Robust Network Coding for Bidirected Networks

A. Sprintson, S. Y. El Rouayheb, and C. N. Georghiades

***Abstract*— We consider the problem of finding a linear network code that guarantees an instantaneous recovery from edge failures in communication networks. With instantaneous recovery, lost data can be recovered at the destination without the need for path re-routing or packet re-transmission.**

We focus on a special class of bidirected networks. In such networks, for each edge there exists a corresponding edge in the reverse direction of equal capacity. We assume that at most one pair of bidirected edges can fail at any time. For unicast connections, we establish an upper bound of $O(2^{2^h})$ on the minimum required field size and present an algorithm that constructs a linear network code over $GF(2^{2^h})$. For multicast connections, we show that the minimum required field size is bounded by $O(t \cdot 2^{2^h})$, where t is the number of terminals. We also discuss link- and flow-cyclic bidirected coding networks with instantaneous recovery.

I. Introduction

In past years, major effort has been spent on improving the resilience and survivability of communication networks. The major challenge is to cope with the failure of network edges. Such failures are frequent due to the inherent vulnerability of the underlying communication infrastructure [1]. With the dramatic increase in the rate of data transmission, even a single edge failure may result in vast data loss and cause major service disruptions to many users. Accordingly, many service providers are interested in providing *instantaneous recovery* from edge failures. With instantaneous recovery, the lost data can be recovered at the destination without the need for path re-routing or packet re-transmission.

The standard approach for guaranteeing instantaneous recovery is to provision two disjoint paths between the source and the destination nodes and send a copy of each packet over both paths (see Figure 1(a)). This approach, however, may result in inefficient use of network resources. An alternative approach, referred to as *diversity coding* [2], is to provision several disjoint paths, some of them are used for sending the original packets and others for sending parity check packets (see Figure 1(b)). This approach, however, is not always feasible, as it requires a large number of disjoint paths between the source and the destination nodes.

Recently, it was proposed to use the *network coding* technique for instantaneous recovery from edge failures [3], [4]. The network coding technique generalizes the disjoint path and diversity coding approaches and allows minimize the amount of network resources that need to be allocated to support instantaneous recovery. Network coding was introduced in the seminal paper by Ahlswede et al. [5]. The basic idea of network coding is to allow intermediate network nodes to generate new packets by performing algebraic operations on packets received over their incoming edges (see Figure 1(c)). This is in contrast to the traditional approach in which the intermediate nodes can only forward and duplicate their incoming packets.

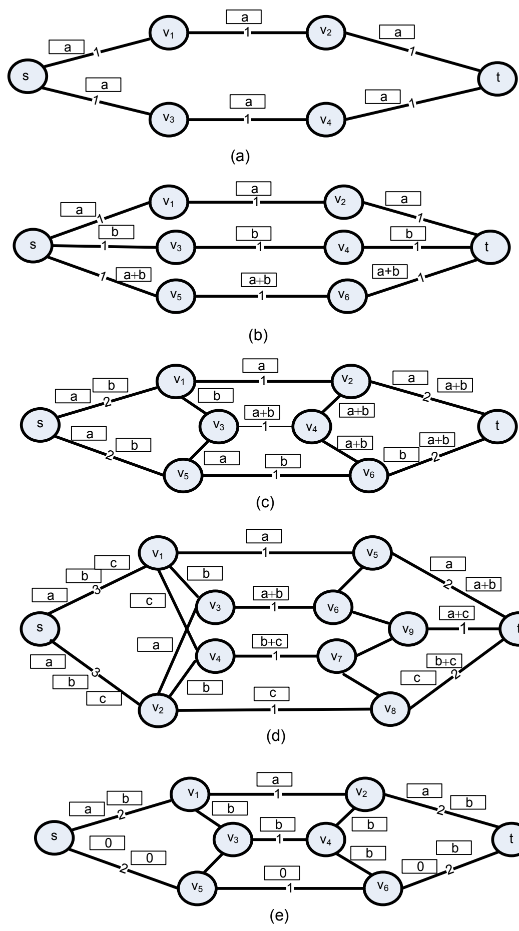

Fig. 1. Different methods for achieving instantaneous recovery (all operations are performed over $GF(2)$). (a) The disjoint path approach. A copy of each of each packet is sent over two disjoint paths that connect s and t. (b) The diversity coding approach with three disjoint paths between s and t. The first two paths are used for sending the original packet, while the third path is used for sending the parity check packet $a + b$. (c) A network coding scheme with $h = 2$. (d) A network coding scheme with $h = 3$. (e) Packets transmitted by the edges of network (c) upon a failure of edge (s, v_5).

We assume that each edge in the network has an *integer capacity*; the capacity of an edge specifies the number of packets that can be transmitted over this edge in each communication round. We also assume that packets are not fragmented and are of fixed length.

[1]The authors are with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, Texas, USA. Email: {salim,spalex,georghiades}@ece.tamu.edu

A major parameter in the design of the network coding schemes is the number of packets h sent by the source in each communication round. This parameter determines the shape of the routing topology used for data transmission. For example, for $h = 1$, the only way to guarantee instantaneous recovery is to use two disjoint paths, as depicted in Figure 1(a).[2] For $h = 2$ we can use the same topology as for $h = 1$, i.e., send two packets over two disjoint paths. In this case, all edges that belong to each of the disjoint paths must have capacity of at least two. We can also use three disjoint paths, two of which are used for sending original packets, a and b, and the third is used for transmitting the parity check packet $a+b$ (all operations are performed over $GF(2)$). Alternatively, we can use the network coding approach, as depicted in Figure 1(c). This network contains an encoding node, v_3 which receives two packets, a and b, and generates a new packet, $a + b$. A more complex network coding topology with $h = 3$ is depicted in Figure 1(d). In general, the larger is the value of h, the more flexibility we have in choosing the routing topology.

Design of reliable unicast connections that provide instantaneous recovery from link failures was investigated in [3] and [4]. It was shown that reliable communication can be achieved by using *linear network codes*, in which all operations are performed over a finite field. Practical implementation of the network coding schemes is presented in [6]. In our previous work [7] we considered the problem of establishing reliable unicast communication for the special case of $h = 2$. We showed that in this case the underlying routing topology has a certain combinatorial structure which enables the design of efficient network codes over a small field ($GF(2)$).

With linear network coding, each packet in the network is a linear combination of the packets sent by the source node. This linear combination can be captured by the *global encoding vector*. The global encoding vector can be included in the header of each packet with only a small overhead [6]. We assume that in the case of a failure of an edge all packets transmitted over this edge are identically equal to zero, i.e., have zero global encoding vectors. Thus, in the case of an edge failure, only the nodes incident to this edge change their behavior, while all other nodes perform the same operations as during the normal operation. For example, Figure 1(e) shows the packets transmitted by the edges of the network depicted on Figure 1(c) upon a failure of edge (s, v_5).

In this work, we focus on the design of unicast coding network with $h > 2$. We consider a special class of *bidirected* networks. In such networks, the capacity of each communication channel is equally split in both directions, i.e., for any edge in the underlying communication graph there exists an edge in the reverse direction of equal capacity. We assume that at most one pair $(e_1(v, u), e_2(u, v))$ of bidirected edges can fail at a time. Indeed, in many practical setting a failure of a communication channel is frequent enough in order to warrant consideration. On the other hand, protection from multiple failures incurs excessively high cost in terms of network utilization, which, typically, is not justified by the rare occurrence of simultaneous failures.

[2]We consider only minimal topologies, i.e., topologies that do not contain redundant edges.

Our work makes the following contributions. First, we investigate the instantaneous recovery from edge failures for unicast connections. We establish an upper bound of $O(2^{2h})$ on the required field size in bidirected networks. Our bound only depends on the number of packets h sent at each communication round and does not depend on the size of the underlying communication network. Next, we present an algorithm that constructs a feasible network code over $GF(2^{2h})$. Next, we extend our results for multicast connections and show that the required field size is bounded by $O(t \cdot 2^{2h})$, where t is the number of terminals. Finally, we discuss robust coding networks with cycles.

II. Model and Preliminaries

A. Communication Network

We model a communication network by a directed graph $G(V, E)$, where V is the set of nodes and E is the set of edges. Each edge is associated with a parameter c_e that specifies the *capacity* of the edge, i.e., the number of packets that can be transmitted by the edge per communication round. We assume that the graph $G(V, E)$ is *bidirected*, i.e., for each edge $e(v, u) \in G$ there exists an edge $e'(u, v)$ in the opposite direction such that $c_e = c_{e'}$.

We assume that each packet in the coding network is an element of a finite field $\mathbb{F}$. Each node in the coding network can create new packets by performing algebraic operations on the incoming packets over $\mathbb{F}$. An instance $I(G, c, s, t, h)$ of a unicast network coding problem includes a graph $G(V, E)$, the capacity function c, the source node s, the destination node t and the number h of packets that need to be transmitted from s to t per communication round.

For clarity of presentation, we use a notion of a coding network $\mathbb{N}(G_L, s, t, h)$ which includes a *link graph* $G_L(V, L)$ formed from $G(V, E)$ by substituting each edge $e \in E$ by up to c_e parallel *links*, each link can transmit one packet per communication round. We denote by L_e the set of links that correspond to edge e.

An instance $I(G, c, s, T, h)$ to the multicast network coding problem and a multicast coding network $\mathbb{N}(G_L, s, T, h)$ are defined in a similar manner. Here, T denotes the set of terminal nodes.

B. Network Codes

We denote by $\mathbb{H}^i = \{H_1^i, \ldots, H_h^i\}$ the set of packets sent by the source node at round i. For each link $l \in L$ we denote by y_l^i the packet transmitted on this link at round i.

A network code $\mathbb{F}(\mathbb{N})$ for a coding network $\mathbb{N}(G_L, s, t, h)$ ($\mathbb{N}(G_L, s, T, h)$) is defined by a set of local encoding functions $\mathbb{F}(\mathbb{N}) = \{f_l \mid l \in L\}$. If $l \in L$ is an outgoing link of the source node s, then f_l is a mapping from $\mathbb{F}^h$ to $\mathbb{F}$. Otherwise, f_l is a mapping from $\mathbb{F}^{c_{in}(v)}$ to $\mathbb{F}$, where $c_{in}(v)$ the total number of the incoming links of the tail node v of l.

The encoding function f_l of a link $l(s,v)$ determines the packet y_l^i sent on link l as a linear combination of packets $\mathbb{H}^i$ sent by the source node s at round i. Similarly, the encoding function f_l of link $l(v,u)$ specifies the packet y_l^i transmitted on link l as function of packets that arrive at node v at round $i-1$.

As mentioned in the Introduction, our goal is to design network codes that provide an instantaneous recovery upon a failure of a pair of bidirected edges. When a pair $(e_1(v,u), e_2(u,v)), e_1, e_2 \in E$ of bidirected edges fails, all links $l \in L_{e_1} \cup L_{e_1}$ cannot transmit packets. Accordingly, we assume that the local encoding functions f_l of links in $L_{e_1} \cup L_{e_1}$ are identically equal to zero.

Let $\mathbb{N}(G_L, s, t, h)$ be a unicast coding network and let $\mathbb{F}(\mathbb{N})$ be a network code for $\mathbb{N}$. We say that $\mathbb{F}(\mathbb{N})$ is *robust* if the destination node t can decode the packets sent by the source node s after a fixed number of rounds even if any two bidirected edges $(e_1(v,u), e_2(u,v))$ in E fail. A code $\mathbb{F}(\mathbb{N})$ for a multicast network $\mathbb{N}(G_L, s, T, h)$ is said to be robust if the above requirement holds for each terminal $t \in T$.

C. Cuts and Flows

We define a cut $C(V_1, V \setminus V_1)$ is a partition of V into two subsets, V_1 and $V \setminus V_1$. We say that an edge $e(v,u) \in E$ (a link $l(v,u) \in L$) is a *forward* edge (link) of C in G (G_L) if $v \in V_1$ and $u \in V \setminus V_1$. A edge $e(u,v) \in E$ (a link $l(v,u) \in L$) is referred to as a *backward* edge (link) of C if $u \in V_1$ and $v \in V \setminus V_1$. Note that the same cut $C(V_1, V \setminus V_1)$ in two different graphs $G(V,E')$ and $G(V,E'')$ may include different sets of forward and backward edges.

The previous works on robust network codes [3], [9] established a condition on the underlying network topology G_L necessary for establishing robust network codes.

Theorem 1: ([3], [9]) Let $I(G,c,s,t,h)$ be an instance of the unicast network coding problem and let $\mathbb{N}(G_L, s, t, h)$ be a corresponding unicast coding network. Then, there exists a feasible network code $\mathbb{F}(\mathbb{N})$ for $\mathbb{N}$ if and only if for each cut $C(V_1, V \setminus V_1)$ that separates s and t and for each forward edge e of C in G it holds that the total number of forward links of C in G_L is at least $h + c_e$.

We refer to the coding networks $\mathbb{N}(G_L, s, t, h)$ that satisfy the condition of Theorem 1 conditions as *feasible* networks. A similar condition holds for the multicast coding networks.

In some parts of our paper we use a notion of *network flows* [8].

Definition 2 (Flow): An *integral (s,t)-flow* θ is a function $\theta : L \mapsto \mathbb{R}$ that satisfies the following two properties:

1) For each link $l(u,v) \in L$, it holds that $\theta(l) \in \{0,1\}$
2) For each internal node $v \in V$, $v \neq s$, $v \notin t$ it holds that $\sum_{w:(w,v)\in L} \theta((w,v)) = \sum_{w:(v,w)\in L} \theta((v,w))$.

The *value* $|\theta|$ of a flow θ is defined as $|\theta| = \sum\limits_{v:(s,v)\in L} \theta((s,v))$. The cost $\omega(\theta)$ of a flow θ is defined as $\omega(\theta) = \sum\limits_{l\in L} \theta(l)$.

A minimum cost (s,t)-flow θ can be decomposed into a set of $|\theta|$ link-disjoint paths between s and t in $G_L(V,L)$ [8].

D. Main Result

In this section, we prove the main result of our paper.

Theorem 3: Let $G(V,E)$ be a bidirected graph, $G_L(V,L)$ be the corresponding link graph, and let $\mathbb{N}(G_L, s, t, h)$ be a feasible unicast network over $G_L(V,L)$. Then, there exists a robust network code $\mathbb{F}(\mathbb{N})$ for $\mathbb{N}$ over $GF(2^{2h})$. Moreover, such a network code can be constructed in polynomial time in $|V|$ and $|E|$.

This theorem establishes an upper bound on the minimum required field size of a robust network code. The theorem implies that the minimum required field size does not depend on the size of the network. We also prove the existence of a network code that can be implemented with packets of length at most $2h$ bits.

Our proof is constructive and employs the techniques and tools of the theory of network flows [8].

III. Network Coding Algorithm

The main idea of our algorithm is to identify a set Θ of (s,t)-flows in $G_L(V,L)$ such that for each pair of bidirected edges $(e_1(v,u), e_2(u,v)), e_1, e_2 \in E$ there exists a flow in $\theta \in \Theta$ that does not use neither e_1 nor e_2, i.e., $\theta(l) = 0$ for each $l \in \{L_{e_1} \cup L_{e_2}\}$. We say that such a flow θ *protects* (e_1, e_2). Given a set Θ that protects every bidirected edge pair in the network, we can use a modification of the algorithm presented in [9] in order to find a feasible coding network $\mathbb{N}(G_L, s, T, h)$ and a feasible network code $\mathbb{F}(\mathbb{N})$ for $\mathbb{N}$.

Theorem 4: Let $\mathbb{N}(G_L, s, T, h)$ be a bidirected unicast network and Θ be a set of (s,t)-flows in $\mathbb{N}$, each flow of value h such that for any pair of bidirected edges $(e_1(v,u), e_2(u,v)), e_1, e_2 \in E$, there exist a flow $\theta \in \Theta$ that protects it. Then, there exists a robust network code $\mathbb{F}(\mathbb{N})$ for $\mathbb{N}$ over a field of size $|\Theta|$. Moreover, such code can be found in polynomial time in $|V|$ and $|\Theta|$ through a randomized algorithm.

Proof: The proof follows the same lines as in [9, Theorem 2]. ■

In what follows, we present an algorithm that finds, for any feasible unicast network $\mathbb{N}(G_L, s, t, h)$, a set Θ of (s,t)-flows that protect each edge in $\mathbb{N}$, such that $|\Theta| \leq 2^{2h}$.

The algorithm includes the following steps:

1) Find a minimum cost integral (s,t)-flow θ^r of value h in $G_L(V,L)$.
2) $\Theta \leftarrow \{\theta^r\}$.
3) Decompose θ^r into h link-disjoint (s,t)-paths $\mathbb{R} = \{P_1^r, P_2^r, \ldots, P_h^r\}$. These paths are referred to as *red* paths.
4) For each subset $\mathbb{S}$ of $\mathbb{R}$ (except empty set) do:
   a) Denote by $W_{\mathbb{S}}$ the subset of edges in E, each edge $e \in W_{\mathbb{S}}$ satisfies the following conditions: (a) Each red path that belongs to $\mathbb{S}$ includes a link in L_e; (b) Each red path that does not belong to $\mathbb{S}$ does not include a link in L_e.
   b) Denote by $L(W_{\mathbb{S}})$ the set of links that correspond to edges in $W_{\mathbb{S}}$, i.e., $L(W_{\mathbb{S}}) = \cup_{e\in W_{\mathbb{S}}} L_e$. Note that

since the original graph is bidirected, for any link in $L(W_{\mathbb{S}})$ there exists a link in the reverse direction.

c) Construct an auxiliary graph $\hat{G}(\mathbb{S})$ which is obtained from $G_L(V,L)$ by removing all links in $L(W_{\mathbb{S}})$.

d) Find h link-disjoint paths between s and t. We denote these paths by $\mathbb{B} = \{P_1^b, P_2^b, \ldots, P_h^b\}$ and refer to them as *blue* paths.

e) Denote by $\mathbb{B}_1, \ldots, \mathbb{B}_{2^h}$ the set of all subsets of $\mathbb{B}$.

f) Divide the edges in $W_{\mathbb{S}}$ into 2^h subsets $W_{\mathbb{S}}^1, \ldots, W_{\mathbb{S}}^{2^h}$, that correspond to all subsets of $\mathbb{B}$. Specifically, given a subset $\mathbb{B}_i$ of $\mathbb{B}$, the set $W_{\mathbb{S}}^i$ includes all edges e in $W_{\mathbb{S}}$ that satisfy the following conditions: (a) Each path that belongs to $\mathbb{B}_i$ includes a link in $L_{e'}$ (b) Each path that belongs to $\mathbb{B} \setminus \{\mathbb{B}_i\}$ does not include a link in $L_{e'}$. Here, e' is a reverse edge of e.

g) For each subset $W_{\mathbb{S}}^i$ do

   i) Construct an auxiliary graph $\hat{G}(\mathbb{S}, i)$ formed from by $G_L(V,L)$ by removing all links in $\bigcup_{e \in W_{\mathbb{S}}^i} L_e$ and in $\bigcup_{e \in W_{\mathbb{S}}^i} L_{e'}$, where e' denotes the reverse edge of e.

   ii) Find a flow $\theta(\mathbb{S}, i)$ in $\hat{G}(\mathbb{S}, i)$ and add it to Θ.

Example 1: Consider the network $\mathbb{N}(G_L, s, t, 3)$ depicted in Figure 2(a). Figure 2(b) shows a flow θ^r that includes three red paths $\{P_1^r, P_2^r, P_3^r\}$. Figures 2(c) and (d) demonstrate the iteration of Step 4 that corresponds to $\mathbb{S} = \{P_2^r, P_3^r\}$. In this case, the set $W_{\mathbb{S}}$ includes edges (v_2, v_3), (v_3, v_5), (v_5, v_6), (v_6, v_8), (v_8, v_9), and (v_9, v_{10}). The auxiliary graph $\hat{G}(\mathbb{S})$ constructed in Step 4c appears in Figure 2(c). The set $\mathbb{B} = \{P_1^b, P_2^b, P_3^b\}$ of blue paths identified in Step 4d is shown in Figure 2(d). The auxiliary network $\hat{G}(\mathbb{S}, i)$ that corresponds to the subset $\mathbb{B}_i = \{P_3^b\}$ constructed in Step 4(g)i is shown in Figure 2(e). The figure also shows flow $\theta(\mathbb{S}, i)$ that protects edges (v_5, v_6), (v_6, v_5), (v_8, v_9), and (v_9, v_8).

Our algorithm presented can be extended for the case of multicast. This can be achieved by invoking the unicast algorithm for each of the terminals. The total number of flows is bounded by $O(2^{2h} \cdot t)$, where t is the number of the terminals.

IV. Correctness proof

We need to prove that flow set Θ protects all pairs of bidirected edges of G. First, note that the flow θ^r protects all sets of bidirected edges $(e_1(v,u), e_2(u,v)), e_1 \in E, e_2 \in E$ for which it holds that $\theta^r(l) = 0$ for each $l \in \{L_{e_1} \cup L_{e_2}\}$. It is easy to see that for any other pair of bidirected edges (e_1, e_2) it holds that either e_1 or e_2 belongs to $W_{\mathbb{S}}$ for some subset $\mathbb{S}$ of $\mathbb{R}$. Indeed, for each bidirected pair (e_1, e_2) there exists at least one link in $\{L_{e_1} \cup L_{e_2}\}$ that belongs to a red path. Next, note that each edge $e \in W_{\mathbb{S}}$ belongs to one of the subsets $W_{\mathbb{S}}^1, \ldots, W_{\mathbb{S}}^{2^h}$ defined in Step 4f. Finally, flow $\theta(\mathbb{S}, i)$, added to Θ at Step 4(g)ii of the algorithm, protects all edges of $W_{\mathbb{S}}$ and all edges whose corresponding reverse edges belong

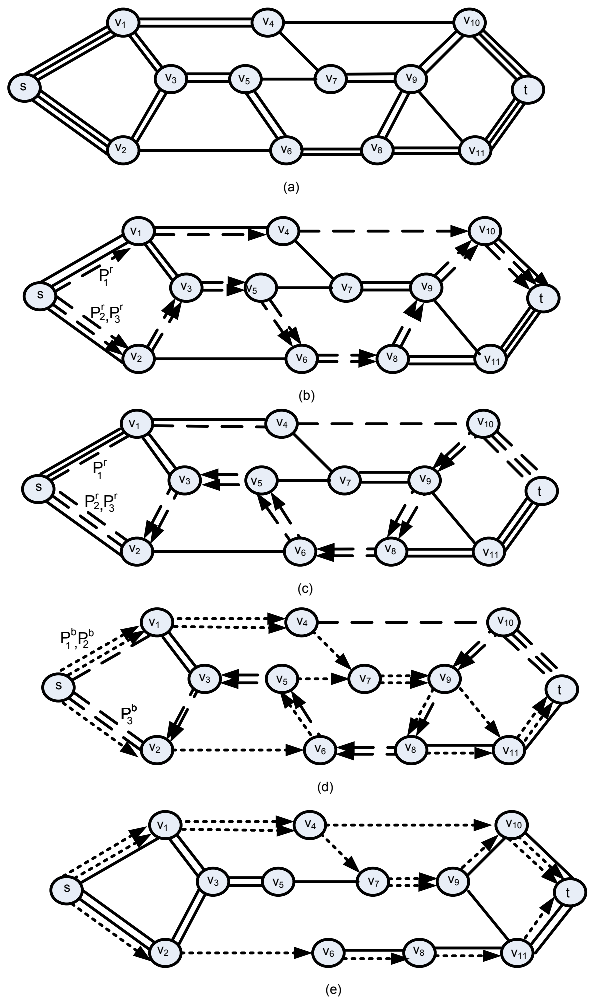

Fig. 2. (a) An example of a coding network $\mathbb{N}(G_L, s, t, 3)$ and the auxiliary graph $\hat{G}(\mathbb{S})$. Bidirected links are shown without line ends. (b) A flow $\theta^r = \{P_1^r, P_2^r, P_3^r\}$ (marked by dashed lines). (c) The auxiliary graph $\hat{G}(\mathbb{S})$ for $\mathbb{S} = \{P_2^r, P_3^r\}$. (d) The set of blue paths $\{P_1^b, P_2^b, P_3^b\}$. (e) The auxiliary network $\hat{G}(\mathbb{S}, i)$ that corresponds to the subset $\mathbb{B}_i = \{P_3^b\}$ and a flow $\theta(\mathbb{S}, i)$ that protects edges (v_5, v_6), (v_6, v_5), (v_8, v_9), and (v_9, v_8).

to $W_{\mathbb{S}}$. It is easy to verify that the total number of flows in Θ is bounded by 2^{2h}.

We proceed to show that the algorithm never fails. The following lemma proves that $\hat{G}(\mathbb{S})$ contains h link-disjoint paths between s and t. Since $\hat{G}(\mathbb{S})$ is a subgraph of $G_L(L(E), V)$, the lemma implies that the blue paths are feasible paths in $G_L(L(E), V)$.

Lemma 5: Graph $\hat{G}(\mathbb{S})$ contains h link-disjoint paths between s and t.

Proof: We prove the lemma by showing that each cut $C(V_1, V \setminus V_1)$ that separates s and t contains at least h forward links $\hat{G}(\mathbb{S})$. This implies, by the Max-Flow Min-Cut theorem [8], that there exist h disjoint paths between s and t in $\hat{G}(\mathbb{S})$.

First, we note that if C includes at most one forward edge that belongs to $W_{\mathbb{S}}$, then, by Theorem 1, this cut contains at least h forward links in $\hat{G}(\mathbb{S})$.

Next, we denote by g the number of edges of $W_{\mathbb{S}}$ which are forward edges of C. Note that each red path in $\mathbb{S}$ crosses C in the forward direction at least g times. This implies that each red path in $\mathbb{S}$ crosses C in the backward direction at least $g-1>1$ times. Since for each link that belongs to a red path there exists a link in $\hat{G}(\mathbb{S})$ in the reverse direction, the red paths in $\mathbb{S}$ correspond to at least $|\mathbb{S}|$ forward links of C in $\hat{G}(\mathbb{S})$. In addition, the red paths that do not belong to $\mathbb{S}$ include at least $h-|\mathbb{S}|$ forward links of C in $\hat{G}(\mathbb{S})$. Thus, the total number of forward links in any (s,t)-cut C of $\hat{G}(\mathbb{S})$ is at least h. This implies, by the Max-Flow Min-Cut theorem, that there are h link-disjoint paths in $\hat{G}(\mathbb{S})$. ■

The next lemma proves that $\hat{G}(\mathbb{S},i)$ contains h link-disjoint graph between s and t.

Lemma 6: Let $W^i_{\mathbb{S}}$ be a subset of $W_{\mathbb{S}}$, as defined in Step 4f of the algorithm. Let $\hat{G}(\mathbb{S},i)$ be a subgraph of $\hat{G}(\mathbb{S})$ constructed in Step 4(g)i by deleting all links that correspond to edges in $W^i_{\mathbb{S}}$ (in both forward and reverse directions). Then, $\hat{G}(\mathbb{S},i)$ contains h link-disjoint graph between s and t.

Proof: We prove the lemma by showing that each cut $C(V_1, V\setminus V_1)$ that separates s and t in $\hat{G}(\mathbb{S},i)$ contains at least h links in the forward direction. By Max-Flow Min-Cut theorem [8], this implies that there exist h disjoint paths between s and t in $\hat{G}(\mathbb{S},i)$.

Let g^f be the number of edges in $W^i_{\mathbb{S}}$ that are forward edges of C in $\hat{G}(\mathbb{S},i)$ and by g^b the number of edges in $W^i_{\mathbb{S}}$ that are backward edges of C.

We consider the two following cases:

1) Case 1: $g^f \geq g^b$. Let P^b_i be a blue path that belongs to $\mathbb{B}_i$. Note that P^b_i crosses C at least g^f times in the backward direction (because for each $e\in W^i_{\mathbb{S}}$ path P^b_i contains a link in the reverse direction). Hence, P^b_i crosses C at least g^f+1 times in the forward direction. Note when P^b_i crosses C in the forward directions, it may use edges that whose corresponding reverse edges belong to $W^i_{\mathbb{S}}$, but the number of such edges is bounded by g^b. This, each path in $\mathbb{B}_i$ includes at least one forward link of C in $\hat{G}(\mathbb{S},i)$. Next, we note that all links of each path in $\mathbb{B}\setminus\mathbb{B}_i$ exist in $\hat{G}(\mathbb{S},i)$, hence each such path includes at least one forward link of C in $\hat{G}(\mathbb{S},i)$. We conclude C contains at least h forward links.
2) Case 2: $g^b > g^f$. In this case each red path P^r_i in $\mathbb{S}$ crosses C at least g^b times in the backward direction, hence it crosses C at least g^b+1 times in the forward direction. Note when P^r_i crosses C in the forward directions, it may use edges that belong to $W^i_{\mathbb{S}}$, but the number of such edges is bounded by g^f. Thus, each red path in $\mathbb{S}$ includes at least one forward link of C in $\hat{G}(\mathbb{S},i)$. In addition, all links of the red paths that do not belong to $\mathbb{S}$ exist in $\hat{G}(\mathbb{S},i)$, hence each of these paths includes at least one forward link. We conclude C contains at least h forward links in $\hat{G}(\mathbb{S},i)$.

■

V. Networks with Cycles

In general, cycles are undesirable in coding networks. Barbero and Ytrehus [10] classify cyclic networks into two

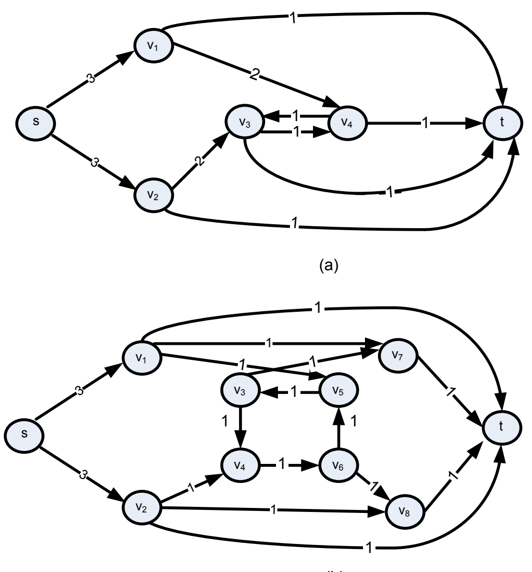

Fig. 3. (a) An example of a coding network which is link-cyclic, but not flow-cyclic. (b) An example of a flow-cyclic network.

classes, *link-cyclic* networks and *flow-cyclic* networks. A network $\mathbb{N}(G_L,s,t,2)$ is referred to as link-cyclic if the link graph the link graph $G_L(V,L)$ contains at least one cycle. A network $\mathbb{N}(G_L,s,t,2)$ is referred to as flow-cyclic if for any set Θ of flows that protect all edges in the underlying communication graph $G(V,E)$, there exists a cycle C such that any two consecutive links of C belong to one flow. For example, Figure 3(a) depicts a network which is link-cyclic, but not flow-cyclic, while Figure 3(b) depicts a flow-cyclic network.

Ahlswede et al. [5] showed that network coding approach can be applied to networks with cycles. If the coding network is link-cyclic, but not flow-cyclic, then a feasible network code can found by a modification of LIF algorithm [10]. Furthermore, such networks can be implemented as *delay-free* networks, i.e., networks with zero-delay links. For flow-cyclic networks, a feasible network code can be found by using the random coding methods proposed in [9], and, in some special cases, by the LIFE-CYCLE algorithm presented in [10].

In general, robust bidirected networks can be link-cyclic. To see this, consider the network depicted in Figure 3(a) and assume that all edges are bidirected and edges (v_3,v_4) and (v_4,v_3) are a pair of bidirected edges. It remains an open question whether bidirected networks can be flow-cyclic. In the following lemma, we prove that for $h=2$ any instance $I(G,c,s,t,h)$ of the network coding problem there exists a coding network which does not contain cycles. The lemma applies both for bidirected and directed networks.

Lemma 7: Let $I(G,c,s,t,h)$ be an instance of the network coding problem for $h=2$. Then, there exists a coding network $\mathbb{N}(G_L,s,t,2)$ such that $\mathbb{N}$ is not link-cyclic (and not flow-cyclic).

Proof: (sketch) In [7] it was shown that $I(G,c,s,t,h)$

is feasible if and only if the corresponding flow network $\hat{G}$ network admits an (s,t)-flow of value 3, where the flow network $\hat{G}$ is formed from G by reducing each edge of capacity 2 or more by an edge of capacity 1.5. Since a minimum-cost flow in $\hat{G}$ does not include cycles, the corresponding coding network $\mathbb{N}(G_L, s, t, 2)$ is acyclic. ■

VI. Conclusion and Future Directions

In this paper, we addressed the problem of finding robust network codes for bidirected communication networks. In a bidirected network, for each edge there exists an edge in the reverse direction of equal capacity. We assume that at most one pair of bidirected edges can fail at any time. Extending the results of our previous work [7], we have considered the general case of $h \geq 2$, where h is the number of packets sent at each communication round. For unicast connections we established an upper bound of $O(2^{2h})$ on the minimum field size and presented an algorithm that constructs a feasible network code over $GF(2^{2h})$. For multicast connections, we showed that the maximum size of the finite field is bounded by $O(2^{2h} \cdot t)$, where t is the number of terminals. As a future direction, we would like to address an open question of whether for each instance $I(G, c, s, t, h)$ of a unicast network coding problem with bidirected network there exists a coding network $\mathbb{N}(G_L, s, T, h)$ which is not flow-cyclic. In addition, we would like to extend our bounds for general directed networks.

References

[1] W. D. Grover. *Mesh-Based Survivable Transport Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking.* Prentice-Hall, New York, NY, 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. Koetter and M. Medard. An Algebraic Approach to Network Coding. *IEEE/ACM Transactions on Networking*, 11(5):782 – 795, 2003.

[4] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. M. G. M. Tolhuizen. Polynomial Time Algorithms for Multicast Network Code Construction. *IEEE Transactions on Information Theory*, 51(6):1973–1982, 2005.

[5] 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.

[6] P. A. Chou, Y. Wu, and K. Jain. Practical Network Coding. In *Proceedings of Allerton Conference on Communication, Control, and Computing*, Monticello, IL, October 2003.

[7] S. El Rouayheb, A. Sprintson, and C. Georghiades. Simple Network Codes for Instantaneous Recovery from Edge Failures in Unicast Connections. In *Proceedings of Proceedings of the UCSD Workshop on Information Theory and its Applications, San Diego, CA (Invited Paper)*, February 2006.

[8] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms, and applications.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[9] T. Ho. *Networking from a Network Coding Perspective.* PhD thesis, MIT, May 2004.

[10] A. I. Barbero and Ø. Ytrehus. Cycle-logical Treatment for "Cyclopathic" Networks. *IEEE Transactions on Information Theory*, 52(6):2795–2804, 2006.