Simple Network Codes for Instantaneous Recovery from Edge Failures in Unicast Connections

Salim Y. El Rouayheb Alexander Sprintson Costas N. Georghiades[1]

***Abstract*— We consider the problem of establishing reliable unicast connections in the presence of edge failures. In this problem, a source node s needs to deliver h packets to the destination node t, even if one of the edges in the network fails. We apply the the technique of network coding in order to guarantee an instantaneous recovery from edge failures. Focusing on the case of $h = 2$, we show that the underlying communication network has an elegant combinatorial structure which enables design of efficient network codes over $GF(2)$.**

I. Introduction

In this paper we investigate the problem of establishing reliable unicast connections in the presence of edge failures. Edge failures are frequent in communication networks due to the inherent vulnerability of existing network infrastructure. With the dramatic increase in the rate of data transmission, even a single failure may result in vast data loss and cause major service disruptions to many users. In past years, major effort was undertaken to improve the resilience of networks to failures and increase their survivability. In particular, providing *instantaneous recovery* from edge failures has become an important goal for many service providers. With instantaneous recovery, the lost data can be recovered at the destination, eliminating the need to introduce any changes at intermediate nodes (such as rerouting, *etc.*).

Recently, it was recognized that the novel technique of network coding is instrumental for providing instantaneous recovery from edge failures [1], [2]. Network coding was introduced in a seminal paper by Ahlswede et al. as a way of increasing throughput of multicast connections [3]. The basic idea of network coding is to allow the intermediate network nodes to generate new packets by combining packets received over their incoming edges. Network coding techniques attracted a large body of research (see [1], [2], [4], [5], [6], and references therein). In this paper, we investigate network codes that enable instantaneous recovery from a single edge failure. Previous works [1], [2] showed that this task can be achieved by employing *linear* network codes, in which all packets are symbols of some finite field $\mathbb{F}$, and each packet is a linear combination of the packets generated by source s.

We focus on unicast connections that need to deliver h packets per communication round from source node s to destination node t. The network coding technique allows to maximize the number of packets that can be delivered throughout the network with instantaneous recovery from an edge failure. For instance, consider the communication network depicted in Fig. 1. In this network, edges (v_1, v_3) and (v_1, v_2) can deliver two packets per communication round, whereas all the other edges can only deliver one packet. The network code shown in this figure allows the destination to recover two packets $a \in GF(2)$ and $b \in GF(2)$ sent by the source node t even if one of the intermediate edges fails, without any need to change the encoding or routing at intermediate nodes. We assume that all operation are performed over $GF(2)$ and all packets sent over a failed edge carry the zero symbol. It can be verified that without network coding, the instantaneous recovery can be guaranteed for at most one packet per communication round.

Contribution: We consider the problem of designing efficient network codes for instantaneous recovery from edge failures. Our contribution can be summarized as follows. First, we introduce the concept of a *simple* unicast network and show that designing a linear network code for any unicast network is equivalent to designing a code for the corresponding simple network. The key property of simple networks is minimality, i.e, the deletion of any edge from the network decreases the number of packets that can be reliably sent (with immediate restoration) from the source to destination. Then, we show that simple unicast networks have a very elegant combinatorial structure which enables the design of efficient network codes over $GF(2)$; whereas the best known bound on the field size is linear in the number of edges in the network [2].

Related work: The study of network codes for the instantaneous recovery from edge failures was done in [1]. Jaggi et al. [2] presented a randomized polynomial algorithm for finding network codes that provide instantaneous recovery from edge failures. In [7], an information-theoretic framework for network management was presented. In [8], [9], the authors describe a practical implementation of network codes, and show that it has a high potential for providing resilience to failures in practical networks.

Due to space limitation, some proofs and technical details are omitted from this paper and can be found in [10].

II. Model and Preliminaries

A. Unicast Connections

A communication network is modeled by a directed graph $G(V, E)$, where V is the set of nodes and E is the set of edges or *spans*. We consider a *unicast* setting in which the source node $s \in V$ sends data packets to the destination node $t \in V$,

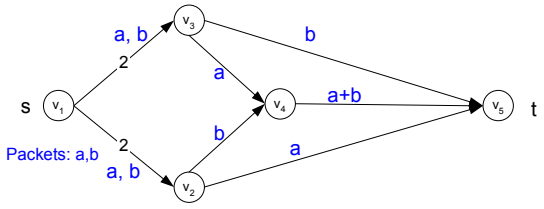

Fig. 1. An example of a robust network.

[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

each packet is a symbol of some alphabet Σ. The packets are transmitted in rounds, at each round h packets need to be delivered from s to t. Each edge $e \in E$ is associated with the capacity $c(e)$ that specifies the maximum number of packets that can be sent on edge e at each communication round. For clarity of presentation, we assume that each edge $e(v,u) \in E$ comprises $c(e)$ parallel *links*, each link can deliver one packet from s to t per communication round. The set of links that corresponds to a set of edges E is denoted by $L(E)$.

Definition 1 (Unicast Network): A unicast network $\mathbb{N}(G(V,E),c,s,t,h)$ is a 5-tuple that includes a directed graph $G(V,E)$, a capacity function $c : E \rightarrow \{1,2,...\}$, a source s, a destination t, and the required number h of packets that must be delivered from s to t at each communication round.

B. Coding Network

We focus on *linear network codes* in which the alphabet Σ is a finite field $\mathbb{F}$, and the encoding functions implemented at each node are linear over $\mathbb{F}$. Such codes are sufficient for providing an instantaneous recovery from edge failures [1].

We associate each link $l(v,u) \in L(E)$ in the network with an encoding function f_l that specifies the packet transmitted on link l. For each link $l(s,u) \in L(E)$, f_l is a function of the h packets sent by the source node s, i.e., $f_l : \Sigma^h \rightarrow \Sigma$. For each link $l(v,u) \in L(E)$, $v \neq s$, f_l is a function of packets received by node v within the same communication round, i.e., $f_l : \Sigma^m \rightarrow \Sigma$, where m is the number of incoming links of v.

Definition 2 (Network code): Let $\mathbb{N}(G(V,E),c,s,t,h)$ be a unicast network. A network code NC for $\mathbb{N}$ is the set of encoding functions associated to the links in $L(E)$, i.e., $NC = \{f_l \mid l \in L(E)\}$.

We assume that a failure of an edge $e \in E$ may result in a failure of an arbitrary subset of links that comprise e. Typically, such a failure will result in a failure of all such links. As a failed link transmits no information, we assume that its encoding function is identically zero, i.e., $f_l \equiv 0$ for every failed edge l, (where 0 is the additive identity of the field).

Definition 3 (Robust network code): Let $\mathbb{N}(G(V,E),c,s,t,h)$ be a unicast network. A network code is said to be *robust* or *resilient to a single edge failure* if at each communication round the destination node t can reconstruct the h packets sent by the source node s even if one of the edges $e \in E$ in the network fails.

C. Cut conditions

A natural question that arises is what are the necessary and sufficient conditions for the existence of a robust network code for a given unicast network $\mathbb{N}(G(V,E),c,s,t,h)$. It turns out that the feasibility of a unicast network $\mathbb{N}(G(V,E),c,s,t,h)$ can be fully characterized by a condition on the cuts of the graph $G(V,E)$. 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 $V_2 = V \setminus V_1$. We say that a cut $C = (V_1,V_2)$ separates nodes s and t if $s \in V_1$ and $t \in V_2$. We denote by $E(V_1,V_2)$ that set of edges that connect a node in V_1 to a node in V_2. The capacity of a cut C is equal to the total capacity of edges in $E(V_1,V_2)$.

Theorem 4: Let $\mathbb{N}(G(V,E),c,s,t,h)$ be a unicast network. Then, there exists a network code for $\mathbb{N}$ that is resilient to a single edge failure if and only if for every cut $C = (V_1,V_2)$ of $G(V,E)$ that separates s from t it holds that $\sum_{e \in E(V_1,V_2)} c(e) \geq h + \max_{e \in E(V_1,V_2)} c(e)$.

Proof: Follows directly from [1]. ∎

We refer to a unicast network that satisfies the condition of Theorem 4 as a *feasible* unicast network. Menger's theorem [11] implies that a unicast network $\mathbb{N}(G(V,E),c,s,t,h)$ is feasible if and only if for every edge $e \in E$ it holds that the graph $(V, L(E \setminus \{e\}))$ has h link-disjoint paths from s to t.

III. Simple unicast networks

In this section we present a special class of *simple unicast networks*. Simple unicast networks have a special structure which enables us to find a corresponding robust network code in a fast and efficient manner.

Definition 5 (Simple Unicast Network): A unicast network $\mathbb{N}(G(V,E),c,s,t,h)$ is said to be *simple* if and only if it satisfies the following properties: (1) $\mathbb{N}(G(V,E),c,s,t,h)$ is a minimal network, i.e. $\mathbb{N}$ stops being feasible upon the deletion of any link of G. (2) The source node s has exactly $h+1$ outgoing edges. (3) The destination node t has exactly $h+1$ incoming edges . (4) The total degree of every node $v \in V$, $v \notin \{s,t\}$, is exactly 3; (5) For every two nodes u and v, there is at most one edge in E from u to v.

The following theorem shows that designing a linear network code for any unicast network is equivalent to designing a code for the corresponding simple network.

Theorem 6: For every feasible unicast network $\mathbb{N}(G(V,E),c,s,t,h)$ there exists a corresponding simple network $\hat{\mathbb{N}}(\hat{G}(\hat{V},\hat{E}),\hat{c},\hat{s},\hat{t},h)$ such that the existence of a robust linear network code for $\hat{\mathbb{N}}$ implies existence of a robust code for $\mathbb{N}$ over the same field.

IV. Structure of simple networks

In this section we focus on simple unicast networks for $h = 2$, and show that such networks have a special structure that enables efficient construction of robust network codes.

A. Flow network

We begin with a definition of a flow network $\bar{\mathbb{N}}(G(V,E),\bar{c},s,t,\bar{h})$.

Definition 7 (Flow Network): A *flow network* $\bar{\mathbb{N}}(G(V,E),\bar{c},s,t,\bar{h})$ is a 5-tuple that includes a directed graph $G(V,E)$, a capacity function $\bar{c} : E \rightarrow \mathbb{R}^+$, a source node s, and a destination node t, and a flow requirement $\bar{h}$.

The cost of a flow b in $\bar{\mathbb{N}}$ is defined to be $\sum_{e \in E} b(e)$. We say that a flow network $\bar{\mathbb{N}}(G(V,E),\bar{c},s,t,\bar{h})$ is feasible if there exists a flow of value $\bar{h}$ in $\bar{\mathbb{N}}$ between source s and destination t.

For a given unicast network $\mathbb{N}(G(V,E),c,s,t,2)$, the corresponding flow network $\bar{\mathbb{N}}(G(V,E),\bar{c},s,t,3)$ is defined by setting $\bar{c}(e) = 1.5$ for any edge $e \in G$ of capacity two and $\bar{c}(e) = 1$ for each edge of unit capacity.,

Theorem 8: A unicast network $\mathbb{N}(G(V,E),c,s,t,2)$ is feasible if and only if the corresponding flow network $\bar{\mathbb{N}}(G(V,E),\bar{c},s,t,3)$ is feasible.

Proof: First, let $\mathbb{N}(G(V,E),c,s,t,2)$ be a feasible unicast network and let $\bar{\mathbb{N}}(G(V,E),\bar{c},s,t,3)$ be the corresponding flow network. Let C be a cut in $G(V,E)$ that separates s from t. We show that $\sum_{e \in E(C)} \bar{c}(e) \geq 3$. This certainly holds if $E(C)$ includes at least three edges, because the capacity

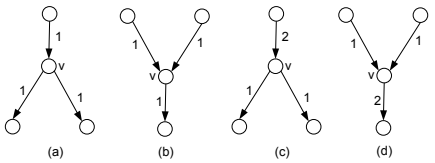

Fig. 2. Four categories of nodes (a) a node of type A (b) a node of type B (c) a node of type C (d) a node of type D

of each edge in $\bar{\mathbb{N}}(G(V,E),\bar{c},s,t,3)$ is at least one. If $E(C)$ includes only two edges, then the capacity $c(e)$ of each edge $e \in C$ is at least 2, otherwise $\mathbb{N}(G(V,E),c,s,t,2)$ would not be feasible. Therefore, $\bar{c}(e) = 1.5$ for each $e \in E(C)$. Hence, $\sum_{e\in E(C)} \bar{c}(e) \geq 3$. The Min-Cut max-flow theorem [1] implies that $\bar{\mathbb{N}}(G(V,E),\bar{c},s,t,3)$ is a feasible network.

Second, let $\mathbb{N}(G(V,E),c,s,t,2)$ be a unicast network and suppose that the corresponding flow network $\bar{\mathbb{N}}(G(V,E),\bar{c},s,t,3)$ is feasible. We show that for each cut C in $G(V,E)$ that separates s and t it holds that $\sum_{e\in E(C)} c(e) \geq h+\max_{e\in E(C)} c(e)$, which, in turn, implies that $\mathbb{N}$ is a feasible network. Since $\bar{\mathbb{N}}$ admits an (s,t)-flow of value 3, it holds that $\sum_{e\in E(C)} \bar{b}_e \geq 3$. Let e' be the edge with the maximum value of $c(e)$ in $E(C)$. If C includes at least three edges, there are at least two edges in $E(C)\setminus\{e'\}$ of capacity one. Otherwise, since $\bar{c}(e') \leq 1.5$, there must be at least one edge e in $E(C)$ for which it holds that $\bar{c}(e) = 1.5$. For this edge it holds that $c(e) = 2$ and the theorem follows. ∎

By integrality property [12, Theorem 9.10], there always exists a minimum cost flow such that $b_e \in \{0, 0.5, 1, 1.5\}$ for $\forall e \in E$. We refer to such flow as a *half-integral* flow.

B. Properties of simple unicast networks

We begin by proving two properties of simple networks.

Lemma 9: Let $\mathbb{N}(G(V,E),c,s,t,2)$ be a simple unicast network, $\bar{\mathbb{N}}(G(V,E),\bar{c},s,t,3)$ be the corresponding flow network and b be a half-integral 3-flow in $\bar{\mathbb{N}}(G(V,E),\bar{c},s,t,3)$. Then, for every edge $e \in E$ it holds that if $c(e) = 2$ ($\bar{c}(e) = 1.5$) then $b_e = 1.5$ and if $\bar{c}(e) = 1$ then $b_e \in \{0.5, 1\}$.

Lemma 10: Let $\mathbb{N}(G(V,E),c,s,t)$ be a minimal unicast network. Then $G(V,E)$ is a directed acyclic graph.

We proceed by classifying the nodes in $V \setminus \{s,t\}$.

Lemma 11: Let $\mathbb{N}(G(V,E),c,s,t,2)$ be a simple network. Then, each node $v \in V\setminus\{s,t\}$ belong to one of the types A, B, C, or D depicted in Figure 2.

Proof: (sketch) Let $\bar{\mathbb{N}}(G(V,E),\bar{c},s,t,3)$ be the flow network that corresponds to $\mathbb{N}$. By Theorem 8, $\bar{\mathbb{N}}$ is a feasible network. Let b be a minimum cost half-integral flow of value 3 in $\bar{\mathbb{N}}$. First, consider a node $v \in V\setminus\{s,t\}$ that has one incoming edge e_1 and two outgoing edges e_2 and e_3. The flow b_{e_1} on edge e_1 is either equal to 1 or 1.5. Indeed, by Lemma 9, it holds that $b_{e_1} > 0$. If $b_1 = 0.5$, then the flow of on one of the outgoing edges e_1 or e_2 would be 0, in violation of Lemma 9. If $b_{e_1} = 1$, then it holds that $b_{e_2} = b_{e_3} = 0.5$ and, by Lemma 9, $c(e_1) = c(e_2) = c(e_3) = 1$, hence v is a node of type A. If $b_{e_1} = 1.5$, then it holds that $b_{e_2} = 1$ and $b_{e_3} = 0.5$, or $b_{e_2} = 0.5$ and $b_{e_1} = 1$. Again, by Lemma 9, $c(e_1) = 2$ and $c(e_2) = c(e_3) = 1$, hence v is a node of type C. Similarly, we prove that nodes of in-degree 2 and out-degree 1 are either of type B or D.

∎

C. Residual Graphs

Let $\mathbb{N}(G(V,E),c,s,t,2)$ be a simple network, $\bar{\mathbb{N}}(G(V,E),\bar{c},s,t,3)$ be the corresponding flow network, and b be a half-integer 3-flow in $\bar{\mathbb{N}}$. We denote by $\hat{E}$ the set of edges of capacity 1 that have 0.5 units of flow, i.e., $\hat{E} = \{e \in E \mid \bar{c}(e) = 1 \text{ and } b_e = 0.5\}$. Then we fix a subset $\hat{E}'$ of $\hat{E}$. This subset defines a *coloring* of the edges in $G(V,E)$ in the following way. All edges in $\hat{E}'$ are referred to as *red* edges. All edges in $\hat{E}\setminus\hat{E}'$ and all edges with $\bar{c} = 1.5$ are referred to as *blue* edges. All other edges in the network are referred to as *black* edges.

We proceed to discuss a concept of a residual graph. The definition of the residual graph depends on the way the edges are colored.

Definition 12 (Residual Graph, Residual Cycle): Let $\hat{E}'$ be a subset of $\hat{E}$. Then, the *residual graph* $G_{\hat{E}'}(b)$ of $G(V,E)$ is formed by reversing all blue and red edges in $G(V,E)$, where the edges are colored according to the subset $\hat{E}'$. All edges in the residual graph retain their original color. A cycle C in $G_{\hat{E}'}(b)$ is referred to as *residual cycle* if it includes at least one blue edge.

Lemma 13: If $\mathbb{N}(G(V,E),c,s,t,2)$ is a simple network, then $G_{\hat{E}'}(b)$ does not have a residual cycle.

Proof: Suppose, by way of contradiction, that there exists a cycle C in $G_{\hat{E}'}(b)$ that includes a blue edge $\hat{e}$. We now create a new flow b' by augmenting b along C. In particular, for each red edge $e \in C$ we identify the edge e' that corresponds to e in the original graph. Then, if e is a red edge we set $b'_{e'} = b_{e'} + 0.5$, otherwise we set $b'_{e'} = b_{e'} - 0.5$. It is easy to verify that b'_e is a feasible half-integer flow in $\bar{\mathbb{N}}(G(V,E),\bar{c},s,t,3)$. Let $\hat{e}'$ be the edge that corresponds to $\hat{e}$ in the original network. If $\hat{e}'$ belongs to $\hat{E}$ then it holds that $b'_{\hat{e}'} = 0$, which, by Lemma 9, contradicts the minimality of $\bar{\mathbb{N}}(G(V,E),c,s,t,2)$. Otherwise, it holds that $\bar{c}(\hat{e}') = 1.5$ and $b'_{\hat{e}'} = 1$, which again and by the same lemma contradicts the minimality of $\mathbb{N}(G(V,E),c,s,t,2)$. ∎

D. Topological Cuts

We begin by defining a notion of *topological cut*.

Definition 14 (Topological Cut): Let $G(V,E)$ be an acyclic directed graph. A cut $C = (V_1, V_2)$ is said to be a *topological cut* if there is no edge in G that connects a node in V_2 to a node in V_1.

We visit the nodes of G in topological order [12, Section 3.4]. Note G has a topological ordering because it is acyclic (by Lemma 10). The first topological cut that we encounter is $C(s, E\setminus s)$ that consists of three edges $e_1(v_1,u_1)$, $e_2(v_2,u_2)$, and $e_3(v_3,u_3)$, each of unit capacity. We refer to all topological cuts that include three edges of capacity one as cuts of type I.

For each node u_i, $1 \leq i \leq 3$, in a cut of type I, it holds that u_i is either of type A, D, or identical to a terminal node t. Clearly, a node cannot be of type C because edges e_1, e_2, and e_3 have unit capacity. We also observe that for $b_{e_i} = 1$ for $1 \leq i \leq 3$. This fact, coupled with Lemma 9 rules out the possibility that any of nodes u_1, u_2, and u_3 is of type B.

Next, we show that at most one of the nodes u_1, u_2, or u_3 is of type A. We consider two cases. First, we assume, by way of contradiction that all three nodes u_1, u_2, or u_3 are type A nodes. In this case all of the outgoing edges of u_1,

u_2, and u_3 belong to $\hat{E}$. It is easy to verify that we can pick three edges e^1, e^2, and e^3 such that for $1 \leq i \leq 3$ it holds that e^i is an outgoing edge of u_i and there does not exist a node v which is incident to any two edges e^1, e^2, and e^3. We set $\hat{E}' = \{e^1, e^2, e^3\}$. Let $G_{\hat{E}'}(b)$ be the residual graph of $G(V, E)$ with respect to $\hat{E}'$ and let G' be the subgraph of $G_{\hat{E}'}(b)$ induced by nodes in V_2 determined by the cut C. It is easy to verify that each node in G' has outdegree at least 1, hence G' contains a cycle. Such a cycle must include an edge e^i, for some $1 \leq i \leq 3$, because if we exclude edges $\{e^1, e^2, e^3\}$ from G' then the resulting graph would be acyclic. This implies, that a cycle includes at least one blue edge, which is the edge incident to u_i. We conclude that G' and, in turn, $G_{\hat{E}'}(b)$, includes a residual cycle, which, by Lemma 13 contradicts the minimality of $\bar{\mathbb{N}}(G(V, E), \bar{c}, s, t, 3)$. Similarly, we prove that we cannot have two nodes in the set $\{u_1, u_2, u_3\}$ that are of type A, and the other node is of type D.

Next, we prove that if one of the nodes $\{u_1, u_2, u_3\}$ is of type A, then it must be connected to the two other nodes which are of type D. We assume, without loss of generality, that u_1 is a type A node and u_2 and u_3 are of type D. Suppose, by way of contradiction, that u_1 is not connected to, say, u_2. Then, we denote by e the outgoing edge of u_1 which is not connected to u_3 and set $\hat{E}' = \{e\}$. Let $G_{\hat{E}'}(b)$ be the residual graph of $G(V, E)$ with respect to $\hat{E}'$ and let G' be the subgraph of $G_{\hat{E}'}(b)$ induced by nodes in V_2. It is easy to verify that each node in G' has outdegree at least 1, hence G' includes a cycle. Such a cycle must include edge e, and, in turn, one blue edge which is incident to u_1. As a result, G' and, in turn, $G_{\hat{E}'}(b)$ include a residual cycle, in contradiction to Lemma 13.

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 A.

We conclude our discussion by the following theorem.

Theorem 15: Let $\mathbb{N}(G(V, E), c, s, t, 2)$ be a simple network. Let $C = (V_1, V_2)$ be a topological cut of $G(V, E)$ such that $E(V_1, V_2)$ includes three edges $e_1(v_1, u_1)$, $e_2(v_2, u_2)$, and $e_3(v_3, u_3)$, each one of them of unit capacity. Then it either holds that $u_1 = u_2 = u_3 = t$ or it holds that one of the nodes u_1, u_2, or u_3 is of type A, while two other nodes are of type D and the node of type A is connected to two other nodes by two edges.

Thus, starting by a topological cut of type I, and if it is not the last topological cut in the network $(C(E\backslash, t)$, we will end up with a topological cut which consists of two edges of capacity 2. We refer to such cuts as Type II cuts.

Theorem 16: Let $\mathbb{N}(G(V, E), c, s, t, 2)$ be a simple network. Let $C = (V_1, V_2)$ be a topological cut of $G(V, E)$ such that $E(V_1, V_2)$ includes two edges $e_1(v_1, u_1)$, $e_2(v_2, u_2)$, each one of them of capacity of two units. Then, there exist either a type B node $w \in V_2$ and two edges (u_1, w) and (u_2, w) of capacity 1, or two type-D nodes w_1 and w_2 and four edges (u_1, w_1), (u_1, w_2), (u_2, w_1), and (u_2, w_2) of capacity 1.

E. Block Decomposition

The previous discussion leads to the following result.

Theorem 17: Let $\mathbb{N}(G(V, E), c, s, t, 2)$ be a simple unicast network with $|V| > 2$. Then, $\mathbb{N}(G(V, E), c, s, t, 2)$ can be decomposed into blocks A, B, C, as depicted in Figure 3. The blocks can be connected in an arbitrary order, subject to the following rules: (1) the first block is a block A and s is connected to all its input edges (2) the last block is a block B and the destination node t is connected to all its output edges (3) block A can be be followed by blocks B or C (4) Block B, if not the last, must be followed by block A (5) Block C must be followed by blocks B or C.

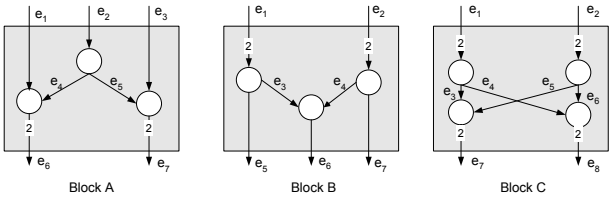

Fig. 3. Basic building blocks for a simple unicast network ($h = 2$).

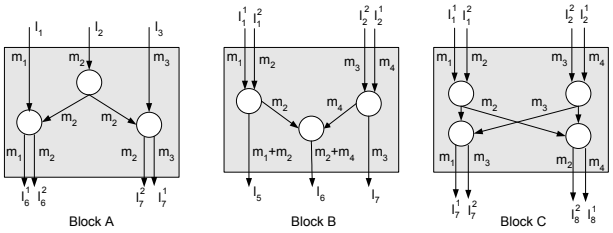

Fig. 4. Network code for simple unicast networks.

V. Network Codes for Simple Networks

In this section, we present a robust network code over $GF(2)$ for a simple unicast network $\mathbb{N}(G(V, E), c, s, t, 2)$. As we proved in the previous section, a simple unicast network consists of blocks of types A, B, and C, as depicted in Figure 3. The network code for all blocks is presented in Figure 4. In particular, the figure shows for each edge e_i of capacity one, the corresponding link l_i. For each edge e_i of capacity two the figure shows two corresponding links l_i^1 and l_i^2. We choose the notation of the links in such a way that if edge e_i of block X coincides with edge e_j of block Y then link l_i^1 of block X coincides with link l_j^1 of block Y and link l_i^2 of block X coincides with link l_j^2 of block Y.

Note that all the nodes of blocks of type A or C just forward incoming packets. In blocks of type B, one node forwards the incoming packets while other two nodes output a sum (over $GF(2)$ of their incoming packets.

To complete the definition of the encoding scheme, we need to define the encoding performed by the source node s. We denote by $a \in GF(2)$ and $b \in GF(2)$ the two packets that the source node has to transmit to the destination t at the current round. By Theorem 17, the source node is connected by three edges of capacity one to the nodes of the first block. The source node sends packets a, b, and $a + b$ on these edges, in an arbitrary order.

We proceed to prove that the network code described above is robust. Due to space constraints, we only consider the case in which the network $\mathbb{N}$ only includes blocks of type A and B.[2] Such networks are formed by the sequence of blocks A followed by a block B. We enumerate the blocks of the network by $A_1, B_1, A_2, B_2 \ldots, A_x, B_x$ such that s is

[2]Extension for general case is presented in [10]

connected A_1, A_i is connected to B_i for $1 \leq i \leq x$, B_i is connected to A_{i+1} for $1 \leq i \leq x-1$ and B_x is connected to t. We also denote by C_i, $1 \leq i \leq x-1$ the cut that separates blocks B_i from A_{i+1}. We also denote by C_0 (resp. C_x) the cut that separate the source (resp. the destination node) from the rest of the network.

We now pick an arbitrary edge $\hat{e} \in E$ and suppose that $\hat{e}$ fails. We denote by $\hat{i}$ the smallest value of i such that $\hat{e}$ belongs to block A_i or B_i.

Recall that upon a failure of the edge every link corresponding to this edge carries a packet identically equal to zero. For each link $l \in L(E)$ we denote by m_l the packet carried by link l.

Lemma 18: Consider a cut C_i, $0 \leq i < \hat{i}$. We denote by l_1, l_2, and l_3 the set of the links in this cut. Then, the packets $\{m_{l_1}, m_{l_2}, m_{l_3}\}$ carry the symbols $\{a, b, a+b\}$ in an arbitrary order.

Definition 19 (Valid Cut): Consider a cut C_i, $0 \leq i \leq x$. We denote by l_1, l_2, and l_3 the set of the links in this cut. Then C_i is said to be *valid* if at least two of the packets in the set $\{m_{l_1}, m_{l_2}, m_{l_3}\}$ are linearly independent.

Lemma 20: The cut $C_{\hat{i}}$ is a valid cut.

Proof: We depict blocks $A_{\hat{i}}$ and $B_{\hat{i}}$ on Figure 5 and show that upon the failure of any edge in $A_{\hat{i}}$ or $B_{\hat{i}}$, the cut formed by edges e_{12}, e_{13}, and e_{14} remains valid.

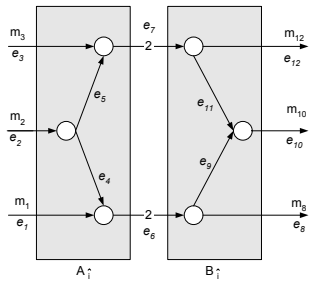

Fig. 5. Blocks $A_{\hat{i}}$ and $B_{\hat{i}}$.

Let m_1, m_2, and m_3 be the packets received on the incoming edges e_1, e_2, and e_3 of block $A_{\hat{i}}$. The following table lists the outputs of this block upon the failure of each of its edges.

failure	(m_{12}, m_{13}, m_{14})	failure	(m_{12}, m_{13}, m_{14})
ϕ	(m_3, m_1, m_2)	e_7	$(m_3, m_2, 0)$
e_1	(m_2, m_1, m_2)	e_{10}	(m_3, m_3, m_2)
e_2	$(m_1, m_3, 0)$	e_{11}	(m_3, m_2, m_2)
e_3	(m_3, m_2, m_2)	e_{12}	$(0, m_1, m_2)$
e_4	(m_1, m_3, m_2)	e_{13}	$(m_3, 0, m_2)$
e_5	$(m_3, m_1, 0)$	e_{14}	$(m_3, m_1, 0)$
e_6	$(0, m_3, m_2)$		

By Lemma 18, the packets m_1, m_2, and m_3 carry symbols $\{a, b, a+b\}$ in an arbitrary order. This implies that the cut formed by edges e_{12}, e_{13}, and e_{14} remains valid under any failure. ■

Lemma 21: Each cut C_i, $\hat{i} < i \leq x$ is a valid cut.

Proof: The proof is by induction on i. As a basis step of the induction, consider two blocks $A_{\hat{i}+1}$ and $B_{\hat{i}+1}$. We assume that the edges in $A_{\hat{i}+1}$ and $B_{\hat{i}+1}$ are denoted as shown in Figure 5. Then, it holds that

$$\begin{pmatrix} m_8 \\ m_{10} \\ m_{12} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix} \quad (1)$$

Since the matrix in Equation 1 is of full rank, and since at least two of the packets m_1, m_2, and m_3 are linearly independent, it holds that at least two of the packets m_8, m_{10}, and m_{12} are linearly independent. The inductive step follows from the same argument. ■

By Lemma 21, C_x is a valid cut. This implies that the destination node receives at least two linearly independent packets. We conclude that the destination node can always reconstruct the two original packets a and b, which, in turn, implies that the network code described above is robust.

We summarize our discussion in the following theorem.

Theorem 22: There exists a robust linear code over $GF(2)$ for any feasible network $\mathbb{N}(G(V,E), c, s, t, 2)$.

VI. Conclusion

In this paper, we addressed the problem of constructing robust network codes for unicast networks. We defined an important class of simple unicast network. We showed that, for $h = 2$, simple unicast networks have an elegant combinatorial structure. This structure enables the design of a robust linear network code over $GF(2)$ for *any* feasible network.

Many open questions, that are the topics of our current research, naturally arise here: For arbitrary h $(h > 2)$, what is the structure of feasible and minimal unicast networks? What is the minimum field size over which a robust code exists for a feasible unicast network when $h > 2$? How can these results be used to derive an efficient algorithm for finding robust network codes for unicast networks?

As a partial answer to the second open question, we conjecture that the minimum field size is linear in h, independently of the size of the underlying network.

References

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

[2] 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.

[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] 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.

[6] M. Charikar and A. Agarwal. On the Advantage of Network Coding for Improving Network Throughput. In *Proceedings of IEEE Information Theory Workshop, San Antonio*, 2004.

[7] T. Ho, M. Médard, and R. Koetter. An Information-Theoretic View of Network Management. *IEEE Transactions on Information Theory*, 51(4), April 2005.

[8] Y. Wu, P. A. Chou, and K. Jain. A Comparison of Network Coding and Tree Packing. In *Proceedings of ISIT*, 2004.

[9] 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.

[10] S. Y. El Rouayheb, A. Sprintson, and C. Georghiades. Simple Network Codes for Instantaneous Recovery from Edge Failures in Unicast Connections. Technical Report WCL-TR-06-101, Texas A&M University, College Staion, Texas, January 2006.

[11] K. Menger. Zur allgemeinen Kurventheorie. *Fund. Math*, 10:95–115, 1927.

[12] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Networks Flows*. Prentice-Hall, NJ, USA, 1993.