposed method of identifying the critical destinations can be implemented almost "for free" in the practical network coding system [3].

The practical network coding system is based on random linear network coding with buffering. The system uses an operating field of $GF(2^m)$ so that each symbol amounts to m bits. The source node produces a stream of source packets, each containing a fixed number of symbols (e.g., 1000 symbols in $GF(2^8)$). The source packets are grouped into multiple *generations*, each containing h packets. Random linear network coding is applied separately to different generations; in other words, only packets belonging to the same generation are mixed. The packet format in the practical network coding system [3] is shown in Fig. 3.

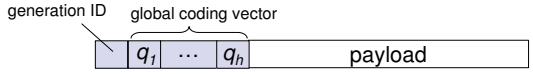

Fig. 3. The packet format in the practical network coding system [3].

Each node in the network maintains a buffer. Whenever a node receives a packet via one of its incoming links, it stores the packet into its buffer. Whenever there is a transmission opportunity available on one of its outgoing links, a node generates an output packet by linearly mixing the packets in the buffer with random coefficients in $GF(2^m)$. Each node v can perform Gaussian elimination and detect each incoming packet as *innovative* or *non-innovative* (to v). In other words, if a packet arriving at node v is spanned by the content in the buffer of v, then it is non-innovative and thus can be discarded.

To implement the proposed method of identifying the critical destinations, let the source packets in a generation be

$$\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{h_0}, \boldsymbol{x}_{h_0+1} = \boldsymbol{0}, \ldots, \boldsymbol{x}_h = \boldsymbol{0}. \tag{45}$$

Thus the payload of a packet with global coding vector $\boldsymbol{q} = [q_1, \ldots, q_h]$ will be

$$\sum_{i=1}^{h_0} q_i \boldsymbol{x}_i. \tag{46}$$

Since each payload does not involve $[q_{h_0+1}, \ldots, q_h]$, these entries of $\boldsymbol{q}$ are not used in the final decoding at each destination. These amount to the overhead incurred for testing criticality of destinations.

To leave a margin against temporary network outages, we should set h_0 such that the corresponding rate is slightly below the nominal multicast capacity. This assures that decoding will be successful at each destination with high probability. We should set h such that the corresponding rate is slightly above the multicast capacity. In this way, the critical destinations will exhibit lower ranks than the noncritical destinations.

IV. Summary of the Proposed Distributed Algorithm

We now summarize the proposed distributed algorithm. It is assumed that each node in the network can store data pertaining to its outgoing edges and perform computations on them.

For simplicity, let us assume that the algorithm is already in the steady phase. In other words, we assume that the practical network coding system is already up running. Thus we can use the rank information collected at the destinations for identifying the critical destinations.

Algorithm 1 (Primal subgradient method via critical cuts): Suppose a current solution $\boldsymbol{g}^{(k)}$ is just obtained. This vector is stored distributively in the network. Suppose further that the source s has a coarse estimate of the range of $R(\boldsymbol{g}^{(k)})$, e.g., via $R(\boldsymbol{g}^{(k-1)})$. Then the following steps are performed.

1) Set the practical network coding system with the parameters h_0 and h at the source such that the multicast rate associated with h_0 is slightly below $R(\boldsymbol{g}^{(k)})$ and the multicast rate associated with h is slightly above $R(\boldsymbol{g}^{(k)})$; see Section III-C for details. Each destination monitors the rank information for one or more generations of data multicasting and reports the information to s.
2) Using the reported rank information, the source s determines $R(\boldsymbol{g}^{(k)})$ (by taking the minimum of the rank of the destinations and dividing it by the time interval), as well as a worst destination t^* which has the worst (minimum) throughput. The source s then initiates the execution of the preflow-push algorithm, to find a minimum s-t^*-cut $(U^*, \overline{U^*})$. This represents an s-t^*-critical cut.
3) The source s conveys the value $\dot{U}(R(\boldsymbol{g}^{(k)}))$ to all nodes involved in $(U^*, \overline{U^*})$.
4) The subgradient update (25) is implemented in parallel. For each edge vw going from U^* to $\overline{U^*}$, set
$$g_{vw}^{(k+1)} = P_{vw}\left[g_{vw}^{(k)} + \alpha_k \left(\dot{U}(R(\boldsymbol{g}^{(k)})) - \dot{p}(g_{vw}^{(k)})\right)\right], \tag{47}$$
where $P_{vw}[x] = \min\{\max\{x, 0\}, c_{vw}\}$. For each other edge, set
$$g_{vw}^{(k+1)} = P_{vw}\left[g_{vw}^{(k)} - \alpha_k \dot{p}(g_{vw}^{(k)})\right]. \tag{48}$$

V. Extensions

To address the needs in practical systems, two useful extensions of the proposed utility maximization formulation and distributed algorithm are discussed in this section: multiple multicast sessions and lower bound on rate.

A. Multiple multicast sessions

Consider the scenario where there are multiple multicast sessions in the network. Label them with indices $m = 1, \ldots, M$. Denote the source and the destination set of the m-th session by s_m and T_m, respectively. Assume the m-th multicast session communicates using its exclusive share of resource $\boldsymbol{g}_m$. In this case the multicast rate for this session is

$$R_m(\boldsymbol{g}_m) \equiv \min_{t \in T_m} \rho_{m,t}(\boldsymbol{g}_m). \tag{49}$$

We consider the following optimization, where the variables are $[\boldsymbol{g}_1; \ldots; \boldsymbol{g}_M]$

$$\begin{aligned} \text{maximize} \quad & \sum_{m=1}^{M} U_m\left(R_m(\boldsymbol{g}_m)\right) - \sum_{vw \in E} p_{vw} \left(\sum_{m=1}^{M} g_{vw}^m \right) \\ \text{subject to:} \quad & \mathbf{0} \le \boldsymbol{g}_m, \quad \forall m \\ & \boldsymbol{g}_1 + \ldots + \boldsymbol{g}_M \le \boldsymbol{c} \end{aligned} \tag{50}$$

A subgradient of the objective function can be obtained as the stacked vector $[\boldsymbol{\xi}_1; \ldots; \boldsymbol{\xi}_M]$, where

$$\boldsymbol{\xi}_m \equiv \dot{U}_m\left(R_m(\boldsymbol{g}_m)\right) \boldsymbol{\xi}_{R_m}(\boldsymbol{g}_m) - \dot{\boldsymbol{p}} \left(\sum_{m=1}^{M} \boldsymbol{g}_m \right) \tag{51}$$

and

$$\boldsymbol{\xi}_{R_m}(\boldsymbol{g}_m) \in \partial R_m(\boldsymbol{g}_m). \tag{52}$$

Now the projection of the updated vector $\boldsymbol{g}^{(k+1)}$ to the feasible region of (50) becomes slightly more complicated. Since the constraints in (50) are decoupled for the edges, we need to solve the following problem for each edge $vw \in E$

$$\begin{aligned} P[\boldsymbol{g}_{vw}] \equiv \arg\min_{\boldsymbol{g}'_{vw}} \quad & \|\boldsymbol{g}'_{vw} - \boldsymbol{g}_{vw}\|^2 \\ \text{subject to:} \quad & \mathbf{0} \le \boldsymbol{g}'_{vw}, \\ & \mathbf{1}^T \boldsymbol{g}'_{vw} \le c_{vw}, \end{aligned} \tag{53}$$

where $\boldsymbol{g}_{vw}$ is a length-M vector consisting of the current value of g_{vw}^m, and $\mathbf{1}$ is a vector of all 1's.

There is an additional note worth mentioning. In a distributed environment, it is unlikely that all the sessions can be synchronized to finish computing a subgradient (s_m-T_m critical cut) $\boldsymbol{\xi}_{R_m}(\boldsymbol{g}_m)$ at the same time. So an approximation would be to perform the gradient update using the most recent value of $\boldsymbol{\xi}_{R_m}$. More specifically, suppose an s_m-t-cut $(U_0, \overline{U_0})$ is an s_m-T_m critical cut for the previous point $\boldsymbol{g}_{m0}$. Then, assuming the current point $\boldsymbol{g}_m$ is close to $\boldsymbol{g}_{m0}$, the capacity of this cut will be close to the current minimum $R_m(\boldsymbol{g}_m)$. Then, can we use $\boldsymbol{I}_{\delta(U_0)}$ as an approximate subgradient for the current point? In the following we discuss these more formally.

1) Approximate subgradients: In the literature, theoretical results have been presented regarding approximate subgradients; see, e.g., [15] and the references therein. For a concave function f, an ϵ-subgradient at $\boldsymbol{x}$ [15] is a vector $\boldsymbol{\zeta}$ such that

$$f(\boldsymbol{y}) \le f(\boldsymbol{x}) + \boldsymbol{\zeta}^T(\boldsymbol{y} - \boldsymbol{x}) + \epsilon \tag{54}$$

An approximate subgradient method for the generic maximization $\max_{\boldsymbol{x} \in \mathcal{C}} f(\boldsymbol{x})$ uses the iteration

$$\boldsymbol{x}^{(k+1)} = P\left[\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{\zeta}^{(k)}\right], \tag{55}$$

where $\boldsymbol{\zeta}^{(k)}$ is an $\epsilon^{(k)}$-subgradient at $\boldsymbol{x}^{(k)}$. The convergence results about approximate subgradient updates are referred to [15] [19].

In the following lemma, we present some perturbations properties of $R(\boldsymbol{g})$, which are useful in quantifying the effect of using an s-T critical cut evaluated at an earlier graph $\boldsymbol{g}_0$ to generate an approximate subgradient at the current point $\boldsymbol{g}$.

Lemma 7 (Perturbations of $R(\boldsymbol{g})$):
Given $(V, E, \boldsymbol{g}_0)$, let $(U_0, \overline{U_0})$ be an s-T critical cut. Let

$$\theta \equiv \|\boldsymbol{g}_0 - \boldsymbol{g}\|_1, \tag{56}$$

where $\|\cdot\|_1$ stands for the 1-norm of a vector (the sum of absolute values of the elements). Then

$$|R(\boldsymbol{g}_0) - R(\boldsymbol{g})| \le \theta \tag{57}$$

$$\boldsymbol{I}^T_{\delta(U_0)} \boldsymbol{g} - R(\boldsymbol{g}) \le 2\theta \tag{58}$$

Furthermore, $\boldsymbol{I}_{\delta(U_0)}$ is an 2θ-subgradient of R at $\boldsymbol{g}$, i.e.,

$$R(\boldsymbol{g}') - R(\boldsymbol{g}) \le \boldsymbol{I}^T_{\delta(U_0)}(\boldsymbol{g}' - \boldsymbol{g}) + 2\theta, \quad \forall \boldsymbol{g}' \ge \mathbf{0} \tag{59}$$

Remark: We interpret $\boldsymbol{g}_0$ as the old point and $\boldsymbol{g}$ as the current point. Property (57) bounds the possible change in R as $\boldsymbol{g}_0$ becomes $\boldsymbol{g}$. Property (58) quantifies the level of sub-optimality in using a previously found critical cut as an approximate critical cut for the current point.

Proof: The definition of $R(\boldsymbol{g})$ can be equivalently written as

$$\begin{aligned} R(\boldsymbol{g}) &= \min_{t \in T} \min_{U:\ s \in U,\ t \in \overline{U}} \boldsymbol{I}^T_{\delta(U)} \boldsymbol{g} && (60) \\ &= \min_{U:\ s \in U,\ \overline{U} \cap T \neq \emptyset} \boldsymbol{I}^T_{\delta(U)} \boldsymbol{g}. && (61) \end{aligned}$$

Thus

$$R(\boldsymbol{g}) \le \boldsymbol{I}^T_{\delta(U_0)} \boldsymbol{g}.$$

Since $R(\boldsymbol{g}_0) = \boldsymbol{I}^T_{\delta(U_0)} \boldsymbol{g}_0$, we have

$$R(\boldsymbol{g}) - R(\boldsymbol{g}_0) \le \boldsymbol{I}^T_{\delta(U_0)} \boldsymbol{g} - \boldsymbol{I}^T_{\delta(U_0)} \boldsymbol{g}_0 \le \theta.$$

Due to symmetry, we also have $R(\boldsymbol{g}_0) - R(\boldsymbol{g}) \le \theta$. This establishes (57).

$$\begin{aligned} & \boldsymbol{I}^T_{\delta(U_0)} \boldsymbol{g} - R(\boldsymbol{g}) \\ =& \boldsymbol{I}^T_{\delta(U_0)} \boldsymbol{g} - \boldsymbol{I}^T_{\delta(U_0)} \boldsymbol{g}_0 + \boldsymbol{I}^T_{\delta(U_0)} \boldsymbol{g}_0 - R(\boldsymbol{g}) \\ =& \left(\boldsymbol{I}^T_{\delta(U_0)} \boldsymbol{g} - \boldsymbol{I}^T_{\delta(U_0)} \boldsymbol{g} \right) + \left(R(\boldsymbol{g}_0) - R(\boldsymbol{g}) \right) \\ \le& \theta + \theta. \end{aligned}$$

This establishes (58).

To establish (59), consider the left hand side minus the right hand side.

$$\begin{aligned} & R(\boldsymbol{g}') - R(\boldsymbol{g}) - \boldsymbol{I}^T_{\delta(U_0)}(\boldsymbol{g}' - \boldsymbol{g}) - 2\theta \\ =& \left(R(\boldsymbol{g}') - \boldsymbol{I}^T_{\delta(U_0)} \boldsymbol{g}' \right) + \left(\boldsymbol{I}^T_{\delta(U_0)} \boldsymbol{g} - R(\boldsymbol{g}) \right) - 2\theta \\ \le& 0 + 2\theta - 2\theta = 0. \end{aligned}$$

■

B. Lower bound constraint on the rate

Consider the following generalization of (5)

$$\begin{aligned} \text{maximize} \quad & U_{\text{net}}(\boldsymbol{g}) \\ \text{subject to:} \quad & r_0 \le R(\boldsymbol{g}), \\ & \mathbf{0} \le \boldsymbol{g} \le \boldsymbol{c}. \end{aligned} \tag{62}$$

Here r_0 is a lower bound of the multicast rate that must be met. This formulation encompasses the special case where the multicast application wants a fixed rate r_0. Specifically, if we set $U(r) = 0$, then since by assumption p_{vw} are nondecreasing, it follows that one optimal solution will attain rate r_0.

The subgradient method has a variant for dealing with inequality constraints; see, e.g., [20] [21]. This can be applied here for solving (62). Specifically, define

$$\mathcal{Q} \equiv \{\boldsymbol{g} | r_0 \le R(\boldsymbol{g})\}. \tag{63}$$

Then the subgradient iteration

$$\boldsymbol{g}^{(k+1)} = P\left[\boldsymbol{g}^{(k)} + \alpha_k \boldsymbol{\xi}^{(k)}\right] \tag{64}$$

will be used with a different rule for selecting $\boldsymbol{\xi}^{(k)}$. If $\boldsymbol{g}^{(k)} \in \mathcal{Q}$, then $\boldsymbol{\xi}^{(k)}$ can be chosen as any subgradient of U_{net} at $\boldsymbol{g}^{(k)}$. If $\boldsymbol{g}^{(k)} \notin \mathcal{Q}$, then $\boldsymbol{\xi}^{(k)}$ can be chosen as any subgradient of $R(\boldsymbol{g})$ at $\boldsymbol{g}^{(k)}$. In other words, when the current solution cannot provide the required rate, the algorithm switches to the throughput-maximization mode, where the critical cuts are allocated more resources and the resource efficiency consideration is temporarily put aside.

For the convergence properties of this constrained subgradient method, please refer to [20] [21]. Note that to establish the convergence properties, we need to assume that $\mathcal{Q}$ has an nonempty interior, which amounts to the following assumption.

Assumption 1 (Interior point):

$$r_0 < R(\boldsymbol{c}). \tag{65}$$

This assumption does not impose any significant constraint in practice.

An alternative approach to handle the constraint $r_0 \le R(\boldsymbol{g})$ is via the augmented Lagrangian method. This can be done by following the standard steps for applying augmented Lagrangian method to inequality constrained nonlinear optimization problems; see Chapter 4 of [15] for details.

VI. Simulations

A. A small example

Consider the graph given in Fig. 4, with

$$\begin{aligned} U(r) &= \ln(1+r), \\ p_{vw}(g) &= ag^2 + bg, \quad \forall vw \in E, \\ c_{vw} &= c. \end{aligned}$$

Thus the parameters are a, b, c. We use a constant step size $\alpha_k = h$ in the algorithm.

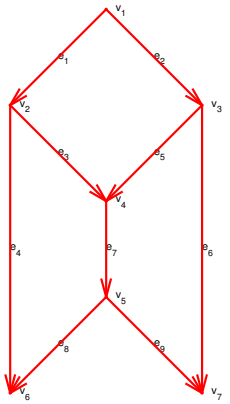

Fig. 4. An example network. This is the classical example of network coding, introduced in [1].

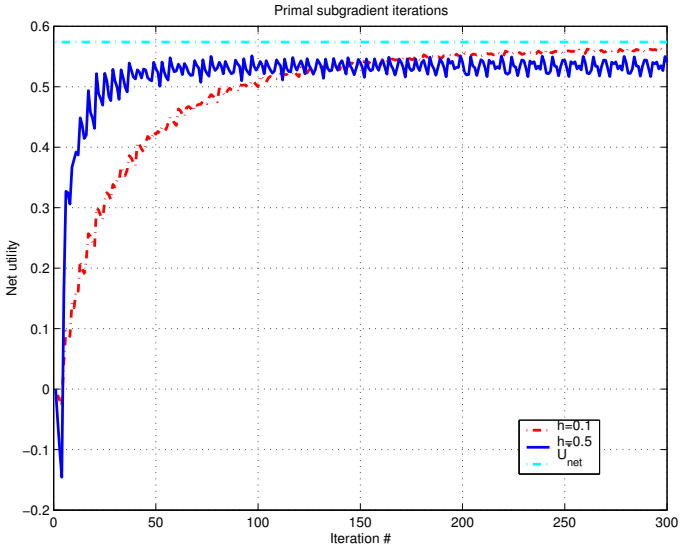

Fig. 5. Primal subgradient iterations for $a = 0.01$, $b = 0.05$, $c = 10$. The results for $h = 0.1$ and $h = 0.5$ are shown.

1) Convex cost functions: For $a = 0.01$, $b = 0.05$, $c = 10$, Fig. 5 plots the net-utilities attained by the sequence of solutions. The results for $h = 0.1$ and $h = 0.5$ are shown. With a constant step size, the subgradient method is guaranteed to eventually reach some range of the true optimum; see Lemma 2. It is seen that the curve with the smaller step size $h = 0.1$ is smoother and reaches closer to the true optimum, but it converges slower.

The optimal solution is shown in Fig. 6(a), which attains the maximum net-utility 0.573847. For $h = 0.1$, the solution found in the last iteration of the subgradient iterations is shown in Fig. 6(b), which attains a net-utility 0.5576.

2) Linear cost functions: For $a = 0$, $b = 0.05$, $c = 10$, Fig. 7 plots the net-utilities attained by the sequence of solutions with $h = 1.0$. The optimal solution is shown in Fig. 8(a), which attains the maximum net-utility 0.809438. The solution found in the last iteration of the subgradient iterations is shown in Fig. 8(b), which attains a net-utility 0.7625.

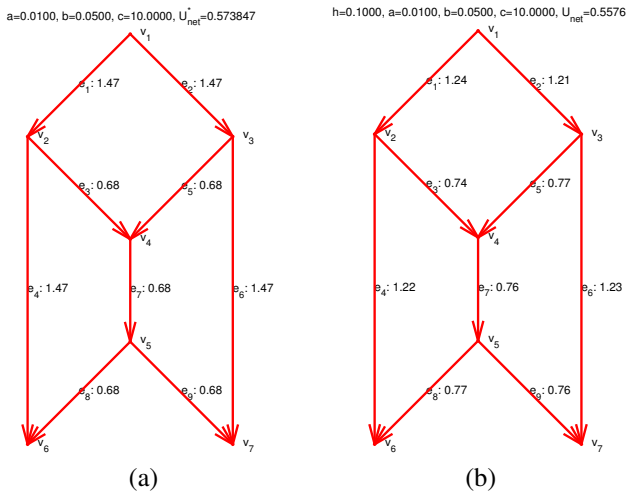

Fig. 6. (a) The optimal solution for $a = 0.01$, $b = 0.05$, $c = 10$. The maximum net-utility is 0.573847. (b) The solution found in the last iteration of the subgradient iterations for $h = 0.1$. The attained net-utility is 0.5576.

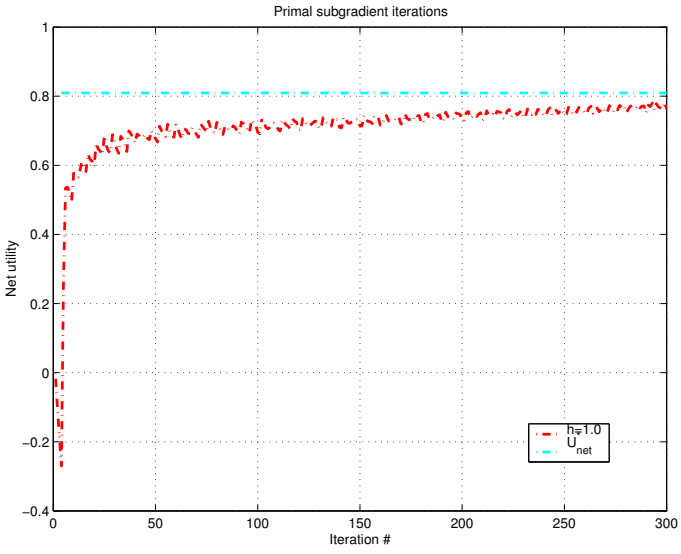

Fig. 7. Primal subgradient iterations for $a = 0$, $b = 0.05$, $c = 10$.

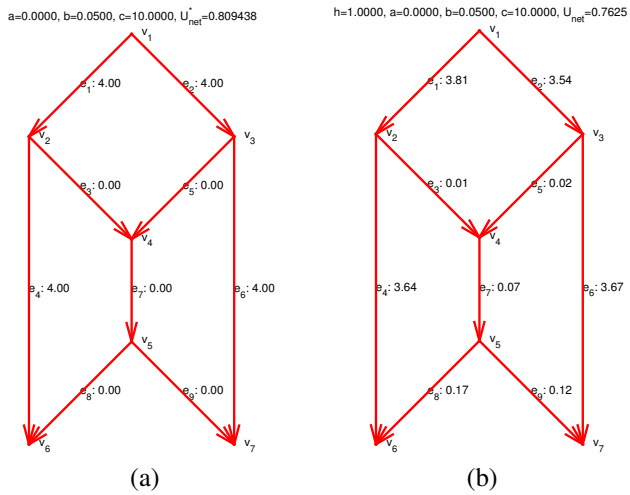

Fig. 8. (a) The optimal solution for $a = 0$, $b = 0.05$, $c = 10$. The maximum net-utility is 0.809438.(b) The solution found in the last iteration of the subgradient iterations for $h = 1.0$. The attained net-utility is 0.7625.

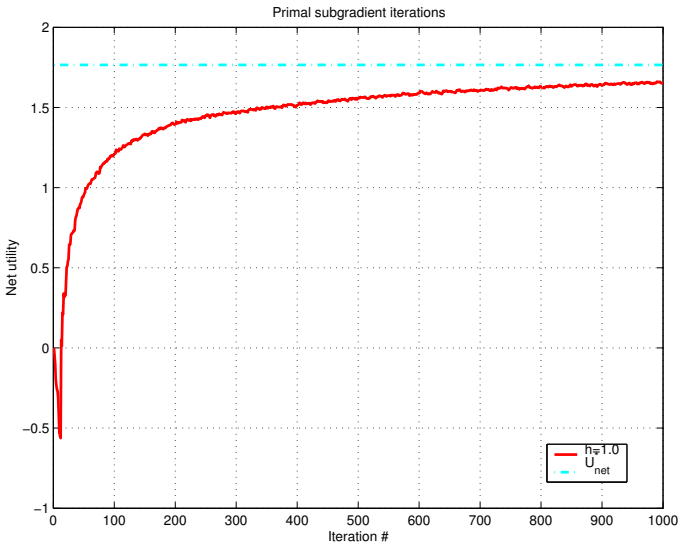

Fig. 9. Primal subgradient iterations for the large scale test case.

B. A large scale test case

We also tested the algorithm in a large scale scenario. The graph (V, E) is the topology of an ISP (Exodus) backbone obtained from the Rocketfuel project at the University of Washington [22], [23]. There are 79 nodes and 294 links. We arbitrarily placed a source node at New York, and 8 destination nodes at Oak Brook, Jersey City, Weehawken, Atlanta, Austin, San Jose, Santa Clara, and Palo Alto. The utility function, the link cost functions and link capacities are set as

$$U(r) = \ln(1 + r), \tag{66}$$

$$p_{vw}(g) = 0.005g, \quad \forall vw \in E, \tag{67}$$

$$c_{vw} = 10, \quad \forall vw \in E. \tag{68}$$

Fig. 9 presents the simulation results. The straight line is the optimal net-utility U^*_{net} and the other curve corresponds to $U^{(k)}_{\text{net}}$ generated by the proposed primal subgradient algorithm for step size $h = 1.0$. The results confirm the convergence of the proposed algorithm.

VII. Conclusion

For network-coding-based multicasting, we have proposed a net-utility maximization problem to capture the tradeoff between multicast throughput utility and resource provisioning costs. By deriving a subgradient through indicator vectors for $s - T$ critical cuts, we propose a distributed algorithm to globally solve the utility maximization problem. Further exploiting the algebraic properties of practical linear network coding, we show a low-complexity efficient implementation of the algorithm. Useful extensions to multiple multicast sessions and minimum rate guarantees are then carried out.

REFERENCES

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Information Theory*, vol. IT-46, no. 4, pp. 1204–1216, July 2000.

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

[3] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. 41st Allerton Conf. Communication, Control and Computing*, Monticello, IL, Oct. 2003.

[4] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of Operations Research Society*, vol. 49, no. 3, pp. 237–252, Mar. 1998.

[5] S. H. Low, "A duality model of tcp and queue management algorithms," *IEEE/ACM Trans. on Networking*, vol. 11, pp. 525–536, Aug. 2003.

[6] S. Kunniyur and R. Srikant, "End-to-end congestion control: utility functions, random losses and ecn marks," *IEEE/ACM Trans. on Networking*, vol. 10, no. 5, pp. 689–702, Oct. 2003.

[7] R. J. La and V. Anantharam, "Utility-based rate control in the internet for elastic traffic," *IEEE/ACM Trans. on Networking*, vol. 9, no. 2, pp. 272–286, Apr. 2002.

[8] S. H. Low and D. E. Lapsley, "Optimization flow control, i: basic algorithm and convergence," *IEEE/ACM Trans. on Networking*, vol. 7, pp. 861–874, Dec. 1999.

[9] S. H. Low, L. Peterson, and L. Wang, "Understanding vegas: a duality model," *J. of ACM*, vol. 49, pp. 207–235, Mar. 2002.

[10] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Trans. on Networking*, vol. 8, no. 5, pp. 556–567, Oct. 2000.

[11] D. S. Lun, N. Ratnakar, R. Koetter, M. Medard, E. Ahmed, and H. Lee, "Achieving minimum-cost multicast: A decentralized approach based on network coding," in *Proc. INFOCOM*. Miami, FL: IEEE, Mar. 2005.

[12] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum flow problem," in *Proc. 18th ACM Symp. Theory of Computing*, 1986, pp. 136–146.

[13] N. Z. Shor, *Minimization methods for non-differentiable functions*, ser. Springer Series in Computational Mathematics. Springer-Verlag, 1985, translated from Russian by K. C. Kiwiel and A. Ruszczyński.

[14] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient methods," Oct. 2003, lecture note.

[15] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA: Athena Scientific, 1999.

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

[17] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "Toward a random operation of networks," *IEEE Trans. Information Theory*, 2004, submitted.

[18] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial time algorithms for network code construction," *IEEE Trans. Information Theory*, vol. 51, pp. 1973–1982, June 2005.

[19] K. C. Kiwiel, "Convergence of approximate and incremental subgradient methods for convex optimization," *SIAM Journal on Optimization*, vol. 14, no. 3, pp. 807–840, 2004.

[20] Y. Nesterov, *Introductory lectures on convex optimization*. Kluwer Academic, 2004.

[21] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*. Wiley, 1993.

[22] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with Rocketfuel," in *Proc. SIGCOMM*. Pittsburg, PA: ACM, Aug. 2002.

[23] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring link weights using end-to-end measurements," in *Proc. Internet Measurement Workshop*. Marseille, France: ACM, Nov. 2002.