Max-Flow Protection using Network Coding

Osameh M. Al-Kofahi
Department of Computer Engineering
Yarmouk University, Irbid, Jordan

Ahmed E. Kamal
Department of Electrical and Computer Engineering
Iowa State University, Ames, IA 50010

***Abstract*—In this paper we present a new way to enhance the survivability of the information flow between two communicating nodes S and T without compromising the maximum achievable S-T information rate. To do this, bottleneck links should only forward useful information, and not redundant data units. We introduce the idea of extra source or destination connectivity with respect to a certain S-T max-flow, and then we introduce two problems: namely, pre-cut protection and post-cut protection. Because of space limitations we only focus on the pre-cut protection problem. Specifically, we show that the pre-cut protection problem is NP-hard, we propose a heuristic approach to solve it, and we compare the performance of this heuristic to an ILP. Simulations show that the performance of the heuristic is acceptable even on relatively large networks.**

I. Introduction

The survivability of an information flow between two terminal nodes, S and T, can be enhanced by using part of the available network resources (bandwidth) to forward redundant information from S to T. Depending on the used survivability mechanism, the redundant information can be used to recover from data corruption or failures of network components. Traditionally, to protect a data unit against q link failures, $q+1$ edge-disjoint S-T paths are used to forward $q+1$ copies of the data unit from S to T. This is usually accomplished by means of a multipath routing protocol, such as MDVA [1] or AOMDV [2]. Let h denote the value of the S-T min-cut. Then, if we want to forward data units from S to T and protect them against q failures, we cannot send more than $k = \lfloor \frac{h}{q+1} \rfloor$ data units since $q+1$ copies of each data unit should be forwarded.

It is clear that traditional proactive protection approaches are very demanding and waste a lot of resources. Even if $q = 1$ the useful S-T information rate will be reduced by at least 50%. Network coding [3] can be used to overcome this problem. The basic idea of network coding is that it allows intermediate network nodes to generate combinations from the original data units, instead of just forwarding them as is. Therefore, to forward data units from S to T and protect them against q failures, we can send at most $k = h-q$ data units. This is done by designing a network code that creates $k+q$ combinations at intermediate network nodes such that any k of them are solvable. That is, it is enough to receive only k combinations to recover the k data units at T. This simple analysis shows that the useful information rate of network coding-based protection is better than that of traditional protection approaches as long as $h > q+1$, which is usually the case. Examples of network coding-based protection can be found in [4], [5], [6], [7].

Network coding-based protection and traditional protection schemes, provide end-to-end protection of the whole S-T paths used to forward useful data from S to T. In these approaches, the more we enhance the S-T flow survivability, the more we reduce the useful S-T information rate. This is because such approaches treat all network links equally, i.e., bottleneck and non-bottleneck links are used to forward redundant data. Usually, most of the links in a network are not bottleneck links, which means that link failures are more likely to affect non-bottleneck links than links in the min-cut. Therefore, we can enhance the survivability of the S-T information flow without reducing the useful S-T rate below the max-flow, if we provide protection to the non-bottleneck links only. We call this kind of protection *Max-flow protection* because the max-flow can still be achieved under these conditions as long as no link in the min-cut fails. To the best of our knowledge the problem of max-flow protection has not been studied before.

The rest of this paper is organized as follows. Section II presents the terminology and definitions that will be used throughout the paper. The problems of pre-cut and post-cut protection are presented in Section III. In Section IV we study the pre-cut protection problem. A 3-phase heuristic approach to solve the pre-cut protection problem is described and evaluated in Section V. Section VI describes a simple network code to combine the data units to be sent from S to T. Finally, Section VII concludes the paper.

II. Preliminaries

We represent a network by a directed acyclic graph G=(V,E), where V is the set of network nodes and E is the set of available links, where each link is assumed to have unit capacity. The network has a source node (S) that wants to send data to a destination (T), where the S-T max-flow is assumed to be h. We assume that a multipath routing protocol is used, e.g., [1] or [2], and the source is fully utilizing the available paths by sending h data units to the destination simultaneously. To simplify the analysis, we assume that the network has a single cut [1]. In the rest of this section we define the meaning of extra connectivity with respect to the S-T max-flow. After that we discuss some of the properties of nodes with extra connectivity.

This work was supported in part by the National Science Foundation under grants CNS-0626822, CNS-0626741, CNS-0721453, ECCS-0926029 and ECS-0601570 and by a gift from Cisco Systems.

[1] In a future study, we plan to extend this work to the more general case of multiple min-cuts, which is a much harder case.

A. Terminology

Let $f^{(A)}(B)$ denote the max-flow from the nodes in set A to the nodes in set B on a directed graph, which can be calculated by computing the max-flow between a virtual source/sink pair, such that the virtual source is connected to the nodes in A with infinite capacity edges and the virtual sink is connected to the nodes in B with infinite capacity edges. Using this notation, the max-flow $h = f^S(T)$. We also define the following:

1) A node with Extra Source Connectivity (wESC) is a node, u, such that, $f^S(u,T) > h$, and $f^{(S,u)}(T) = h$.
2) A node with Extra Destination Connectivity (wEDC) is a node, v, such that, $f^S(v,T) = h$, and $f^{(S,v)}(T) > h$.
3) A node with No Extra Connectivity (wNEC) has $f^S(v,T) = f^{(S,v)}(T) = h$.

Of course, a node with both extra source and extra destination connectivity cannot exist, because this contradicts the assumption that the max-flow equals h. Consider the graph G in Figure 2. The S-T max-flow in G is 4, which implies that four data units can be forwarded from S to T on four link-disjoint paths. Assume we found the following paths, $P_1 = \{S \rightarrow A \rightarrow E \rightarrow J \rightarrow T\}$ that forwards data unit w, $P_2 = \{S \rightarrow B \rightarrow F \rightarrow G \rightarrow T\}$ that forwards data unit x, $P_3 = \{S \rightarrow F \rightarrow H \rightarrow T\}$ that forwards data unit y, and $P_4 = \{S \rightarrow D \rightarrow I \rightarrow T\}$ that forwards data unit z. Each path P_i contains a cutting edge C_i, which , if deleted, will result in reducing the max-flow by exactly 1 unit of flow because path P_i will be disconnected and cannot be reestablished in any way. In our example, P_1 contains $C_1 = \{(J,T)\}$, P_2 contains $C_2 = \{(G,T)\}$, P_3 contains $C_3 = \{(F,H)\}$, and P_4 contains $C_4 = \{(I,T)\}$. Note that the min-cut may not always be unique, but in this paper we assume that the graph under consideration has only one cut.

B. Properties of nodes wESC/wEDC

In general, removing the min-cut edges (i.e., the edges in $\cup_{i=1}^{h} C_i$) partitions the network into two partitions A and A', such that $S \in A$ and $T \in A'$. Note that, after deleting the min-cut edges, each of the partitions A and A' is a connected component (at least weakly), and that partition A contains nodes wESC, but partition A' contains nodes wEDC.

Lemma 1. *Any node $u \in A, u \neq S$ is a node wESC.*

Proof: We prove this by contradiction. Let $u \in A, u \neq S$, but u is not a node wESC. Then, $f^S(u,T) = h$, which means that node u cannot receive additional flow from S if the S-T max-flow is established. This implies that either node u is behind the min-cut (i.e., $u \in A'$), which contradicts the starting assumptions, or that there is another min-cut between S and u, which contradicts the single min-cut assumption. ■

In a similar fashion, we can prove the following for any node $v \in A', v \neq T$.

Lemma 2. *Any node $v \in A', v \neq T$ is a node wEDC.*

In our following discussion we refer to A as the *pre-cut* portion of the network, and to A' as the *post-cut* portion of the network. Figure 1 summarizes the previous discussion.

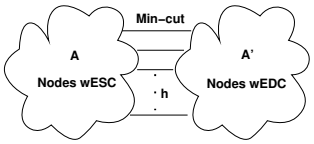


Fig. 1. Nodes wESC, wNEC and wEDC with respect to min-cuts

III. Problem description

The cutting-edges, cannot be protected unless we trade bandwidth for survivability (i.e., unless we use an S-T path to carry redundant information to the destination), which reduces the useful S-T information rate. This tradeoff not only protects the cutting-edges, but also protects any edge carrying data in the network. However, the non-cutting-edges (or a subset of them) can be protected without reducing the S-T information rate, if the graph contains nodes wESC and/or wEDC. For example, nodes E, F, I and J in Figure 2 are nodes wESC, and node H is a node wEDC. There are four possible ways to utilize the extra source connectivity in Figure 2; 1) protect data units x and y by sending $x+y$ to F through C, 2) protect w by sending a duplicate to E through C and F, 3) protect w by sending a duplicate to J through C, F, E and G 4) protect z by sending a duplicate to I through C and F. The first option is better than the other three since sending $x+y$ to F enhances the chances of two data units (x and y) to reach T, compared to duplicating w or z alone, which protects a single data unit only. Figure 3 shows the first option, and it also shows how to utilize the extra destination connectivity from node H, where H sends a duplicate of y to T through node K.

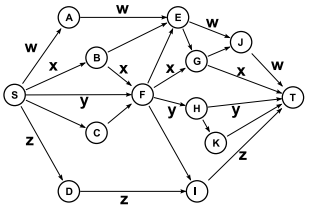


Fig. 2. Graph G with S-T max-flow = 4

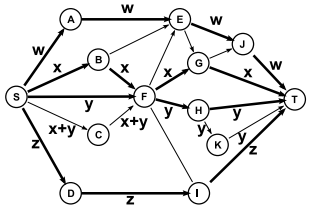


Fig. 3. Utilizing extra connectivity

In this work, we propose a different way to handle the "survivability vs. bandwidth" trade-off. We propose a new approach to provide protection to the S-T information flow without reducing the useful S-T data rate. Basically, we avoid protecting the bottlenecks in the network (the min-cut links), and we try to efficiently utilize (by using network coding if possible) the available network connectivity before and/or after the bottleneck to provide protection to the non-min-cut links in the graph. We divide the problem into two sub-problems as follows:

1) Pre-cut protection: Our objective is to maximize the number of pre-cut-protected S-T paths. We show that this problem is NP-hard, and we provide a heuristic to solve it. To evaluate our heuristic we compare its performance to an ILP.
2) Post-cut protection: We aim to maximize the number of post-cut-protected S-T paths. Let e_i be the closest cutting edge to the destination T on path P_i. We show

in [8] that all the paths that do not have T as the head node of e_i , where $1 \leq i \leq h$, can be post-cut-protected together against at least one failure. Because of space limitations, we do not provide any further discussion of this problem in this paper.

In this paper we discuss the pre-cut protection problem, and provide a general description of the used heuristic. A detailed discussion of the heuristic and the ILP can be found in [8].

IV. Pre-Cut: Nodes with Extra Source Connectivity

As discussed in Section II, all nodes wESC are located in the pre-cut portion of the network. Assume that the set $\mathcal{X}$ contains all the nodes wESC, $\mathcal{X} = A \backslash S = \{u_1, u_2, \ldots, u_{|\mathcal{X}|}\}$. Then, the following is true:

$$(\sum_{i=1}^{|\mathcal{X}|} f^S(u_i, T)) - |\mathcal{X}| f^S(T) \geq f^S(\mathcal{X}, T) - f^S(T) \quad (1)$$

This is because the extra source connectivity may be shared between the nodes in $\mathcal{X}$. Therefore, the right hand side of the inequality is what really determines the available extra source connectivity (ESC). This implies that not all nodes wESC in $\mathcal{X}$ can receive redundant flows from S to be used to protect the S-T max-flow, and thus, a subset $X \subseteq \mathcal{X}$ should be intelligently selected to receive the available extra source flow and utilize it in the best way possible. Note that the number of nodes in X cannot exceed the extra available connectivity, i.e.:

$$ESC = f^S(u_1, u_2, \ldots, u_{|\mathcal{X}|}, T) - f^S(T) \geq |X| \quad (2)$$

The selection of X depends on how the S-T max-flow is routed on the graph. Consider the graph in Figures 4(a) and 4(b), the S-T max-flow in this network is 2, and there is only one S-T min-cut in the graph, which contains the edges (A,T) and (C,T). Nodes A, B and C are nodes wESC, and the total available extra source connectivity equals $f^S(A, B, C, T) - f^S(T) = 4 - 2 = 2$. Assume that the max-flow is routed as shown in Figure 4(a) (the dashed lines), in this case $X_1 = \{B, C\}$ since the extra source connectivity is consumed by B and C. Moreover, note that only the path forwarding **b** can be pre-cut-protected by sending copies of **b** on (S, B) and (S, C). Now consider the routing shown in Figure 4(b), in this case $X_2 = \{A, C\}$. Unlike the previous case, both paths can be pre-cut-protected by sending a second copy of **a** to A, and a second copy of **b** to C through B. Obviously, the second routing option is better since it allows the protection of both paths (equivalently both data units), in this sense we say X_2 is better than X_1.

It was shown in the previous example that routing the max-flow and selecting X are inseparable problems, and that routing the S-T max-flow corresponds to selecting X. Let us define the extra source connectivity to a node u with respect to the routing of the S-T max-flow in the network as:

$$EC(u) = f^S(u, T) - f^S(T)$$

We say that an S-T path is pre-cut-protected if a segment of this path in the pre-cut portion of the network is protected. That is, a path is pre-cut-protected if it contains a node wESC

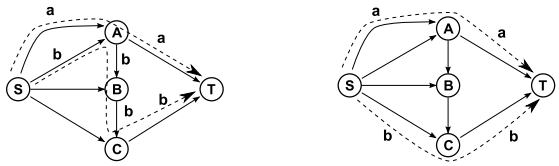


Fig. 4. Routing the max-flow is what determines X. In (a) $X = \{B, C\}$, and one path is protected. In (b) $X = \{A, C\}$, and both paths are protected.

with respect to the routing of the S-T max-flow. *Therefore, maximizing the number of pre-cut-protected paths means maximizing the number of paths containing nodes wESC.*

For large networks, trying-out all possible routing choices to find the best one that will maximize the number of paths containing nodes wESC is computationally expensive. The following theorem proves that this problem is in fact an NP-hard problem. The full-proof can be found in [8], it is omitted here due to space limitations.

Theorem 1. *Routing the S-T max-flow to maximize the number of S-T paths containing nodes wESC is NP-hard.*

Note that if network coding was not allowed, then from equation (2) we cannot protect more than ESC data units. Therefore, to utilize the extra source connectivity in a more efficient manner we should apply network coding whenever possible. Network coding can be used if a node wESC, say u, lies on more than one S-T path, and has $EC(u) \geq 1$. For example, let u be a node wESC that lies on two S-T paths, and that has $EC(u) = 1$. A network code can be designed to deliver three combinations in two data units to u, such that any two combinations are solvable, i.e., two data units are protected from S to u against a single link failure. Note that the number of failures that can be tolerated is at most $EC(u)$. Therefore, the nodes in X should have the following properties:

1) Each node $u_i \in X$ must have $f^S(u_i) > f^{u_i}(T)$.
2) The combinations received by a node $u_i \in X$ must be solvable if at most $e = f^S(u_i) - f^{u_i}(T)$ failures occurred on the $f^S(u_i)$ paths from S to u_i.

The first condition requires the flow from the source to each node $u_i \in X$ to be larger than the flow from that node to the destination. This condition is necessary to introduce redundancy in the forwarding process from S to the nodes in X. The second condition can be satisfied by designing a network code that delivers, for each node u_i, a set of $f^S(u_i)$ combinations, such that any $f^{u_i}(T)$ combinations of them are solvable. These two conditions allow a node u_i to act as pre-cut decoding node, which can recover the data units sent from S to T through u_i, if at most $e = f^S(u_i) - f^{u_i}(T)$ failures occurred on the S-u_i link-disjoint paths, and then send these native data units to T.

In the next section we describe a heuristic approach to solve the pre-cut protection problem.

V. Heuristic approach

Our heuristic works in three phases; the first one greedily selects an initial set X'; the second one modifies the flow on the graph (if needed) to guarantee that the S-T max-flow is achieved, and the third one utilizes any remaining connectivity and produces the final set X. The first phase works in iterations, where a single node is added to X' in each iteration. Each time we add the node that can send the most flow to the destination, while being able to receive more flow from the source, to satisfy the two conditions stated at the end of Section IV. If no more nodes satisfy this criteria and the S-T flow is still less than h, the second phase is entered. The second phase finds as much augmenting paths as possible from S to T so that the S-T max-flow is maximized. Finally, the third phase checks the nodes in the pre-cut portion of the graph to see if there are any remaining nodes wESC, and makes use of this extra connectivity. A detailed discussion of the three phases of the heuristic can be found in [8].

Recall that if all the min-cut edges are deleted, then the graph will be divided into two partitions A (pre-cut), and A' (post-cut). Note that the routing of the S-T flow in the post-cut portion of the network is independent from the routing of the S-T flow in the pre-cut portion of the network. Therefore, we use a simplified version of the original graph in our heuristic, which is described in Algorithm 1. We simplify the graph under consideration and just focus on the sub-graph, H, induced by the nodes in A with a little modification. Specifically, given a directed graph $G(V_G, E_G)$, let F_S be the set of tail nodes on the min-cut edges. We transform graph G to $H(V_H, E_H)$ as follows:

1) Delete the nodes in $\{V_G \backslash A\}$
2) $V_H = \{A, T'\}$, where T' is a dummy destination node.
3) $E_H = \{(u,v)|(u,v) \in E_G\} \bigcup \{(u,T')|\forall u \in F_S\}$. Note that $\{S, F_S\} \subset A$.

A. Evaluation

In this section we compare the results from our heuristic to the results from the ILP presented in [8]. The heuristic was compared to the ILP in five different cases. Each case represents a different network size, where the number of network nodes V was changed to take the values $\{5, 10, 15, 20, 25\}$. In each case eighty random network instances were generated, and fed to the heuristic and the ILP.

To gain a better insight on the operation of the heuristic compared to the ILP we measured the S-T max-flow, counted the number of pre-cut-protected paths from the heuristic, and the number of pre-cut-protected paths resulting from the ILP in each time the heuristic and the ILP were executed (on the same network instance).

The histograms for the cases of V = 10, 15, 20, and 25 are shown in Figures 5(a), 5(b), 5(c), and 5(d) respectively. In general, the results from the heuristic are close to those from the ILP. Note that in some cases, the number of times the heuristic is able to protect X_1 paths may be larger than the number of times the ILP is able to protect the same number of paths X_1. However, this does not invalidate the heuristic because it comes at the price of protecting a larger number of paths $X_2 > X_1$ a fewer number of times. For example, in Figure 5(c), the heuristic was able to protect $X_1 = 2$ paths more than the ILP, but the ILP was able to protect $X_2 = 4$ paths more than the heuristic.

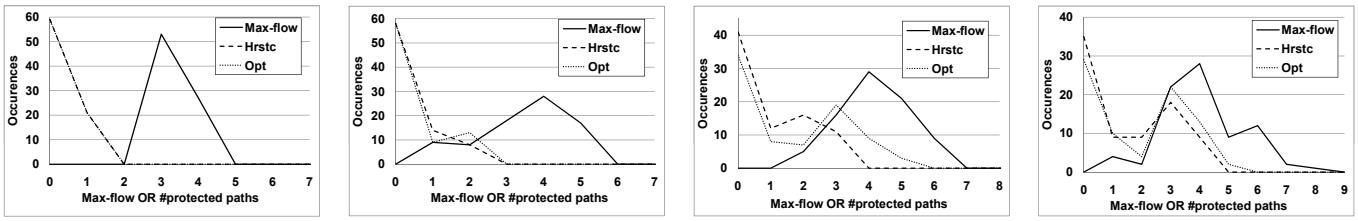
Algorithm 1 Selecting set X

Input: Graph $H(V_H, E_H)$, h = S-T max-flow
Output: Set X containing nodes wESC
$X' = \phi$, $ST\_flow = 0$, $Phase\_done = 0$
Create matrices $Flow_S[V_H]$, $Flow_T[V_H]$ //One dimensional matrices initialized to all zeros, to store the final flow from S to each node in X, and from each node in X to T'. This information will be used for coding later
//Phase 1
Create graphs H^S and H^T, where $V_{HS} = V_{HT} = V_H$ and $E_{HS} = E_{HT} = E_H$.
while $(Phase\_done == 0)$ **do**
  Compute $f^S(u)$ on graph H^S, $\forall u \in V_{HS}$
  Compute $f^u(T')$ on graph H^T, $\forall u \in V_{HT}$
  Select node x, where $f^x(T') \geq f^u(T')$, $\forall u \in V_H$, and $f^S(x) > f^x(T')$
  if (No such node exists) **then**
    $Phase\_done = 1$
  else
    Find $f^x(T') + 1$ augmenting paths from S to x on H^S
    Delete all forward edges in H^T if they are reversed in H^S //due to augmentation
    Find $f^x(T')$ augmenting paths from x to T'
    Delete all forward edges in H^S if the are reversed in H^T
    $X' = X' \cup \{x\}$
    $ST\_flow = ST\_flow + f^x(T')$
    $Flow_S[x] = f^x(T') + 1$
    $Flow_T[x] = f^x(T')$
  end if
end while
for all $((u,v) \in E_H)$ **do**
  if $((v,u) \in E_{HS} || (v,u) \in E_{HT})$ **then**
    Reverse (u,v) in H
  end if
end for
$Phase\_done = 0$ //End of Phase 1, and beginning of Phase 2
while $(Phase\_done == 0)$ **do**
  if $(ST\_flow = h)$ **then**
    $Phase\_done = 1$
  else
    Find an S-T' augmenting path in H
    ST_flow++
  end if
end while
$Phase\_done = 0$ //End of Phase 2, and beginning of Phase 3
for all $(u \in V_H)$ **do**
  Compute $p = f^S(u)$ on the current residual graph of H
  if $(f^S(u) > 0)$ **then**
    Find p augmenting paths from S to u on H
    $Flow_S[u] = Flow_S[u] + p$
  end if
  if $(u \notin X')$ **then**
    Compute $q = f^{T'}(u)$ on H^T
    $Flow_T[u] = Flow_T[u] + q$
    $X' = X' \cup \{u\}$
  end if
end for
return X'

VI. Coding

The resulting S-T' flow from the heuristic can be decomposed into two parts; the first, a one-to-many flow from S to the nodes in X, and the second is a many-to-one flow from S and the nodes in X to T'. The many-to-one flow is not and cannot be coded, since it is composed from the h native data units that are forwarded from S and the nodes in X (possibly after decoding) to T', on h disjoint paths. However, the one-to-many flow from S to the nodes in X can, and should be coded to utilize the extra source connectivity in the most efficient manner. Note that this one-to-many flow is different from normal multicast flow since different data is sent to different nodes. Therefore, a standard multicast network code cannot be used. In fact, the coding in our case is simpler, and needs to be done at a limited number of network nodes as we will show in the following discussion.

After the heuristic is done and the flow is constructed in the pre-cut portion of the graph. A node $u \in X$ can receive $k + e = Flow_S[u]$ units of flow from S and can send $k =$



(a) (b) (c) (d)

Fig. 5. All figures are histograms, which count three different frequencies: the max-flow, the number of protected paths from the heuristic and the number of protected paths from the ILP. (a) has $V = 10$, (b) has $V = 15$, (c) has $V = 20$ and (d) has $V = 25$. The x axis is the number paths either protected or counted in the max-flow, and the y axis is the number of times each number of paths occurred as a max-flow or protected by the ILP or the Heuristic

$Flow_T[u]$ units of flow to T' (these values were computed in the heuristic). This implies that there are $k + e$ edge-disjoint paths from S to u, and k edge disjoint paths from u to T' (or equivalently to T). Note that k represents the number of S-T paths (or data units) going through node u, and that e represents the paths used to carry redundant information to u.

Let N_{x_i} be the set of 1-hop neighbors of the source on all the $k + e$ paths from S to x_i. Assume that all the nodes in N_{x_i} have received the same set of k data units from the source (the k data units on the k S-T paths). To construct a network code that delivers $k + e$ combinations to x_i such that any k of them are solvable using the received data units, we need to assign the proper coding vectors to the nodes in N_{x_i}. The coding vectors can be assigned from an $k \times (k+e)$ matrix that has no singular $k \times k$ submatrices, i.e., any $k \times k$ submatrix is invertible. A class of matrices that satisfies this requirement is the Cauchy matrices [10]. Therefore, we can simply assign to each node in N_{x_i} a column from a $k \times (k+e)$ Cauchy matrix, such that no two nodes are assigned the same column.

However, such a coding scheme requires decoding at the nodes in X in each transmission round. An alternative way that will require a fewer number of decoding operations would be to use a systematic code. In a systematic code, k out of the $k + e$ combinations will be trivial combinations, where each of which carries one of the k native data units. In this case, decoding is necessary at a node $x_i \in X$, only if one of the native data units was lost due to a failure on one of the k S-T paths going through node x_i. A simple way to do this is presented in [11]. Basically, let $\mathcal{M}_i$ denote a $k \times (k + e)$ Cauchy matrix with columns representing the coding vectors of the nodes in N_{x_i}. We can view $\mathcal{M}_i$ as two side-by-side matrices $\mathcal{M}_i = (\mathcal{M}_{k_i}|\mathcal{M}_{e_i})$, where $\mathcal{M}_{k_i}$ is a $k \times k$ matrix , and $\mathcal{M}_{e_i}$ is a $k \times e$ matrix. Let $\mathcal{M}'_i$ be the $k \times (k+e)$ matrix resulting from multiplying $\mathcal{M}_{k_i}^{-1}$ by $\mathcal{M}_i$:

$$\mathcal{M}'_i = \mathcal{M}_{k_i}^{-1} \times \mathcal{M}_i = (I_k|\mathcal{M}_{k_i}^{-1} \times \mathcal{M}_{e_i}) = (I_k|\mathcal{M}'_{e_i})$$

Since the original matrix $\mathcal{M}_i$ has no singular submatrices, then the resulting matrix $\mathcal{M}'_i$ has no singular submatrices also. Note that although the non-singularity property is preserved, the matrix is no longer a Cauchy matrix. Therefore, given that the source has already transmitted the k data units to the nodes in N_{x_i}, assigning the columns of $\mathcal{M}'_i$ to the nodes in N_{x_i} will create $k + e$ combinations such that any k of them are solvable. Moreover, the code is systematic, where out of the $k+e$ combinations there are k trivial combinations (from I_k), each of which is composed of a single native data unit.

A special case is when $e = 1$. In this case, after the source finishes transmitting the k data units to the nodes in N_{x_i} (where $|N_{x_i}| = k + 1$), one of the nodes in N_{x_i} can sum all the received data units and send this sum along with the k native data units on $k + 1$ paths to x_i.

VII. Conclusions

We presented a new protection approach, called max-flow protection, which can enhance the survivability of the S-T information flow without compromising the maximum achievable information rate (the S-T max-flow). The basic idea is not to protect links in the min-cut, but try to protect all other links if possible. We divided the problem into two problems; pre-cut protection and post-cut protection. Pre-cut protection is NP-hard. Therefore, we proposed a heuristic to solve it and we compared the performance of the heuristic to the optimal performance of an ILP. Finally, a simple network code is proposed to maximize the number of pre-cut protected paths.

References

[1] S. Vutukury and J.J. Garcia Luna-Aceves. Mdva: A distance-vector multipath routing protocol. In Proceedings of the INFOCOM, 2001.
[2] M. K. Marina and S. R. Das. On-demand multipath distance vector routing in ad hoc networks. In Proceedings of ICNP 2001.
[3] R. Ahlswede, N. Cai, S. R. Li, and R. Yeung. Network information flow. IEEE Trans on. Info. Thry. Vol 46, No. 4, July 2000.
[4] A. E. Kamal. 1+n network protection for mesh networks: Network coding-based protection using p-cycles. To appear in IEEE/ACM Trans on. Net.
[5] O. M. Al-Kofahi and A. E. Kamal. Network coding-based protection of many-to-one wireless flows. IEEE JSAC, VOL. 27, NO. 5, JUNE 2009.
[6] O. Al-Kofahi and A. Kamal. Scalable redundancy for sensors-to-sink communication. In the Proceeding of Globecom 2008.
[7] A. Sprintson, S.Y.E. Rouayheb, and C.N Georghiades. Robust network coding for bidirected networks. Info. Thry. and Apps. Workshop, 2007.
[8] O. M. Al-Kofahi and A. E. Kamal. Max-flow protection using network coding. http://arxiv.org/abs/0908.0722v1.
[9] C. Chekuri and A. Kumar. Maximum coverage problem with group budget constraints and applications. In proceedings of APPROX 2004, Lecture Notes in Computer Science 3122, 7283.
[10] F. J. MacWilliams and N. J. A. Sloane. The theory of error-correcting codes. North Holland, 1977.
[11] J. Lacan and J. Fimes. A construction of matrices with no singular square submatrices. In proceedings of the 7th International Conference on Finite Fields and Applications, May 2003.