Robust Network Codes for Unicast Connections: A Case Study

Salim Y. El Rouayheb, Alex Sprintson, and Costas Georghiades
Department of Electrical and Computer Engineering
Texas A&M University
College Station, TX, USA

Abstract

We consider the problem of establishing reliable unicast connections across a communication network with non-uniform edge capacities. Our goal is to provide *instantaneous recovery* from single edge failures. With instantaneous recovery, the destination node can decode the packets sent by the source node even if one of the network edges fails, without the need of retransmission or rerouting.

It has been recognized that for this problem the *network coding* technique offers significant advantages over standard solutions such as disjoint path routing and diversity coding. Focusing on a practically important case in which the sender needs to deliver two packets per communication round, we present an efficient network coding algorithm over a small finite field ($GF(2)$). The small size of the underlying field results in a significant reduction of the computational and communication overhead associated with the practical implementation of the network coding technique. Our algorithm exploits the unique structure of *minimum coding networks*, i.e., networks that do not contain redundant edges.

We also consider the related capacity reservation problem and present an algorithm that achieves an approximation ratio of two compared to the optimal solution.

Index Terms

Network coding, instantaneous recovery, unicast, reliable communication.

I. Introduction

In recent years, a significant effort has been devoted to improving the resilience of communication networks to failures and increasing their survivability. Edge failures are frequent in communication networks due to the inherent vulnerability of the communication infrastructure [1]. With the dramatic increase in data transmission rates, even a single failure may result in vast data losses and cause major service disruptions for many users. Accordingly, there is a significant interest in network recovery mechanisms that enable continuous flow of data from the source to the destination with minimal data loss in the event of a failure.

Edge failures may occur due to several reasons, such as physical damage, misconfiguration, or a human error. Networks are typically designed to be resilient against a single edge failure. Indeed, protection from multiple failures incurs high costs in terms of network utilization, which is usually not justified by the rare occurrence of such failures.

In this paper, we consider the problem of establishing reliable unicast (single-source single-destination) connections across a communication network with non-uniform edge capacities. Our goal is to provide *instantaneous recovery* from single edge failures. The instantaneous recovery mechanisms ensure continuous flow of data from the source to the destination node, with no interruption and data loss in the event of a failure. Such mechanisms eliminate the need of packets retransmissions and rerouting. Instantaneous recovery is typically achieved by sending packets over multiple paths in a way that ensures that the destination node can recover the data it needs form the received packets. Below, we discuss three major techniques for achieving instantaneous recovery: *dedicated path protection scheme*, *diversity coding*, and *network coding*.

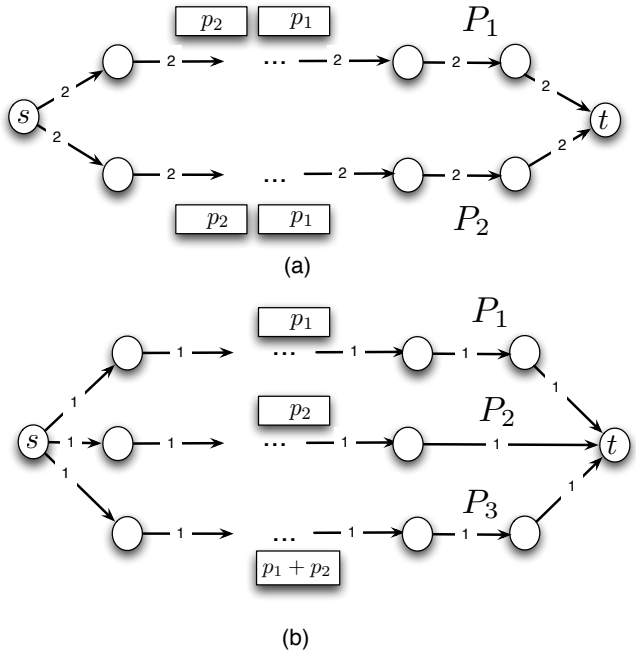
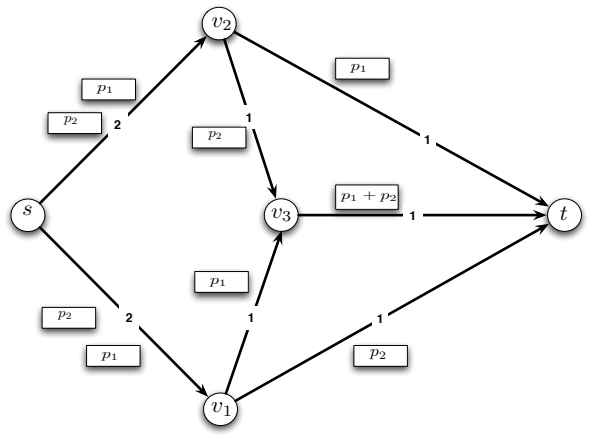


Fig. 1. (a) Dedicated path protection method ($1 + 1$ path protection); (b) Diversity coding method for $h = 2$.

Network model. We model the communication network as a directed graph $G(V, E)$. We assume that each packet is an element of a certain finite field $\mathbb{F} = GF(2^m)$, where m is the packet length (in bits). We also assume that the data exchange is performed in rounds, such that each edge $e \in E$ can transmit $c(e)$ packets per communication round. We assume that $c(e)$ is an integer number and refer to it as the *capacity* of edge e. The goal of a unicast connection is to transmit data from the source node $s \in V$ to the destination node $t \in V$. The *rate* h of the unicast connection is defined to be the number of packets that are delivered from s to t per communication round. The capacity of an (s, t)-path P,from node s to node t, is defined to be the minimum capacity of an edge that belongs to P. In the normal condition, each edge represents a lossless delay-free communication channel. Network edges can fail, but at most one of the network edges can be faulty at any given time. We assume that a failed edge cannot transmit any data and that an edge failure can be detected by its head node.

Dedicated path protection scheme. There are several techniques for achieving instantaneous recovery. A standard technique employed by today's networks is the $1 + 1$ *dedicated path protection scheme* [1]. This approach requires provisioning two disjoint paths P_1 and P_2 between s and t (see Fig. 1(a)). Each packet generated by the source node is sent over both paths, P_1 and P_2. In the case of a single edge failure, at least one of the paths remains operational, hence the destination node will be able to receive the data without interruption. With this scheme both P_1 or P_2 must be of capacity at least h. While the dedicated path protection scheme is simple and easy to implement, it incurs high communication overhead due to the need to transmit two copies of each packet. In addition, it requires two disjoint paths which include edges of high capacity.

The diversity coding technique. The *diversity coding* technique [2] extends the dedicated path protection scheme by using multiple disjoint paths for sending the data. Fig. 1(b) shows an example of a diversity coding scheme that uses three disjoint paths P_1, P_2, and P_3 between s and t. The first two paths, P_1 and P_2 transmit the original packets, while P_3 transmits parity check packets. More specifically, for $h = 2$, paths P_1 and P_2 transmit packets p_1 and p_2, respectively, while path P_3 transmits the packet $p_1 + p_2$, where p_1 and p_2 are the packets that need to be transmitted during the current communication round. Note that



	(v_1, t)	(v_3, t)	(v_2, t)
no failure	p_1	$p_1 + p_2$	p_2
(s, v_1)	0	p_1	p_2
(s, v_2)	p_1	b	0
(v_2, v_3)	p_1	p_1	p_2
(v_1, v_3)	p_1	p_2	p_2
(v_2, t)	0	$p_1 + p_2$	p_2
(v_1, t)	p_1	0	p_2
(v_3, t)	p_1	$p_1 + p_2$	0

(b)

Fig. 2. (a) A network coding approach for $h = 2$ (b) A list of the packets received by the destination t for each single edge failure scenario.

three disjoint paths can also be used for larger values of h by appropriate scaling of the edge capacities. Specifically, suppose that h is an even number and let $p_1, p_2, p_3, \ldots p_h$ be a set of packets that need to be transmitted during the current round. Then, path P_1 transmits odd packets $p_1, p_3, \ldots, p_{h-1}$, path P_2 transmits even packets $p_2, p_4, \ldots, p_h$, and path P_3 transmits parity check packets $p_1 + p_2, \ldots, p_{h-1} + p_h$ (all operations are over $GF(2)$). Note that each path P_1, P_2, and P_3 must have capacity of at least $\frac{h}{2}$.

In general, the diversity coding scheme may include $k > 3$ disjoint paths. In this case, each of the paths transmits $\lceil \frac{h}{k-1} \rceil$ packets per round, hence the capacity of each of the paths must be at least $\lceil \frac{h}{k-1} \rceil$.

The network coding technique. While the disjoint coding technique offers more flexibility than the dedicated path protection scheme, it is not the most general approach. For example, consider the network depicted in Fig. 2(a). In this network, edges (s, v_1) and (s, v_2) have capacity two, while all other edges have unit capacity. Our goal is to establish a unicast connection that delivers two packets from s to t per communication round. We note that this network does not contain two disjoint paths of capacity two between s and t, hence the dedicated path protection scheme cannot be used. Note also that the diversity coding approach cannot be used as well, because this network does not have three disjoint paths that connect s and t. However, instantaneous recovery from edge failures can be achieved by using the network coding approach. With this approach, the intermediate node v_3 combines packets received over its two incoming edges. Fig. 2(b) shows that the destination node can decode the packets sent by the source node in any single edge failure scenario. Note that without the encoding operation at the intermediate node v_3, instantaneous recovery would not be possible. In fact, network coding is the most general approach for providing instantaneous recovery from edge failures. In particular, the network coding approach enables instantaneous recovery for any settings where such recovery is possible.

Path diversity *vs.* capacity requirements. Note that in the diversity coding technique, there is a trade-off between the path diversity and the capacity requirement of the disjoint paths. In particular, the more disjoint paths are available between the source and the destination nodes, the less is the capacity requirement on the paths, and, as a result, the smaller is the total communication overhead. Specifically, with the diversity coding scheme with k disjoint paths, the capacity requirement is equal to $\lceil \frac{h}{k-1} \rceil$, and the total data sent over the network is equal to $k \cdot \lceil \frac{h}{k-1} \rceil$ per round. We observe that for the purpose of the analysis of the diversity coding technique with k disjoint paths we can assume, without loss of generality, that $h = k - 1$. Indeed, the larger values of h can be handled by scaling edge capacities.

A similar trade-off exists for the network coding approach. In this paper, we can restrict our attention to the network topologies that correspond to $h = 2$. Intuitively, each network topology we consider can

be divided into several parts, each part is either comprised of two disjoint paths of capacity two or three disjoint paths of capacity one. The network coding operations must be performed on some of the nodes that connect these parts. This has practical importance because in a typical network scenarios it is unlikely that more than three disjoint paths will be used for a single connection. Note that our approach can be used for sending more than two packets per time unit by appropriate scaling of edge capacities.

A. Related work

The network coding technique has been introduced in the seminal paper of Ahlswede et al. [3]. Initial work on network coding has focused on multicast connections. It was shown in [3] that the maximum rate of a multicast network is equal to the minimum total capacity of a cut that separates the source from a terminal. This maximum rate can be achieved by using linear network codes [4]. Koetter and Médard [5] developed an algebraic framework for linear network codes. Ho et al. [6] showed that the maximum rate can be achieved by using random linear network codes. Jaggi et al. [7] proposed a deterministic polynomial-time algorithm for finding feasible network codes in multicast networks. Network coding for networks with cycles has been studied in [8] and [9]. Network coding algorithms resilient to malicious interference have been studied in [10], [11], and [12]. Comprehensive surveys on the network coding techniques are available in the recent books [13], [14], and [15].

The idea of using network coding for instantaneous recovery from edge failures was first described by Koetter and Medard [5]. They showed that if the network has a sufficient capacity to recover from each failure scenario (e.g., by rerouting) then instantaneous recovery from each failure scenario can be achieved by employing linear network codes. Jaggi et al. [7] presented a polynomial-time algorithm for finding robust linear network codes. In [16], an information-theoretic framework for network management for recovery from edge failures has been presented. Using network coding for reliable communication was also discussed in [17] and [18]. References [19] and [20] describe practical implementations of network coding and demonstrate its benefits for improving reliability and robustness in communication networks. The problem of minimizing the amount of network resources allocated to a coding network has been considered in [21].

B. Our contribution

The paper makes the following contributions. First, we show that *minimal* coding networks, i.e., networks that do not include redundant edges or edges of excessive capacity, have a unique combinatorial structure. More specifically, any minimal network can be decomposed into basic building blocks of types A, B, and C, as depicted in Fig. 3(a). Fig. 3(b) depicts an example of such a network which consists of five consecutive blocks of types A, C, B, A and B. Second, we exploit the combinatorial structure of such networks to show the existence of efficient network codes that require a small finite field ($GF(2)$). Finally, we present an algorithm that finds such network codes in an efficient way.

A robust network code for both unicast and multicast networks can be established through the standard network coding algorithm presented in [7]. However, this algorithm is designed for the general case, and as a result, requires a field size of $O(|E|)$, where E is the set of network edges. In contrast, our scheme requires a small field size ($GF(2)$), which does not depend on the size of the underlying communication network. The size of the finite field is a very important factor in practical implementation schemes [19] as it determines the amount of communication and computational overhead. Our algorithm has a significantly smaller computational complexity associated with finding a feasible network code then the existing solutions. Specifically, the computational complexity of our algorithm is $O(|V|^2)$ compared with $O(|E|^2)$ incurred by application of the algorithm due to [7].

We also addresse the problem of efficient allocation of network resources for a robust coding network. By exploiting the properties of minimal coding networks, we present an algorithm that finds a feasible solution whose cost is at most two times more than the optimum. To the best of our knowledge this is the best approximation ratio for the problem at hand reported in the literature.

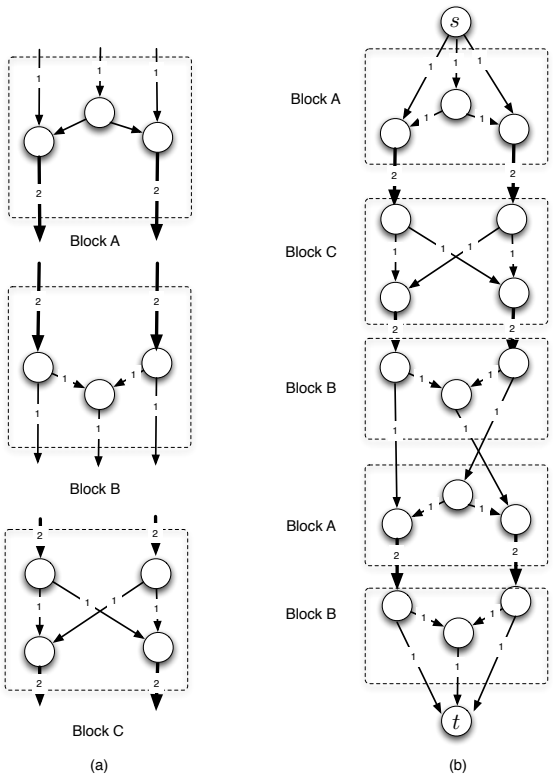


Fig. 3. (a) Basic building blocks for a unicast networks. (b) Block decomposition of a simple unicast network.

C. Paper outline

The rest of the paper is organized as follows. In Section II, we formulate the network model. In Section III, we discuss the properties of minimal networks. In Section IV, we show the structure of the minimum coding networks for the problem at hand. Next, in Section V, we describe an efficient algorithm for finding a network code over a small finite field. In Section VI, we discuss the capacity reservation scheme. Finally, conclusions and directions for future work appear in Section VII.

II. Model and preliminaries

A. Network codes

For clarity of presentation, we define an auxiliary graph $\hat{G}(V, A)$ formed by the network graph $G(V, E)$ by substituting each edge $e \in E$ by $c(e)$ parallel *arcs* that have the same tail and head nodes as e; each arc can transmit one packet per round. We denote by $A(e) \subseteq A$ the set of arcs that correspond to edge e. In what follows we only refer to packets sent at the current communication round. The packets sent in the subsequent rounds are handled in a similar manner.

We denote by $\mathbb{P} = \{p_1, p_2, \ldots, p_h\}$ the set of h packets that need to be delivered from s to t at the current communication round. A network code is defined by associating with each arc $a(v, u) \in A$ in the network an encoding function f_a that specifies the packet transmitted on arc a each time unit. For each arc $a(s, u) \in A(E)$, f_a is a function of the original h packets $\mathbb{P}$, i.e., $f_a : \mathbb{F}^h \rightarrow \mathbb{F}$. For each arc $a(v, u) \in A(E)$, $v \neq s$, f_a is a function of the packets received by node v at the current round, i.e., $f_a : \mathbb{F}^l \rightarrow \mathbb{F}$, where l is the number of incoming arcs of v in $\hat{G}$. A network code $\mathbb{C}$ is a set of encoding functions associated with the arcs in $A(E)$, i.e., $\mathbb{C} = \{f_a \mid a \in A(E)\}$. In a *linear network code* all packets are elements of a finite field and all encoding functions are also linear over that field.

As mentioned in the introduction, we assume that only one of the edges in the network can fail at any time. Since a failed edge e cannot transmit packets, we assume that the encoding function f_a of each arc

$a \in A(e)$ is identically equal to zero, i.e., $f_a \equiv \mathbf{0}$. To guarantee instantaneous recovery, it is sufficient to ensure that for each edge failure there exists a set of h linearly independent packets among the packets received by t.

Definition 1 (Robust Network Code): A network code $\mathbb{C}$ is said to be *robust*, or *resilient to single edge failures*, if for each $e \in E$ it holds that the destination node t can reconstruct the h packets sent by the source node s when all arcs in $A(e)$ fail.

B. Flow and cut conditions

A cut $C = (V_1, V_2)$ in graph $G(V, E)$ is a partition of the nodes of V into two subsets V_1 and $V_2 = V \backslash V_1$. We say that a cut $C = (V_1, V_2)$ is an (s, t)-cut if it separates nodes s and t, i.e., if $s \in V_1$ and $t \in V_2$. We say that an edge $e \in E$ belongs to the cut (V_1, V_2) if its tail node belong to V_1 and its head node belong to V_2. The capacity of the cut is defined to be the total capacity of all edges that belong to the cut.

An (s, t)-flow θ in a graph $G(V, E)$ is a function $\theta : E \mapsto \mathbb{R}$ that satisfies the following two properties:

1) For all $e(u, v) \in E$, it holds that $0 \leq \theta(e) \leq c(e)$;
2) For each internal node $v \in V$, $v \neq s$, $v \notin t$ it holds that

$$\sum_{(w,v)\in E} \theta((w, v)) = \sum_{(v,w)\in E} \theta((v, w)).$$

The *value* $|\theta|$ of a flow θ is defined as $|\theta| = \sum_{(s,v)\in E} \theta((s, v)) - \sum_{(v,s)\in E} \theta((v, s))$. The cost $\omega(\theta)$ of a flow θ is defined as $\omega(\theta) = \sum_{(u,v)\in E} \theta((u, v)) \cdot m_e$, where m_e is the cost of reserving unit capacity on edge e. Throughout the paper, except for Section VI, we assume that $m_e = 1$ for all $e \in E$.

A necessary condition for instantaneous recovery is that for each $e \in E$ a network G^e formed from G by removing e must admit an (s, t)-flow of value h. By the max-flow min-cut theorem [22] this condition is equivalent to

$$\min_C \left[\sum_{e\in E(C)} c(e) - \max_{e\in E(C)} c(e) \right] \geq h, \tag{1}$$

where the minimum is taken over all (s, t)-cuts $C(V_1, V_2)$ that separate s and t in G, and $E(C)$ is the set of edges that belong to C, i.e., the set of edges that connect a node in V_1 to a node in V_2. In [5] it was shown that this condition is also sufficient for providing instantaneous recovery from edge failures. Moreover, it was shown that the instantaneous recovery can be achieved by using *linear* network codes. Therefore, we refer to a graph $G(V, E)$ that satisfies this condition as a *feasible* graph or network.

III. Minimal and simple networks

A network $G(V, E)$ is said to be *minimal* with respect to the the capacity function $c(e)$ if it satisfies the following two conditions:

- $G(V, E)$ is a feasible network;
- The removal of an edge or a reduction in its capacity results in a violation of the network feasibility property.

A. Reduced capacity function

Networks can be made minimal by iteratively removing redundant edges and decreasing the capacity of the remaining edges. However, this approach may incur a significant computational overhead. Accordingly, we introduce the *reduced capacity* function $\bar{c}$ that allows to identify minimal networks in a very efficient way through the application of network flow techniques. This function is also instrumental for establishing the unique combinatorial structure of simple networks.

The reduced capacity function $\bar{c}$ is defined as follows.

$$\bar{c}(e) = \begin{cases} 1.5 & \textbf{if } c(e) \geq 2; \\ 1 & otherwise \end{cases} \tag{2}$$

We refer to $\bar{c}(e)$ as a *reduced capacity* of e, as opposed to the original capacity $c(e)$ of e.

The following theorem establishes a connection between the feasibility of network $G(V, E)$ with respect to the capacity function c and the existence of a flow of value three in the network $G(V, E)$ with reduced edge capacities.

Theorem 2: Let $G(V, E)$ be a network , $s \in V$ be a source node, $t \in V$ be a destination node. Then, the network $G(V, E)$ is feasible with respect to capacity function c if and only if it admits a flow of value three with respect to the reduced capacity function $\bar{c}$.

Proof: First, we show if $G(V, E)$ is feasible, then it admits a flow of value three with respect to reduced capacity function $\bar{c}$. Let C be an (s, t)-cut in $G(V, E)$, we show that the reduced capacity of this cut is at least three. The lemma will then follow from the max-flow min-cut theorem. We observe that by Equation (1), C contains at least two edges. If C contains exactly two edges, then both edges much be of capacity two; hence their reduced capacity is equal to 1.5, or three in total. If C contains more than three edges, then their total reduced capacity is also at least three.

Second, suppose that network $G(V, E)$ admits flow of value three with respect to reduced capacities. This implies that the reduced capacity of any (s, t)-cut C in $G(V, E)$ is at least three. Since the reduced capacity of any edge is at most 1.5, the cut C has at least two edges. If C contains exactly two edges, then the reduced capacity of each edge is equal to 1.5, which implies that their original capacity is equal to 2. If C contains there edges, then the original capacity of each edge is at least one. In both cases, the conditions of Equation (1) is satisfied for $h = 2$. ■

The function $\bar{c}$ can be used to verify whether a given network $G(V, E)$ is feasible with respect to a given capacity function c. This function will also serve as a building block for the algorithm that finds minimal networks.

Suppose that $G(V, E)$ admits a flow of value three with respect to the reduced capacity function. Then, by the integrality property [22, Theorem 9.10], there always exists a minimum cost flow of value three such that $\theta(e) \in \{0, 0.5, 1, 1.5\}$ for each $e \in E$. We refer to such flow as a *half-integral* flow.

The following lemma establishes a relation between the original capacities in a minimal network and the corresponding edge flow in the network with reduced capacities.

Lemma 3: Let $G(V, E)$ be a minimal network and let θ be a half-integral flow of value three in the network $G(V, E)$ with the reduced capacity function $\bar{c}$. Then, the following conditions hold:

1) For each edge $e \in E$ for which $\bar{c}(e) = 1.5$ it holds that $\theta(e) = 1.5$;
2) For each edge $e \in E$ for which $\bar{c}(e) = 1$ it holds that either $\theta(e) = 0.5$ or $\theta(e) = 1$.

Proof: Suppose that there exists an edge e such that $\bar{c}(e) = 1.5$ and $\theta(e) \leq 1$. Let c' be the capacity function formed from c by reducing the capacity of e by one. Then, the network $G(V, E)$ will be feasible with respect to c', which contradicts the minimality assumption. Using a similar argument it can be shown that the existence of an edge e such that $\bar{c}(e) = 1$ and $\theta(e) = 0$ also contradicts the minimality of the network. ■

The next lemma shows that a minimal network $G(V, E)$ is acyclic.

Proposition 4: Let $G(V, E)$ be a minimal network with respect to the capacity function c. Then $G(V, E)$ does not contain cycles.

Proof: Suppose, by way of contradiction, that $G(V, E)$ contains a cycle W. Let θ be a minimum cost flow in $G(V, E)$ with reduced capacities. The Negative Cycle Optimality condition [22, Theorem 9.1] implies that there does not exists a cycle in $G(V, E)$ with strictly positive flow on each edge. Thus, there exists an edge $e \in W$ for which it holds that $\theta(e) = 0$, in contradiction to Lemma 3. ■

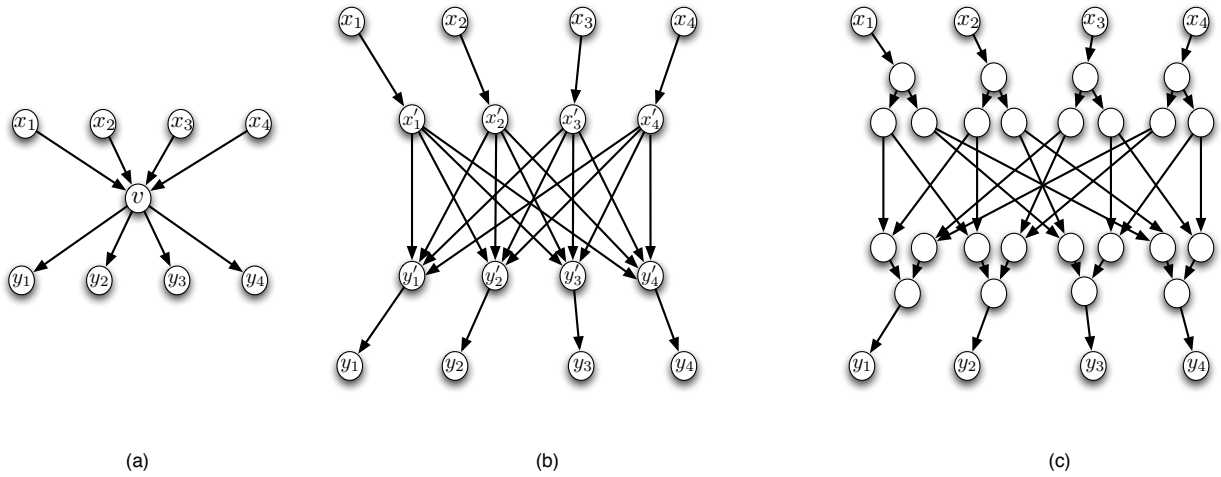


Fig. 4. (a) Node v of degree eight; (b) The intermediate step in constructing gadget Γ_v; (c) The final step in constructing gadget Γ_v.

B. Simple networks

As mentioned in the Introduction, our goal is to establish the combinatorial structure of minimal networks. For clarity of presentation, we focus on a special class of such networks, referred to as *simple networks*. For any minimal network a corresponding simple network can be constructed trough a simple and efficient procedure.

Definition 5 (Simple Unicast Network): A unicast network $G(V, E)$ is said to be *simple* with respect to the capacity function c if it satisfies the following conditions:

1) $G(V, E)$ is a minimal network with respect to c;
2) The source node s has exactly three outgoing edges of capacity one; the destination node t has exactly three incoming edges of capacity one;
3) The degree of each node $v \notin \{s, t\}$ is exactly three;
4) For every two nodes u and v, there is at most one edge in E from u to v, i.e., E does not contain parallel edges.

We proceed to describe an algorithm that transforms any arbitrary network $G(V, E)$ with capacity function c into a simple network. The transformation includes a sequence of steps that remove redundant edges and reduce excessive edge capacities. The transformation preservers the feasibility of the graph. Moreover, any feasible network code for the simple network can be used for the original network as well, with some straightforward modifications. Our algorithm includes the following steps:

1) Add a new source node $\hat{s}$ to G and connect it to s by three edges of capacity one. Similarly, add a new destination node $\hat{t}$ and connect t to $\hat{t}$ by three edges of capacity one.
2) Find a minimum cost half-integral flow θ of value three with respect to unit edge costs and with respect to reduced capacity function $\bar{c}$ (as defined by Equation (2)).
3) Remove redundant edges and decrease capacities:
   a) Remove from $G(V, E)$ all edges e for which it holds that $\theta(e) = 0$;
   b) For each edge $e \in E$ for which it holds that $\theta(e) \in \{0.5, 1\}$ set $c(e) \leftarrow 1$;
   c) For each edge $e \in E$ for which it holds that $\theta(e) \in \{1.5\}$ set $c(e) \leftarrow 2$.
4) Substitute each internal node $v \in V$, $v \notin \{\hat{s}, \hat{t}\}$, in the resulting network of degree larger than three by a gadget Γ_v, constructed as follows:
   a) Let E_v^{in} and E_v^{out} be the incoming and outgoing edges of v, respectively. For each edge $(x, v) \in E_v^{in}$ we add a node x' to Γ_v and substitute edge (x, v) by edge (x, x') (of the same capacity). Similarly, for every edge $(v, y) \in E_v^{out}$ we add a node y' to Γ_v and substitute edge

(v, y) by edge (y', y) (of the same capacity). Next, for each edge $(x, v) \in E_v^{in}$ and each edge $(v, y) \in E_v^{out}$ we connect the nodes x' and y' by edges of capacity two. Fig. 4(b) depicts an example of the resulting gadget;

b) Each node whose in-degree is equal to one and out-degree is more than two is substituted by by a binary tree as depicted in Fig. 4(c). A similar operation is performed for nodes whose out-degree is equal to one and whose in-degree is more than one. Note that in the resulting network, the total degree of each node is at most three.

5) For each edge $e \in E$ we check if the removal of e from E would result in a violation of the feasibility condition as per Equation (1). To this end we check whether there exists an (s, t)-flow of size three in the network $G(V, E)$ with respect to reduced capacities. A similar procedure is performed to reduce the capacity of each edge to the minimum possible amount while keeping the network feasible.
6) If $v \in G(V, E)$ is of degree two, then v has one incoming edge (u, v), and one outgoing edge (v, w). For each such node v, we substitute edges (u, v) and (v, w) by a single edge (u, w) of the same capacity and removing node v.
7) For every two nodes v and $u \in V \setminus \{s, t\}$ connected by two parallel edges $e'(v, u)$ of capacity c' and $e''(v, u)$ of capacity c'', we substitute the edges $e'(v, u)$ and $e''(v, u)$ by a single edge $e(v, u)$ of capacity $c' + c''$.

The purpose of Step 1 is to ensure that the source node s has exactly three outgoing edges and the destination node t has three incoming edges of capacity one. In Step 2 we find a minimum cost flow of value three which is used in Step 3 to remove redundant edges and decrease the capacity of other edges. The goal of Step 4 to ensure that the degree of each node is bounded by three. The goal of Step 5 is to ensure the minimality of the resulting graph. In Step 6, we remove nodes of degree two. In Step 7 we substitute parallel edges by a single edge of larger capacity. Note that the transformation to a simple network is not unique in general.

We note that constructing a simple network can be accomplished in $O(V^2)$ time. Indeed, Step 1 requires linear time, while steps 2 and 3 require $O(E)$ time. We also note that after Step 2, the network contains only $O(V)$ edges. This implies that the computational complexity of Steps 5, 6, and 7 is $O(V)$. Finally, Step 4 requires $O(V^2)$ time.

IV. Structure of simple networks

A. Node properties of simple unicast networks

Let $G(V, E)$ with source node s and destination node t be a simple network with respect to the capacity function c. We say that a node $v \in V$ is of Type I if it has one incoming edge and two outgoing edges, all of capacity one; of Type II if it has two incoming edges and one outgoing edge, all of capacity one; of Type III if it has one incoming edge of capacity two and two outgoing edges of capacity one; of Type IV if it has two incoming edges of capacity one and one outgoing edge of capacity two. Fig. 5 depicts nodes of types I-IV.

The next theorem proves that each node $v \in V \setminus \{s, t\}$ in a simple network is either of Type I, II, III, or IV.

Lemma 6: Let $G(V, E)$ be a simple network. Then, each node $v \in V \setminus \{s, t\}$ belongs to one of the types I, II, III, or IV.

Proof: Since $G(V, E)$ is a feasible graph, Theorem 2 implies that there exists a flow of value three in the $G(V, E)$ with reduced edge capacities. Let θ be a minimum cost half-integral flow in $G(V, E)$ with respect to $\bar{c}$. Let $v \in V \setminus \{s, t\}$ be an internal node of the network. Since the network is simple, the total degree of v is equal to three. Assume first that v has one incoming edge e and two outgoing edges. If the capacity of e is equal to one, then, by Lemma 3 it holds that $\theta(e) \in \{0.5, 1\}$. It easy to verify that $\theta(e) = 1$, otherwise one of the outgoing edges has zero flow, in contradiction to the minimality of

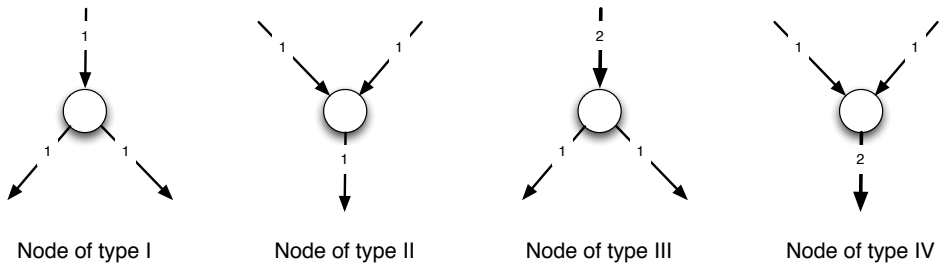


Fig. 5. The four types of nodes in a simple unicast network.

$G(V, E)$. In this case, the flow on the outgoing edges of v is equal to 0.5 and by Lemma 3 their capacity is equal to one, which implies that v is a Type I node. If the capacity of e is equal to two, then, by Lemma 3, it must be the case that $\theta(e) = 1.5$. Then, the outgoing edges of v have flow of value 0.5 and 1. Thus, by the same lemma, the capacity these edges is equal to one and, which implies that v is a Type III node.

By using a similar argument, we can show that if v has two incoming edges and one out-going edge than it is either of Type II or IV. ■

B. Residual graphs and residual cycles

Let $G(V, E)$ be a simple network, and let θ be a minimum cost half-integral flow of value three in $G(V, E)$ with respect to reduced capacity function $\bar{c}$.

We define the set $\hat{E} \subseteq E$ as follows:

$$\hat{E} = \{e \in E \mid \bar{c}(e) = 1 \text{ and } \theta(e) = 0.5\}. \tag{3}$$

Note that $\hat{E}$ includes every edge $e \in E$ for which $\theta(e) \leq \bar{c}(e)$, i.e., these edges have residual capacity and can take up more flow. Let E_1 be a certain subset of $\hat{E}$. We define the subset E_2 as follows:

$$E_2 = \{e \in E \mid \theta(e) = 1.5\} \cup \{e \in \hat{E} \mid e \notin E_1\} \tag{4}$$

Note that the set E_2 depends on set E_1. Intuitively, the set E_2 includes edges for which the amount of flow can be reduced by adding more flow to edges in E_1.

Definition 7 (Residual Graph): Let E_1 be a subset of $\hat{E}$. Then, the *residual graph* $G_{E_1}(\theta)$ of $G(V, E)$ is formed from $G(V, E)$ by reversing all edges in $E \setminus E_1$.

Let W be a cycle in the residual graph. Since the graph network is acyclic, W must contain at least one edge in E_1. By augmenting the flow θ along W we can increase the flow on edges in $W \cap E_1$ and decrease flow on other edges of W. A cycle in the residual graph that includes an edge in E_2 is referred to as a *residual cycle*. An existence of a residual cycle implies that the amount of flow on some edge in E_2 can be reduced. The following lemma shows that if $G(V, E)$ is minimal, then $G_{E_1}(\theta)$ does not contain a residual cycle.

Lemma 8: Let $G(V, E)$ be a simple network, let θ be a minimum cost half-integral flow of value three in $G(V, E)$ with respect to capacity function $\bar{c}$, and let E_1 be a subset of $\hat{E}$. Then, the residual graph $G_{E_1}(\theta)$ does not contain a residual cycle.

Proof: Suppose, by way of contradiction, that there exists a residual cycle W in $G_{E_1}(\theta)$. Such cycle must include at least one edge $e \in E_2$. Let θ' be the flow obtained by augmenting θ along W, i.e.,

$$\theta'(e) = \begin{cases} \theta(e) + 0.5 & \text{if } e \in E_1 \cap W; \\ \theta(e) - 0.5 & \text{if } e \in W \setminus E_1; \\ \theta(e) & \text{otherwise.} \end{cases}$$

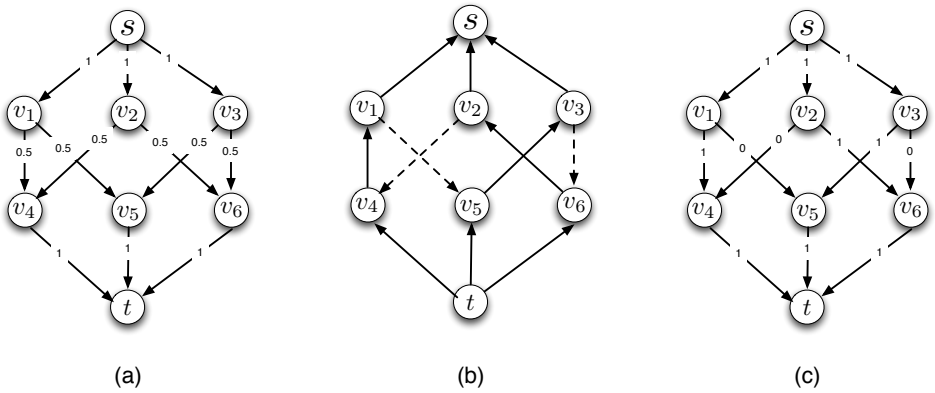


Fig. 6. (a) A graph $G(V, E)$ with edges of unit capacity and a flow θ of value three. Each edge $e \in E$ is labeled with the amount of flow $\theta(e)$ it carries. Set $\hat{E}$ includes edges $(v_1, v_4), (v_1, v_5), (v_2, v_4), (v_2, v_6), (v_3, v_5), (v_3, v_6)$. (b) Residual graph for $E_1 = \{(v_1, v_5), (v_2, v_4), (v_3, v_6)\}$. The graph contains a residual cycle $W = \{v_1, v_5, v_3, v_6, v_2, v_4, v_1\}$. (c) The flow θ' obtained from θ by augmenting along cycle W. Note that edges (v_1, v_5), (v_2, v_4), and (v_3, v_6) are redundant and can be removed from the network without violating its feasibility.

It is easy to verify that θ' is a feasible half-integer flow of value three in $G(V, E)$ with respect to $\bar{c}$. Let e be an edge of the residual cycle that belongs to E_2. Then one of the two following conditions hold:

1) $\bar{c}(e) = 1$ and $\theta'_e = 0$;
2) $\bar{c}(e) = 1.5$ and $\theta'_e = 1$.

By Lemma 3, this contradicts the minimality of $G(V, E)$. ■

Fig. 6 shows an example of a non-minimal network, the construction of a residual graph, and the result of augmenting flow θ along a residual cycle.

C. Block decomposition

Let $G(V, E)$ be a simple unicast network with at least one node other than s and t. We show that this network can be decomposed into a set of blocks of Type A, B, and C, as depicted in Fig. 3(a).

Theorem 9: Let $G(V, E)$ be a simple unicast network with $|V| > 2$. Then, $G(V, E)$ can be decomposed into blocks A, B and C, as depicted in Fig. 3(a). The blocks can be appear in an arbitrary order, subject to the following rules:

1) The first block is of Type A, and s is incident to its input edges;
2) The last block is of Type B, and the destination node t is incident its output edges;
3) A block A is followed by a block of Type B or C;
4) A block B is either followed by a block of Type A or connected to the destination;
5) A block of Type C is followed by a block of Type B or C.

Let $G(V, E)$ be a simple network and let $C(V_1, V_2)$ be an (s, t)-cut of $G(V, E)$. We denote by $E(C) \subseteq E$ the set of edges that belong to C, i.e., the set of edges that connect nodes in V_1 to nodes in V_2. We say that $C(V_1, V_2)$ is a cut of *Type 1* if $E(C)$ includes three edges of unit capacity. A cut $C(V_1, V_2)$ is said to be of *Type 2* if it includes two edges of capacity two. Fig. 7 shows examples of cuts of types 1 and 2.

In what follows we prove two lemmas that capture the properties of simple networks. The first lemma implies that any cut of Type 1 is followed by either the destination node t or a block of Type A.

Lemma 10: Let $G(V, E)$ be a simple network. Let $C = (V_1, V_2)$ be a cut of Type 1 in $G(V, E)$, i.e., $E(C)$ contains three unit capacity edges $e_1(v_1, u_1)$, $e_2(v_2, u_2)$, and $e_3(v_3, u_3)$, originating at V_1 and ending in V_2. Then, either $u_1 = u_2 = u_3 = t$ or one of the nodes u_1, u_2, or u_3 is of Type I, while two other nodes are of Type IV, and the node of Type I is adjacent to the two other nodes as depicted in Fig. 7(a).

The second theorem implies that any cut of Type 2 is followed by either a block of Type B or C.

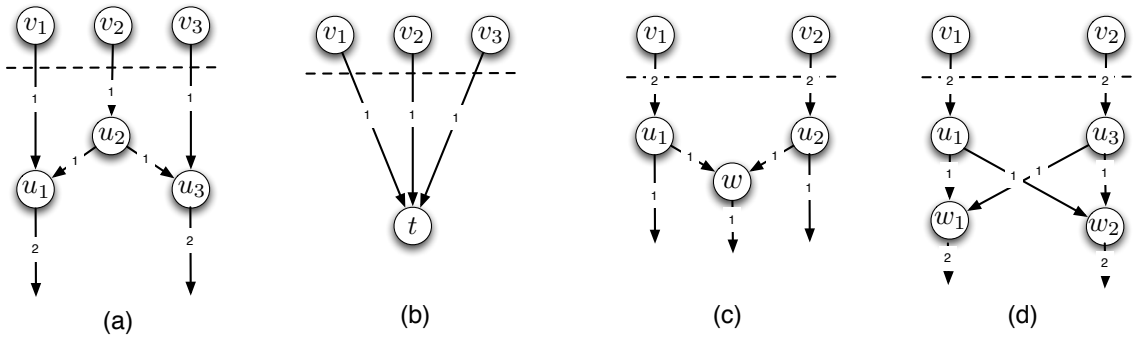


Fig. 7. (a) and (b) Examples of cuts of Type 1; (c) and (d) Examples of cuts of Type 2.

Lemma 11: Let $G(V, E)$ be a simple network. Let $C = (V_1, V_2)$ be a cut of Type 2 in $G(V, E)$, i.e., $E(C)$ includes two edges $e_1(v_1, u_1)$, $e_2(v_2, u_2)$, each one of them is of capacity two. Then, there exist either a Type II node $w \in V_2$ and two edges (u_1, w) and (u_2, w) of unit capacity, or two nodes w_1 and w_2 of Type IV and four edges (u_1, w_1), (u_1, w_2), (u_2, w_1), and (u_2, w_2) of unit capacity 1, as depicted in figures 7(c) and (d).
The theorem implies that any cut of Type 2 is always followed by a block of Type B or C.

The proof of Lemma 10 appears in Section IV-D, while the proof of Lemma 11 appears in Appendix A. It is easy to verify that lemmas 10 and 11 are sufficient for proving the correctness of Theorem 9.

D. Proof of Lemma 10

Let $G(V, E)$ be a simple network, and let θ be a flow of value three with respect to reduced edge capacities $\bar{c}$. Also, let $C(V_1, V_2)$ be an (s, t)-cut $C(V_1, V_2)$ of Type 1 in $G(V, E)$. We denote by $E(C) = \{(v_1, u_1), (v_2, u_2), (v_3, u_3)\}$ the set of the edges that belong to C.

First, we observe that each of the nodes u_i, $1 \leq i \leq 3$, is either a destination node or a node of types I or IV. Indeed, u_i cannot be of Type III because the capacity of edge (v_i, u_i) is equal to one unit. Also, in flow θ for each edge $e_i = (v_i, u_i) \in E(C)$ it must hold that $\theta(e_i) = 1$, which implies that any of u_i cannot be of Type II.

Next, we assume that $t \notin \{u_1, u_2, u_3\}$ and show, by way of contradiction, that at most one of the nodes u_1, u_2, and u_3 is of Type I. We consider two cases. In the first case, all three nodes u_1, u_2, and u_3 are Type I nodes. In the second case, two of the nodes are of Type I, while the other node is of Type IV.

Case 1. Suppose that all three nodes u_1, u_2, and u_3 are of Type I. In this case, all of the outgoing edges of u_1, u_2, and u_3 belong to $\hat{E}$ (as defined by Equation (3)). Since the in-degree of any node of $G \setminus \{s, t\}$ is either 1 or 2, we can always pick three edges e^1, e^2, and e^3, such that, for $1 \leq i \leq 3$, e^i is an outgoing edge of u_i, and e^1, e^2, and e^3 are independent, i.e., there does not exist a node v which is incident to any two edges e^i, e^j $(i \neq j)$. For example, in Fig. 6, we can chose $e^1 = (v_1, v_5)$, $e^2 = (v_2, v_4)$, and $e^3 = (v_3, v_6)$.

Let $E_1 = \{e^1, e^2, e^3\}$ and let E_2 be the set defined by Equation (4). Let $G_{E_1}(\theta)$ be the residual graph of $G(V, E)$ with respect to E_1, and let G' be the subgraph of $G_{E_1}(\theta)$ induced by nodes in V_2. We observe that each node in G' has out-degree at least one due to the following conditions:

1) None of the edges incident to the terminal node t belongs to E_1, hence t has three outgoing edges in G';
2) Any node in $G' \setminus t$ which is not a head or a tail of an edge in E_1, has at least one incoming edge that does not belong to E_1, hence its out-degree in G' is at least one.
3) Nodes u_1, u_2 and u_3 have outgoing edges e^1, e^2 and e^3, respectively. Since these edges belong to E_1 they have the same direction as in the original graph G.

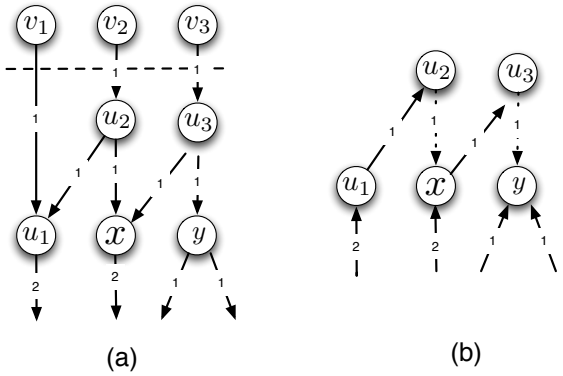


Fig. 8. (a) An example of Type 1 cut with two nodes of Type I and one node of Type IV. (b) The corresponding graph G' with $E_1 = \{(u_2, x), (u_3, y)\}$.

4) Lemma 3 implies that any head node v of an edge in E_1 is either of Type II or IV. In both cases, v have an outgoing edge that does not belong to E_1, hence its out-degree in G' is at least one.

We conclude that G' contains a cycle. Such a cycle must include at least one edge e^i, for some $1 \leq i \leq 3$, because $G' \setminus \{e^1, e^2, e^3\}$ is an acyclic graph. This implies, that this cycle includes at least one edge in E_2, which is the edge incident to u_i. Therefore, G' and, in turn, $G_{E_1}(\theta)$, include a residual cycle, which, by Lemma 8 contradicts the minimality of $G(V, E)$.

Fig. 6 depicts a reduced capacity network with a cut $C(s, E \setminus s)$ of Type I in which all the nodes u_1, u_2 and u_3 are of Type I. The corresponding residual graph contains a residual cycle, and thus the original unicast network is not minimal.

Case 2. In this case, one of the nodes, say u_1 is of Type IV, while the two other nodes u_2 and u_3 are of Type I. We choose $E_1 = \{e^2, e^3\} \subseteq \hat{E}$ such that e^2 is an outgoing edge of u_2, e^3 is an outgoing edge of u_3, neither one of them is incident to u_1, and no node is incident to both e^2 and e^3. It can be verified that such choice is always possible. Let E_2 be the set defined by Equation (4). An example of this cases is depicted in Fig. 8.

Following the same argument as above, we define by $G_{E_1}(\theta)$ the residual graph of $G(V, E)$ with respect to E_1 and by G' the subgraph of $G_{E_1}(\theta)$ induced by nodes in V_2. It is easy to verify that each node in G' has out-degree at least 1, hence G' includes a cycle. Such a cycle must include either e^2 or e^3, or both. This implies that the cycle includes an edge in E_2, which, by Lemma 8, contradicts the minimality of the network.

Next, we prove that if one of the nodes in $\{u_1, u_2, u_3\}$ is of Type I, then it must be adjacent to the two other nodes which are of Type IV. We assume, without loss of generality, that u_1 is a Type I node and u_2 and u_3 are of Type IV. Suppose, by way of contradiction, that u_1 is not adjacent to at least one of the nodes u_1 and u_2. We denote by e an outgoing edge of u_1 which is not adjacent u_2 and u_3. Let $E_1 = \{e\}$ and let E_2 be set defined by Equation (4).

Let $G_{E_1}(\theta)$ be the residual graph of $G(V, E)$ with respect to E_1 and let G' be the subgraph of $G_{E_1}(\theta)$ induced by nodes in V_2. It is easy to verify that the out-degree of each node in G' is at least one, hence hence G' includes a cycle. Such a cycle must include edge e, and, in turn, one edge in E_2 which is incident to u_1. As a result, G' and, in turn, $G_{E_1}(\theta)$ include a residual cycle, in contradiction to Lemma 8.

It can be proven, by using a similar argument that if one of the nodes $\{u_1, u_2, u_3\}$ is identical to t, then all other nodes in $\{u_1, u_2, u_3\}$ are identical to t, otherwise at least one of the nodes $\{u_1, u_2, u_3\}$ must be of Type I, in contradiction to the minimality of the original network.

We have shown that all cases other than that mentioned in the condition of the theorem contradict the minimality of the coding network, hence the theorem follows.

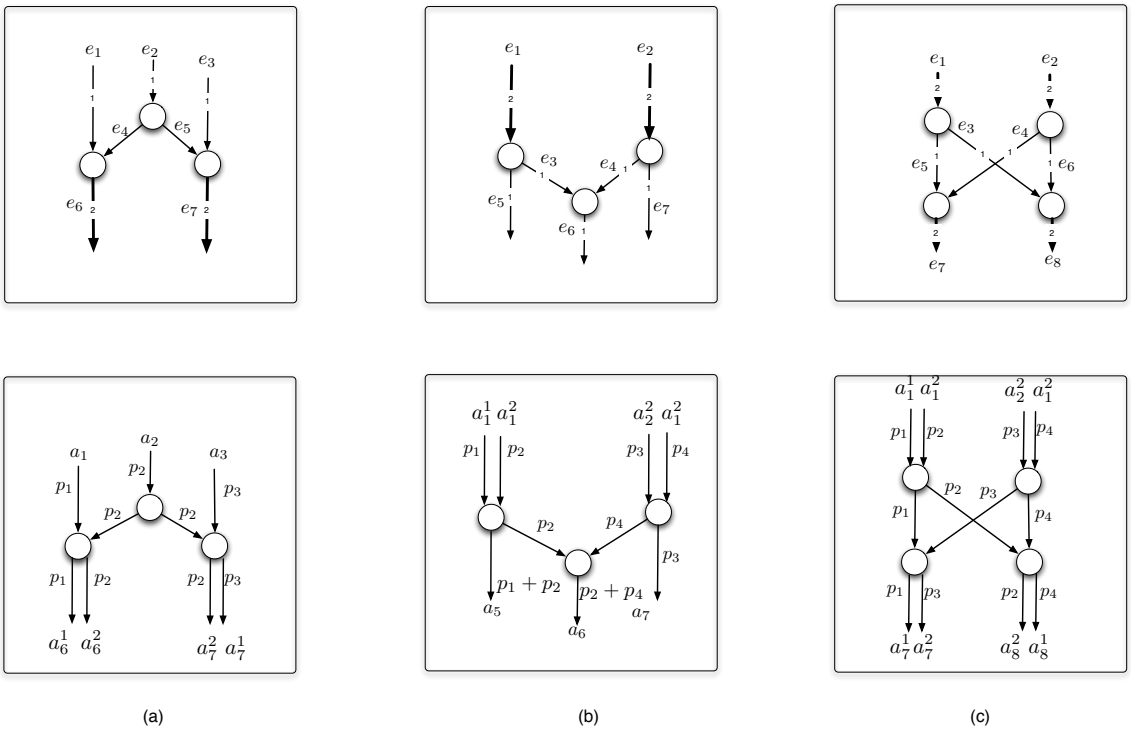


Fig. 9. Network code for simple unicast networks: (a) Encoding for blocks of Type A; (b) Encoding for blocks of Type B; Encoding for blocks of Type C.

V. Network Codes for Simple Networks

In this section, we present a robust network code over $GF(2)$ for the coding network consisting of a simple network G, source node s and the destination node t and prove its correctness.

As shown in the previous section, a simple unicast network consists of a sequence of the blocks of types A, B, and C, depicted in Fig. 3(a). The source node s has three outgoing edges of capacity one, connected to a block of Type A. We denote by $p_1, p_2 \in GF(2)$ the two packets that the source node has to transmit to the destination t at the current round. Our network code can be specified as follows. First, the source node sends packets p_1, p_2, and $p_1 + p_2$ on its outgoing arcs. Second, the encoding function for each arc that belongs to blocks A, B, and C is depicted in Fig. 9. The figure shows for each edge e_i of capacity one the corresponding arc a_i, and for each edge e_i of capacity two the two corresponding arcs a_i^1 and a_i^2. We choose the notation in such a way that if edge e_i of block X coincides with edge e_j of block Y, then arcs a_i^1 and a_i^2 of block X coincide with arcs a_j^1 and a_j^2 of block Y, respectively. Note that all the arcs that belong to the blocks of Type A and C just forward their incoming packets, while each block of Type B has two encoding nodes.

We proceed to prove that the network code described above is robust, i.e., the destination node can recover the original packets even if one of the edges in the original network fails. Consider a simple network $G(V, E)$ that contains n blocks of Type A. Recall that each node of Type A is either followed by block of Type B, or by several blocks of Type C, which, in turn, are followed by a block of Type B. We denote by A_i, $1 \leq i \leq n$, the i^{th} block of Type A from the source. We also denote by B_i, $1 \leq i \leq n$, the i^{th} block of Type B from the source such that B_1 is the first block of Type B after A_1.

We define $I_{A_i} = (p^i_{a_1}, p^i_{a_2}, p^i_{a_3})$ to be the vector of packets entering block A_i, where $p^i_{a_1}, p^i_{a_2}$ and $p^i_{a_3}$ are the packets carried by arcs a_1, a_2 and a_3 of block A_i, respectively. We also define $O_{A_i} = (p^i_{a^1_6}, p^i_{a^2_6}, p^i_{a^2_7}, p^i_{a^1_7})$ to be the vector of packets leaving block A_i, where $p^i_{a^1_6}, p^i_{a^2_6}, p^i_{a^2_7}$, and $p^i_{a^1_7}$ are the packets carried by arcs

a_6^1, a_6^2, a_7^2 and a_7^1 of block A_i, respectively. Similarly, we let $I_{B_i} = (p^i_{a_1^1}, p^i_{a_1^2}, p^i_{a_2^2}, p^i_{a_2^1})$ and $O_{B_i} = (p^i_{a_5}, p^i_{a_6}, p^i_{a_7})$ be the vectors of packets leaving block B_i, respectively.

Recall that if edge e fails, then the encoding function of each arc $a \in A(\{e\})$ that corresponds to e is identically equal to zero, i.e., $f_a \equiv 0$.

Lemma 12: Consider a block A_i, $1 \leq i \leq n$. Suppose that there are no failures in A_i, B_i, and blocks of Type C located between A_i and B_i. Suppose also that the input vector I_{A_i} of A_i is a permutation of $(p_1, p_2, p_1 + p_2)$. Then, the output vector O_{B_i} of B_i is a permutation of $(p_1, p_2, p_1 + p_2)$.

Proof: The proof immediately follows from the network code for blocks of types A, B, and C, depicted in Fig. 9. ∎

From Lemma 12 it follows that if there are no failures in blocks located between s and B_i, then the output vector O_{B_i} of B_i is a permutation of (p_1, p_2, p_1+p_2). The following lemma characterizes the output of block B_i in the case of an edge failure in the blocks A_i, B_i, or a block of Type C located between A_i and B_i.

Lemma 13: Consider a block A_i, $1 \leq i \leq n$. Suppose that there is an edge failure in A_i, B_i, or one of the blocks of Type C located between A_i and B_i. Suppose also that the input vector I_{A_i} of A_i is a permutation of $(p_1, p_2, p_1 + p_2)$. Then, one of the following holds:

1) The output vector O_{B_i} of B_i is a permutation of $(p_1, p_2, p_1 + p_2)$;
2) The output vector O_{B_i} of B_i includes two distinct elements from the set $\{p_1, p_2, p_1 + p_2\}$ and a zero;
3) The output vector O_{B_i} of B_i includes two distinct elements from the set $\{p_1, p_2, p_1 + p_2\}$, one of which appears twice.

Proof: First, we consider a case in which the failed edge belongs to block A_i. Let $O_{A_i} = (p^i_{a_6^1}, p^i_{a_6^2}, p^i_{a_7^2}, p^i_{a_7^1})$ be the output vector of block A_i and let $I_{B_i} = (p^i_{a_1^1}, p^i_{a_1^2}, p^i_{a_2^2}, p^i_{a_2^1})$ be the input vector of block B_i. For all failure scenarios in block A_i, one of the following conditions holds:

1) Vector $(p^i_{a_6^1}, p^i_{a_6^2}, p^i_{a_7^2}, p^i_{a_7^1})$ includes two distinct elements from the set $\{p_1, p_2, p_1+p_2\}$ and two zeros;
2) Vector $(p^i_{a_6^1}, p^i_{a_6^2}, p^i_{a_7^2}, p^i_{a_7^1})$ includes three distinct elements from the set $\{p_1, p_2, p_1+p_2\}$ and one zero;
3) Packets $p^i_{a_6^1}$ and $p^i_{a_6^2}$ carry two distinct elements from the set $\{p_1, p_2, p_1 + p_2\}$, $p^i_{a_7^2} = p^i_{a_6^2}$, and $p^i_{a_7^1}$ is zero;
4) Packets $p^i_{a_7^1}$ and $p^i_{a_7^2}$ carry two distinct elements from the set $\{p_1, p_2, p_1 + p_2\}$, $p^i_{a_6^2} = p^i_{a_7^2}$, and $p^i_{a_6^1}$ is zero.

Due to the structure and the form of the encoding functions of blocks of Type C, it holds that the same condition holds if we substitute packets $p^i_{a_6^1}$, $p^i_{a_6^2}$, $p^i_{a_7^2}$, $p^i_{a_7^1}$ by input packets $(p^i_{a_1^1}, p^i_{a_1^2}, p^i_{a_2^2}, p^i_{a_2^1})$ of block B_i, respectively. It can be easily verified that the output vector O_{B_i} of B_i satisfies the condition of the lemma.

Next, we consider the case in which the failed edge belongs to one of the blocks of Type C located between A_i and B_i. Let $\hat{C}$ be a block with a faulty edge. We denote by $p_{a_1^1}$, $p_{a_1^2}$, $p_{a_2^2}$, and $p_{a_2^1}$ the packets that arrive to arcs a_1^1, a_1^2, a_2^2, and a_2^1 of $\hat{C}$, respectively. Since the preceding blocks of $\hat{C}$ do not contain failed edges it holds that the vector $(p_{a_1^1}, p_{a_1^2}, p_{a_2^2}, p_{a_2^1})$ contains three distinct packets from (p_1, p_2, p_1+p_2) and $p_{a_1^2} = p_{a_2^2}$. Let $p_{a_7^1}$, $p_{a_7^2}$, $p_{a_8^2}$, and $p_{a_8^1}$ the packets carried by the output arcs a_7^1, a_7^2, a_8^2, and a_8^1 of $\hat{C}$. It is easy to verify that these packets satisfy one of the following conditions:

1) Vector $(p_{a_7^1}, p_{a_7^2}, p_{a_8^2}, p_{a_8^1})$ includes two distinct elements from the set $\{p_1, p_2, p_1+p_2\}$ and two zeros;
2) Vector $(p_{a_7^1}, p_{a_7^2}, p_{a_8^2}, p_{a_8^1})$ includes three distinct elements from the set $\{p_1, p_2, p_1+p_2\}$ and one zero.

Due to the structure and the form of the encoding functions of blocks of Type C, it holds that the came condition holds if we substitute packets $p_{a_7^1}$, $p_{a_7^2}$, $p_{a_8^2}$, $p_{a_8^1}$ by input packets $(p^i_{a_1^1}, p^i_{a_1^2}, p^i_{a_2^2}, p^i_{a_2^1})$ of block B_i, respectively. It can be easily verified that the output vector O_{B_i} of B_i satisfies the condition of the lemma.

Finally, we consider the case in which the failed edge belongs to block B_i. Let $(p^i_{a_1^1}, p^i_{a_1^2}, p^i_{a_2^2}, p^i_{a_2^1})$ be the input packets of block B_i. Since the preceding blocks of $\hat{C}$ do not contain failed edges, it holds that

the vector $(p^i_{a^1_1}, p^i_{a^2_1}, p^i_{a^2_2}, p^i_{a^1_2})$ contains three distinct packets from the set $\{p_1, p_2, p_1 + p_2\}$ and $p^i_{a^2_1} = p^i_{a^2_2}$. It can be easily verified that the output vector O_{B_i} of B_i satisfies the condition of the lemma. ■

Lemma 14: Consider a block A_i, $1 \leq i \leq n$. Suppose that there is no edge failure in A_i, B_i, and each of the blocks of Type C located between A_i and B_i. Suppose also that the output vector $I_{B_{i-1}}$ of the previous block B_{i+1} satisfies the conditions of Lemma 13. Then the output vector I_{B_i} of block B_i also satisfies the conditions of that lemma.

Proof: First, we note that if the output vector $O_{B_{i-1}}$ of B_{i-1} is a permutation of $(p_1, p_2, p_1 + p_2)$, then by Lemma 12 the same holds for the output vector I_{B_i} of B_i.

Next, we consider the case in which the output vector $O_{B_{i-1}}$ of B_{i-1} includes two distinct elements of $(p_1, p_2, p_1 + p_2)$ and a zero. Note that this case is equivalent to the failure of one of the incoming edges of block A_i, hence by Lemma 13 the output vector I_{B_i} of B_i satisfies the conditions of the lemma.

Finally, we consider the case in which the output vector $O_{B_{i-1}}$ of B_{i-1} includes two distinct elements of $(p_1, p_2, p_1 + p_2)$, one of which appears twice. In this case the output vector $O_{A_i} = (p^i_{a^1_6}, p^i_{a^2_6}, p^i_{a^2_7}, p^i_{a^1_7})$ of A_i satisfies the following conditions:

1) Vector $(p^i_{a^1_6}, p^i_{a^2_6}, p^i_{a^2_7}, p^i_{a^1_7})$ includes two distinct elements of the set $\{p_1, p_2, p_1 + p_2\}$;
2) Packet $p^i_{a^2_6}$ is equal to $p^i_{a^2_7}$;
3) Vector $(p^i_{a^1_6}, p^i_{a^2_6}, p^i_{a^2_7}, p^i_{a^1_7})$ does not contain zero packets.

The structure of blocks C imply that the same holds if the packets O_{A_i} or substitute by packets from $I_{B_i} = (p^i_{a^1_1}, p^i_{a^2_1}, p^i_{a^2_2}, p^i_{a^1_2})$. Thus, there are two possible cases

- Case 1: $p^i_{a^1_1} = p^i_{a^2_1} = p^i_{a^2_2}$, $p^i_{a^1_2} \neq p^i_{a^2_2}$,
- Case 2: $p^i_{a^2_1} = p^i_{a^2_2} = p^i_{a^1_2}$, $p^i_{a^1_1} \neq p^i_{a^2_1}$.

In both cases, all packets $p^i_{a^1_1}, p^i_{a^2_1}, p^i_{a^2_2}, p^i_{a^1_2}$ are non-zero. It can be verified that the encoding functions of block B ensure that the output vector I_{B_i} of B_i satisfies the conditions of the lemma. ■

We conclude by the following theorem:

Theorem 15: The proposed network code guarantees an instantaneous recovery from edge failures.

Proof: Follows from lemmas 12, 13, and 14. ■

Network coding algorithm. The algorithm for finding a feasible network code includes the following steps. First, we identify the corresponding simple network using the algorithm described in Section III-B. Then, we visit nodes of the graph in topological order and group them into blocks of types A, B, and C. Next, for each block we apply the coding scheme as described above. Finally, we determine the network code for the original network. The computation complexity of the algorithm is $O(V^2)$.

VI. Minimizing the Required Amount of Network Resources

In this section we present an efficient algorithm for capacity allocation for robust unicast networks. Our algorithm takes advantage of the properties of minimum networks, established in the previous section. In the general case, the capacity reservation problem can be formulated as follows. Consider a directed graph $G(V, E)$ with a source node $s \in V$ a destination node $t \in V$ and where each edge $e \in E$ is associated with two parameters:

1) c_e- the capacity of e, i.e., the upper bound on the number of packets that can be transmitted by c_e at each communication round;
2) m_e - the cost a reserving a unit capacity on edge e.

A reservation x in the graph $G(V, E)$ is a map $x : E \to \{0, 1, \ldots, \}$, that assigns to each edge e the non-negative integer value x_e and satisfies the following conditions

- The reservation x_e on each edge cannot exceed its capacity, i.e., $x_e \leq c_e$;
- For every (s, t)-cut C that separates nodes s and t it must hold that:

$$\sum_{e \in E(C)} c(e) - \max_{e \in E(C)} c(e) \geq h. \tag{5}$$

The problem consists of finding an optimum reservation $\hat{x}$ that minimizes the total cost $\sum_{e\in E} x_e \cdot m_e$.

The problem of efficient allocation of network resources for coding networks has been considered in [21]. The approach presented in this paper is based on the linear programming techniques and does not provide provable performance guarantees for integral capacity reservations. Resilient capacity reservations has been addressed in [23], [24]; however, the case of $h = 2$ has not been addressed. In [23] it was shown that the general version of this problem is NP-hard. In what follows we present a simple algorithm that finds the capacity reservation scheme whose cost is at most two times more than the optimum for the special case of $h = 2$. The algorithm is a variation of the algorithm presented in Section III-B and includes the following steps:

1) Find a minimum cost integral flow θ of value three with respect to reduced capacity function $\bar{c}$ (as defined by Equation (2)) and with respect to edge costs $\{m_e \ | e \in E\}$.
2) For each edge $e \in E$ set $x(e) \leftarrow \lceil \theta(e) \rceil$.

In the following theorem we show that the resulting reservation is 2-optimal.

Theorem 16: The algorithm above provides a capacity reservation vector whose cost is at most two times more than the optimal one.

Proof: Let $\hat{x}(e)$, $e \in E$ be the optimal capacity reservation scheme and let OPT to be the cost of $\hat{x}$. We define the reduced capacity function $\hat{c}$ as follows:

$$\hat{c}(e) = \begin{cases} 1.5 & \textbf{if } \hat{x}(e) = 2; \\ 1 & \textbf{if } \hat{x}(e) = 1; \\ 0 & otherwise \end{cases} \tag{6}$$

Note that since $\hat{c}_e \leq \hat{x}_e$, it holds that $\sum_{e\in E} \hat{c}_e \cdot m_e \leq \sum_{e\in E} \hat{x}_e \cdot m_e = OPT$. Since $\hat{x}$ is a feasible reservation vector, Theorem 2 implies that the reduced capacity function admits a flow $\hat{\theta}$ of value three from the source to the destination. Moreover the cost of this flow is at most $\sum_{e\in E} \hat{c}_e \cdot m_e \leq OPT$.

We note that since $\hat{c}_e \leq \hat{c}_e$, the cost of the flow θ found by the algorithm is less than that of $\hat{\theta}$. Since θ is a half-integer flow, and since $x(e) = \lceil \theta(e) \rceil$ we conclude that

$$\sum_{e\in E} x_e \cdot m_e \leq 2 \cdot \sum_{e\in E} \theta_e \cdot m_e \leq 2 \cdot \sum_{e\in E} \hat{\theta}_e \cdot m_e \leq 2 \cdot OPT.$$

■

VII. Conclusion

In this paper we addressed the problem of constructing robust network codes for unicast networks, i.e., codes that enable instantaneous recovery from single edge failures. We proved that for the case of $h = 2$, i.e., when two packets are sent from the source to destination nodes in each communication round, it is possible to efficiently construct a network code over a small finite field ($GF(2)$). To that end, we have exploited the special properties of *minimal coding networks*, i.e., networks that do not contain redundant edges. We have also addressed the problem of efficient resource allocation for this case. As a future research direction, we plan to generalize our results for multicast connections.

The open question that arises in this context is whether it is possible to characterize the structure of feasible coding networks in the case of $h > 2$ or in the case of multiple edge failures. Another open question is whether it is possible to construct robust network codes for $h > 2$ over a finite field whose size depends only on h and does not depend on the number of edges in the network.

Appendix

A. Proof of Lemma 11

Let $G(V, E)$ be a simple network, and let θ be a flow of value three with respect to the reduced edge capacities $\bar{c}$. Also, let $C(V_1, V_2)$ be an (s, t)-cut $C(V_1, V_2)$ of Type 2 in $G(V, E)$. We denote by $E(C) = \{(v_1, u_1), (v_2, u_2)\}$ the set of the edges that belong to C.

We note that nodes u_1 and u_2 must be of Type III, since only Type III nodes have incoming edges of capacity two. Thus, nodes u_1 and u_2 have two outgoing edges each of unit capacity. By Lemma 3, one of the outgoing edges of u_1 carries a flow of one unit; we denote it by $e_1^1 = (u_1, u_1^1)$. The other outgoing edge of u_1 carries a flow of 0.5 units; we denote it by $e_1^2 = (u_1, u_1^2)$. Similarly, one of the outgoing edges of u_2 curries a flow of one unit, we denote it by $e_2^1 = (u_2, u_2^1)$. The other outgoing edge of u_2 carries a flow of 0.5 units, we denote it by $e_2^2 = (u_2, u_2^2)$. A cut of Type 2 is depicted in Fig. 10.

First, suppose that none of the u_i^j nodes coincide with the terminal node t. Then, since simple networks do not have multiple edges between two nodes, we have $u_1^1 \neq u_1^2$ (i.e., u_1^1 and u_1^2 are two distinct nodes) and $u_2^1 \neq u_2^2$. Moreover, $u_1^1 \neq u_2^1$ due to flow constraints. We are then left with four possible cases.

1) All the nodes u_1^1, u_1^2, u_2^1 and u_2^2 are distinct;
2) $u_1^2 = u_2^1$ and $u_1^1 \neq u_2^2$. In other words, node u_1^2 coincides with node u_2^1, but u_1^1 and u_2^2 are distinct;
3) $u_1^2 = u_2^1$ and $u_1^1 = u_2^2$;
4) $u_1^2 = u_2^2$.

We prove, by way of contradiction, that only the last two cases are possible in simple networks. Consider the first case, and suppose that all the nodes u_1^1, u_1^2, u_2^1 and u_2^2 are distinct. We choose $E_1 = \{e_1^2, e_2^2\}$. Let $G_{E_1}(\theta)$ be the residual graph of $G(V, E)$ with respect to E_1, and G' be the subgraph of $G_{E_1}(\theta)$ induced by nodes in V_2. Also, let E_2 be set defined by Equation (4). Each node in G' has out-degree at least one, for the following reasons:

1) The terminal t is of out-degree three in G';
2) All nodes in $G \setminus \{u_1^1, u_1^2, u_2^1, u_2^2, t\}$, have at least one incoming edge that does not belong to E_1. In G' these edges become outgoing edges. Thus, these nodes have out-degree at least one;
3) Nodes u_1 and u_2 have respectively the following outgoing edges e_1^2 and e_2^2 that belong to E_1.
4) Nodes u_1^1 and u_2^1 have both incoming edges in G that do not belong to E_1. Such edges will become outgoing edges in G'.
5) Consider node u_1^2 in G. This node is either a Type II or IV. If u_1^2 is of Type IV, then u_1^2 has an incoming edge that does not belong to E_1. In G', this edge becomes an outgoing edge of this node. If u_1^2 is of Type II, then it has an incoming edge of capacity one and carrying a flow of 0.5 units. This incoming edge does not belong to E_1, since u_1^2 and u_2^2 do not coincide. Thus, u_1^2 has an incoming edge that belong to E_2, which in G' becomes an outgoing edge. The same holds for u_2^2.

Therefore, G' includes a cycle. This cycle should include either e_1^2 or e_2^2 (or both) and, in turn, e_1^1 or e_2^1 (or both). Suppose, without loss of generality, it is e_1^1. Thus, the cycle should include an incoming edge of u_1^1. In G, u_1^1 is either of Type I or IV. In both cases u_1^1 cannot have an incoming edge in E_1 since u_1^1, u_2^2 and u_1^2 are distinct by assumption. Thus, all the incoming edges of u_1^1 belong to E_2. Thus, G' must have a cycle that includes an edge in E_2, i.e. a residual cycle. Hence, G' is not minimal.

For example, consider the network depicted in Fig. 11(a). The corresponding residual graph is depicted in Fig. 11(a). This graph has a cycle $W = \{u_1, u_1^2, u_2^2, u_1^1, u_1\}$. This cycle includes the Type I node u_1^1 and its incoming edge (u_2^2, u_1^1). Since $(u_2^2, u_1^1) \in E_2$, W is a residual cycle.

Thus, at least two of the nodes u_i^j's should coincide. Note that no more than two non-terminal nodes can coincide in a simple graph because of the restriction on the total degree of the nodes.

Consider now the second case, i.e. when u_1^2 and u_2^1 coincide, but u_1^1 and u_2^2 do not. We can similarly prove here, by taking $E_1 = \{e_1^2, e_2^2\}$, that the graph is not minimal. Fig. 11(c) shows an example of this case. Thus, if $u_1^2 = u_2^1$, we must have $u_1^1 = u_2^2$. Also, symmetrically, if $u_1^1 = u_2^2$, we must have $u_1^2 = u_2^1$.

Now, if $u_1^1 \neq u_2^2$ and $u_1^2 \neq u_2^1$, then by elimination, it follows that the only possible case left is when $u_1^2 = u_2^2$.

In the case when u_1^1 or u_2^1 coincide with the terminal node, then we can show, by following the same steps as before, that both u_1^1 and u_2^1 coincide with the terminal node, and $u_1^2 = u_2^2$.

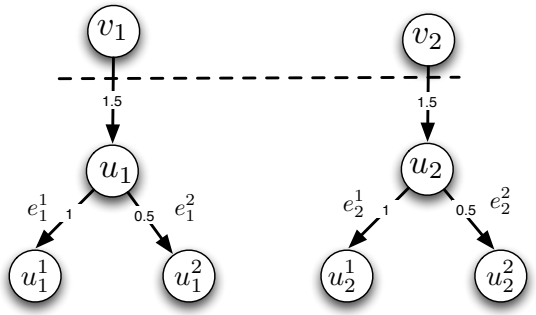


Fig. 10. A cut of Type 2. Each edge is labeled by the amount of flow it carries.

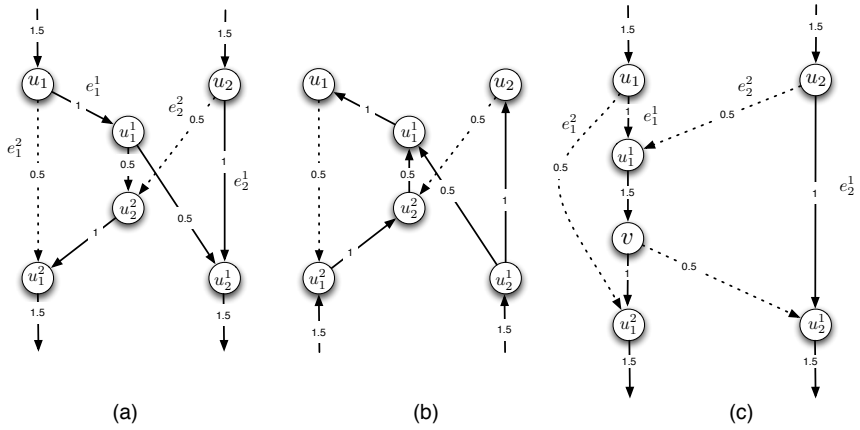


Fig. 11. Examples of subgraphs of non-minimal unicast graph that include a cut of Type 2. The labels on the edges represent the amount of flow they carry. Edge in E_1 are depicted by dashed lines. (a) An example of the case when all the nodes adjacent to u_1 and u_2 are distinct. (b) The corresponding residual graph with residual cycle $W = \{u_1, u_1^2, u_2^2, u_1^1, u_1\}$. (c) An example of the case when u_1^1 coincide with u_2^2, but u_1^2 and u_2^1 are distinct nodes. The residual cycle in this case is $W = \{u_2, u_1^1, u_1, u_1^2, v, u_2^1, u_2\}$.

REFERENCES

[1] W. D. Grover. *Mesh-based Survivable Transport Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2003.

[2] E. Ayanoglu, C.-L. I., R. D. Gitlin, and J.E. Mazo. Diversity coding for transparent self-healing and fault-tolerant communication networks. *IEEE Transactions on Communications*, 41,(11):1677–1686, 1993.

[3] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network Information Flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.

[4] S.-Y. R. Li, R. W. Yeung, and N. Cai. Linear Network Coding. *IEEE Transactions on Information Theory*, 49(2):371 – 381, 2003.

[5] R. Koetter and M. Medard. An Algebraic Approach to Network Coding. *IEEE/ACM Transactions on Networking*, 11(5):782 – 795, 2003.

[6] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros. The Benefits of Coding over Routing in a Randomized Setting. In *Proceedings of the IEEE International Symposium on Information Theory*, 2003.

[7] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen. Polynomial Time Algorithms for Multicast Network Code Construction. *To appear in IEEE Transactions on Information Theory*, 2005.

[8] E. Erez and M. Feder. Convolutional Network Codes. In *IEEE International Symposium on Information Theory*, 2004.

[9] A. Barbero and O. Ytrehus. Cycle-logical Treatment for "Cyclopathic Networks". *IEEE/ACM Transactions on Networking*, 14(SI):2795–2804, 2006.

[10] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, M. Medard, and M. Effros. Resilient Network Coding in the Presence of Byzantine Adversaries. *IEEE Transactions on Information Theory*, 54(6):2596–2603, June 2008.

[11] D. Silva, F.R. Kschischang, and R. Koetter. A Rank-Metric Approach to Error Control in Random Network Coding. *Information Theory, IEEE Transactions on*, 54(9):3951–3967, Sept. 2008.

[12] R. Koetter and F.R. Kschischang. Coding for Errors and Erasures in Random Network Coding. *Information Theory, IEEE Transactions on*, 54(8):3579–3591, Aug. 2008.

[13] T. Ho and D. S. Lun. *Network Coding: An Introduction.* Cambridge University Press, Cambrige, UK, 2008.

[14] C. Fragouli and E. Soljanin. *Network Coding Fundamentals (Foundations and Trends in Networking)*. Now Publishers Inc, 2007.
[15] R. Yeung. *Information Theory and Network Coding*. Springer, 2008.
[16] T. Ho, M. Médard, and R. Koetter. An Information-Theoretic View of Network Management. *IEEE Transactions on Information Theory*, 51(4), April 2005.
[17] Desmond S Lun, Muriel Médard, Ralf Koetter, and Michelle Effros. Further results on coding for reliable communication over packet networks. In *IEEE International Symposium on Information Theory (ISIT 05)*, 2005.
[18] Desmond S. Lun, Muriel Médard, and Michelle Effros. On coding for reliable communication over packet networks. In *in Proc. 42nd Annual Allerton Conference on Communication, Control, and Computing, Sept.–Oct. 2004, invited*, 2004.
[19] P.A. Chou and Yunnan Wu. Network coding for the internet and wireless networks. *Signal Processing Magazine, IEEE*, 24(5):77–85, Sept. 2007.
[20] C. Gkantsidis and P.R. Rodriguez. Network coding for large scale content distribution. *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 4:2235–2245 vol. 4, March 2005.
[21] D.S. Lun, M. Médard, T. Ho., and R. Koetter. Network Coding with a Cost Criterion. In *Proceedings of International Symposium on Information Theory and its Applications (ISITA 2004)*, October 2004.
[22] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Networks Flows*. Prentice-Hall, NJ, USA, 1993.
[23] G. Brightwell, G. Oriolo, and F. B. Shepherd. Reserving resilient capacity in a network. *SIAM J. Discret. Math.*, 14(4):524–539, 2001.
[24] G. Oriolo G. Brightwell and F. B. Shepherd. Reserving resilient capacity for a single commodity with upper-bound constraints. *Networks*, 41(2):87–96, 2003.