Hybrid Network-Erasure Coding Protection of Multi-Source, Multi-Sink Multicast Sessions in WSNs

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

Abstract— **In this paper, we consider the problem of providing fault tolerant operation for multicast networks with multiple sources, e.g., sensors, and delivering data to a pre-defined set of destinations, e.g., sinks. We propose an algorithm that combines network and erasure coding to provide resilience against a predefined number of link failures. For sources unable to meet the flow constraints, support is provided at the cost of reduced throughput and if required, reduced protection. The necessary and sufficient flow conditions for network resource verification are proved. We also prove that the field size for coding coefficients is bounded by the amount of protection offered for erasure coding, and equal to 2 for network coding.**

I. INTRODUCTION

Information theoretic techniques are widely used to implement Forward Error Correction (FEC) strategies, which include the general class of erasure codes that is used in binary erasure channels to recover from errors or losses. Erasure coding scales a small message of k symbols into a larger one of say n symbols such that a subset of k symbols is adequate to recover the original message. Authors in [2] applied this scheme for fault tolerance in multicast networks where n disjoint multicast trees carry packets containing the original and redundant data. Recovery of original encoded data at receivers is effected through reception of a minimum of k out of the n transmitted packets, enabling recovery from failures of at most n-k paths.

Network coding, which was introduced in [1], is being increasingly employed to provide protection. The authors in [3] established the algebraic framework for linear network coding discussing recovery based on prior knowledge of edge failure patterns. Polynomial time construction of network codes that could handle such failures was discussed in [4]. In [5], an information theoretic framework for network management in the presence of edge failures was presented. Authors in [6] employed network coding to protect against node failures.

While network coding for a single source multicast session has been well-researched, results on multi-source multicast sessions have been very limited. For example, reference [7] proposed a constructive scheme that provided achievability theorems for multi-source multicast sessions. The authors in [8] partition the graph into sub graphs and transform the multi-source problem into a combinatorial optimization problem.

Network coding has been sparingly exploited to provide protection to multi-source multicast sessions, which incidentally is gaining prominence with the increase in the density and multitude of online real time applications involving group communication. Consider, as an example, a wireless sensor network environment where a group of sensors needing to send back data to a set of sinks in a sensor network, or the sinks sending commands to all sensors. Another example is many users in a dense environment, who are receiving the same content from multiple sources, such as users of a social network, or multi-user online gaming. If the communication links are vulnerable, and may fail, it is important that provisions be made in order to guarantee the delivery of information even if there are such failures. We propose a hybrid erasure coding + network coding (EC+NC) scheme to provide protection against such failures in an agile, and resource efficient manner. Using an erasure coding approach, e.g., Reed-Solomon codes, Regenerating codes, etc, n shares of a message are generated such that any k of them are sufficient to recover the message. Additionally the n-k shares from each source are combined at intermediate nodes using the technique of network coding in order to reduce the cost of transmitting redundancies. We prove that the field size for the erasure coding strategy is limited by the amount of protection (n-k) that is required, and for network coding, coefficient selection from a binary field will suffice.

The rest of this paper is organized as follows: In Section II we discuss the problem formulation followed by the flow constraints that need to be satisfied by participating nodes in Section III. The algorithm for verification of these constraints, along with the algorithms for implementation of the scheme, is discussed in Section IV. Section V discusses the coding strategy and proofs of field bounds for network and erasure coding. This is followed by simulation results and conclusion in Section VI and VII, respectively.

II. PROBLEM FORMULATION

A. Problem Statement

The multiple source multicast fault tolerance problem can be formally defined as follows:
The network is represented by the undirected graph $G(V,E)$, with V being the set of nodes and E being a set of undirected edges, with N being the number of nodes ($|V|$). Each edge is assumed to have a unit capacity. There are s sources and r receivers, with every receiver requesting data from every source. We define multicast flow (M^F) through one multicast tree as the sum of the flows received by all the destinations, with each receiver receiving equal flow. So, if there are r destinations in one multicast tree (made of unit edges), the multicast flow (a flow of one is received by each destination from the source) is r. We would like to provision a set of n multicast session connection paths between every source and each of the receivers such that

- Every receiver receives at least *k* shares on *k* disjoint paths from each source.
- Every receiver receives an additional *n-k* shares on *n-k* paths, which can be shared with other sources, and therefore shares can be linearly combined with shares from multiple sources.
- At most *n-k* paths between any source and any receiver may fail at any point of time.
- Protection is provided against failures, where each receiver will be able to recover *k* shares from each source, and hence recover all data.
- Minimum cost is required for provisioning sessions, and is defined in terms of the number of edges used
- In the event of the network being unable to support all sources, a maximum number of sources are supported at reduced throughput (exceeding K^{th}), and if necessary reduced protection, where, K^{th} is the minimum threshold throughput required by each source

B. *Solution Approach*

The solution involves constructing two sets of multicast trees. The first set consists of $s*k$ multicast trees (*k* trees from each source) disjoint from each other transmitting the data shares to the destinations for guaranteeing the requested throughput. The second set consisting of *n-k* multicast trees which are shared amongst various source sessions. The data on these trees are coded and aid in recovery at the receiver, hence supporting a failure of up to *n-k* trees. The steps followed in the solution approach can be briefly described as follows:

1. Validation of necessary and sufficient flow constraints
2. Establishing actual multicast trees (both disjoint and shared) for sources satisfying the conditions
3. Throughput and protection compromise approach for sources not meeting the constraints.
4. Determining the coefficients for erasure and network coding. These coefficients will be exchanged between the sources and receivers, and are encoded in the metadata of the packets

To further elaborate on the third step above, depending on the network graph, some sources that cannot satisfy the requirement of *n* paths (*k* disjoint and *n-k* shared paths), and several approaches can be taken, which include:

- Throughput only compromise: Compromising the throughput (*k*) to provide the guaranteed protection (*n-k*) while not going below a source specified minimum threshold throughput, K^{th}.
- Protection only compromise: Compromising protection *(n-k)* by reducing *n* (and keeping *k* constant) and decreasing the number of shared trees (*n-k*) of which the source is a part.
- Joint Throughput and Protection compromise: A combination of both approaches which involves reducing the throughput (*k*) and also protection (reducing *n-k*). A reduced protection is provided only if the source cannot meet the constraints in spite of throughput reduction.

The various symbols used in this paper are listed in Table 1.

Table 1. Network Symbols

Symbol	***Definition***
N	*Total number of nodes in the graph*
s	*Total number of sources*
r	*Total number of sinks*
S_i	*The* i^{th} *source node*
S,S`	*Set of all the sources*
t	*Used to denote a sink*
T, T`	*Collection of all the sinks*
T_j	*The* j^{th} *sink node*
n	*The total number of shares*
k	*The number of shares required to decode the data*
$f_{in}(x)$	*The flow incoming to node x from sources in S*
$f_{out}(x)$	*The flow outgoing from node x to all nodes in T*
$f_{max}(x,y)$	*The maximum flow from node x to node y*
$f_m^r(S_i,T)$	*Multicast flow from* S_i *to each of the destinations in T.*
U_x	*Set containing sources with a protection of n-k-x*
n_i	*Number of total paths for a source i*
k_i	*Number of disjoint paths for a source i*
$G^d_{Si,T}$	*Multicast Tree (disjoint) from* i^{th} *source to T*
$G^h_{S,T}$	*Multicast Tree (shared) from source set S to destination set T*
$f_p(x,y)$	*Flow path p of a tree from source x to destination y*
$f^w_G(S_i,T_j)$	*Flow path from* S_i *to* T_j *in the graph G .The superscript w can take two values:* $w \leftarrow d$ *, context of disjoint trees* $w \leftarrow h$*, context of shared trees*
f(S,t)	*Flow on a tree from source set S to destination t*
d, h	*Quantities with superscripts of d and h imply the meaning of the term in the context of disjoint and shared trees respectively*
e_m	*edge of a flow or a tree*

III. Flow Constraints

A. *Necessary Conditions*

The below flow theorems are necessary and need to be satisfied by the nodes to become a part of the scheme. The theorems are proved by assuming the existence of *k* disjoint trees and *n-k* shared trees and finding the necessary conditions at the nodes.

Theorem 1:
The following three conditions are necessary:

a) $f_{in}(t) \geq k*s+ n- k , \quad \forall t \in T$

b) $f_{out}(S_i) \geq n , \quad \forall S_i \in S$

c) $f_{max}(S_i, t) \geq n , \quad \forall S_i \in S , \ \forall t \in T$

Sketch of Proof:
a) We need k disjoint trees between every source and destination in addition to n-k trees which can be shared with other sources. Since an incoming edge to a destination from any tree corresponds to a flow of one (every edge capacity is one), therefore every destination will need an incoming flow of at least $k*s$ for the s sources. Additionally there is a shared flow of n-k which will come from all the sources put together. Hence the total incoming flow to any destination should at least be $k*s + (n-k)$.
b) Since, for any given source n packets have to be sent on n different trees, we need an outgoing flow of at least n from each source.
c) Similar to the above reasoning since there needs to be n packets sent from any source to any destination on n disjoint paths, a minimum flow of n is required between any source and any destination. ■

Part (c) is needed to guarantee the max flow from each source to each receiver, but does not provide us with disjoint paths across sources. Therefore, it is only necessary, but not sufficient. ■

B. Necessary and Sufficient Conditions

The below theorem is necessary and sufficient for the source nodes to be able to be a part of the scheme.

Theorem 2:
The following are necessary and sufficient conditions:

a) $\mathbf{f}^{\mathbf{d}}_{G - \cup e_m \in f^d(S_{p,p\neq i}, T_l \in T)}(S_i, T_j) \geq k,\ \forall S_i \in S,\ T_j \in T$, *and*

b) $\mathbf{f}^{\mathbf{h}}_{G - \cup e_m \in f^d(S_p \in S, T_l \in T)}(S_i, T_j) \geq n\text{-}k,\ \forall S_i \in S,\ T_j \in T$

Sketch of proof:
Part (a) of the theorem states that the disjoint flow from a source to a destination can share links with a disjoint flow going from the same source to any other destination, but cannot share links going from any other source to any destination, including itself. In this case, the disjoint flow must be at least equal to k. Part (b) of the theorem states that the shared flow from a source to a destination, shall not share any link with any of the disjoint flows, and shall be at least equal to n-k.
Proof:
First we prove that (a) and (b) are necessary conditions, and we prove this by contradiction. Assume that k disjoint flows can be established from some source S_i to some destination T_j, over which k explicit shares can be sent. Now assume that (a) does not hold, e.g., $\mathbf{f}^{\mathbf{d}}_{G - \cup e_m \in f^d(S_{p,p\neq i}, T_l \in T)}(S_i, T_j) = k\text{-}1$, for at least one source, or one destination. This means that k disjoint paths cannot be established from source S_i to destination T_j, which means the destination cannot receive k explicit shares from S_i. This contradicts the assumption, which means that

$\mathbf{f}^{\mathbf{d}}_{G - \cup e_m \in f^d(S_{p,p\neq i}, T_l \in T)}(S_i, T_j) \geq k$. Similarly, let there be n-k redundant shares sent from S_i to T_j, which can be combined with redundant shares from other sources. If (b) did not hold, and for example $\mathbf{f}^{\mathbf{h}}_{G - \cup e_m \in f^d(S_p \in S, T_l \in T)}(S_i, T_j) = n\text{-}k\text{-}1$, then there are not enough paths over which the n-k redundant combinations can be delivered to the destination T_j. Therefore, (b) must hold.
Next, we prove that these are sufficient conditions. We start by assuming the satisfaction of (a) and (b). Since (a) is satisfied, then each source can deliver k shares on k disjoint paths. These paths can be shared with paths from the same source to another destination, hence forming k trees that are used to send the explicit shares. Since (b) is satisfied, then it is possible for each source to transmit an additional n-k shares on an additional n-k paths. Since these paths are not necessarily disjoint for each source, then a share from each source can be combined with up to n-k-1 other shares, such that no more than one share is generated by the same source on a given path. The reason for this last condition, namely, *no more than one share from a source on a given path*, is because if there is more than one share, then another source will have fewer than n-k paths which can be shared, which contradicts the assumption. This proves that if these conditions are satisfied, then it is possible to transmit from each source to a destination on k disjoint paths, and also on n-k paths, which are not necessarily disjoint from shares sent by other sources.
Therefore, the conditions are both necessary and sufficient. ■

An example network is shown in Fig.1 (a) and consists of 21(N) nodes. There are three sources (s=3) A, B and C and two destinations (r=2) T, U. n is 4 and k is 2. Hence two disjoint trees for each of the sources are shown in Fig.1 (b). The two shared trees are shown in Fig. 1(c). For the shared tree represented by the following edges {A-F, B-F, C-F, F-N, N-T, N-U}, the erasure coded packets from the sources A, B and C will be network coded at the node F. For the shared tree {A-I, B-I, I-Q, C-Q, Q-T, Q-U}, erasure coded packets from A and B are combined at node I which is then combined with the packet from node C at node Q.

IV. Multicast Tree Protection Algorithm

The protection algorithm consists of heuristic algorithms for flow constraint verification and creation of multicast trees.

A. Flow Verification Algorithms

The flow verification step consists of heuristic algorithms that verify Theorems 1 and 2. Without the destination and source nodes being able to support the required flow (incoming, outgoing and max), we cannot include them as a part of the scheme. Hence the first step of flow verification is filtering out those nodes with flow lesser than the minimum required. The next step is to examine sources which meet the minimum flow constraint but do not have enough disjoint flow to the destination nodes. Since finding the disjointedness in a network is not a linear time task, we employ heuristics using flow constraints mentioned in Theorem 2. The heuristics also indirectly effect an algorithm called grouping to account for sources not satisfying Theorem 2. Grouping is the process of segregating sources based on the amount of compromise in

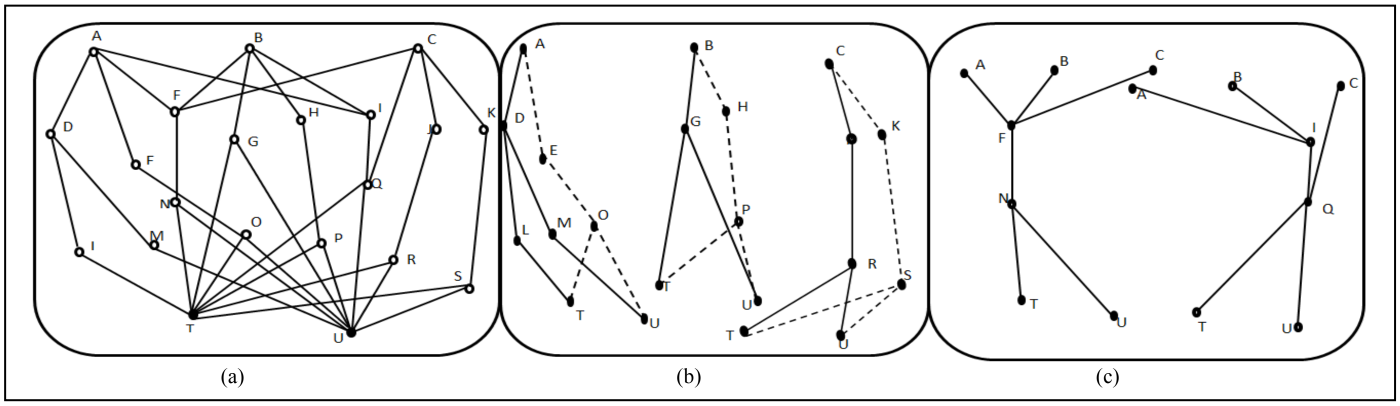


Fig. 1. a) Network graph with N=21 b) Disjoint trees for sources A, B and C respectively c) Shared trees for nodes A, B and C

throughput and protection they require. Sources are ordered based on increasing average max flow to all the destinations. This ensures sources possessing lesser resources are prioritized for finding trees as the algorithm for the estimation of multicast flow is asymptotic, and whose calculation by definition cannot be accomplished in linear time. The sources which have a higher max flow have more flexibility for choosing their paths. In this stage, sources undergo three steps, as explained below:

a) Evaluation of multicast flow constraint for disjoint trees

The algorithm for this step is Disjoint Tree based Source Filtering (*DTSF*), and is shown below. This algorithm consists of three steps.

1) Calculate flow paths from a given source to various destinations and generate all possible combinations of flow paths by taking only one flow each from every destination's flow path set.

2) Rearrange the flow combination set based on the assumption that the best flow combination is the one which has the least percentage of edges common with any other flow combination.

3) Every flow combination contributes to a multicast flow a value equal to ***r***. Find if source has enough multicast flow.

b) Grouping based on the amount of protection offered (n-k)

Grouping also forms an implicit part of *DTSF.* Sources are divided into groups, based on the amount of protection they can afford. Sources which can meet the constraint of $M^F = k*r$ are grouped into the default set U_0. Protection at reduced cost is made available to sources that cannot meet the previous constraint. The U_jth group offers a protection of *n-k-j.*

Algorithm Disjoint Tree based Source Filtering (*DTSF*)
Input Network Graph *G,* Destination Set *T,* Source Set *S`*
Output Source Group set *U,* Network Graph *G`*

```
for (all sources)
    Verify Theorem 2.a)
    if Theorem 2.a) holds for source
        Segregate source into the primary group
    else
        Compromise throughput  (Reduce k (and n) )
        if (Theorem 2.a) holds for source)
            Segregate source into the primary group
        else
            Compromise Protection  (Reduce  n))
            Verify Theorem 2.a)  with new (k, n)
            if (Theorem 2.a) holds for source)
                Segregate source into the relevant group
            else
                Discard Source
end for
```

c) Evaluation of multicast flow constraints for the shared trees

The algorithm for this step is described in 'Shared Tree based Source Filtering' (*STSF*) algorithm below:

Algorithm Shared Tree based Source Filtering (*STSF*)
Input Network Graph *G`* Destination Set *T`*
Source group set *U*
Output Refined Source group set *U*

```
for(all  groups)
  for(all  sources in  a  group)
    Verify Theorem 2.b)
    if (Theorem 2.b) holds for source)
        Segregate source into the relevant group
    else
        Compromise throughput  and /or Protection (Reduce k and n)
        Verify Theorem 2.b) with new (k, n)
        if (Theorem 2.b) holds for source)
            Segregate source into the relevant group
        else
            Discard Source
end for
```

Compromising throughput and protection may require *k.n* iterations.

C. Steiner Tree approach for Multicast tree Generation Algorithms

Connections have to be provisioned on the disjoint and shared path sessions once the feasibility of the scheme is verified.

This is accomplished by establishing multicast trees based on Steiner tree approach from every source to the predefined set of destinations. Since multicast trees shared between sources require higher connectivity, the shared tree connections are established prior to the disjoint set of trees. Since the Steiner Minimum Tree problem is known to be an NP-hard problem, we use the heuristic greedy steiner tree algorithm by Takahashi[13].

a)Shared Tree Generation

.

```
Algorithm   Shared Tree Generation Algorithm(STGA)
            This algorithm is used to establish the shared
            multicast trees.
Input       Network Graph G, Destination Set T`,
            Source group set U
Output      Shared trees $\cancel{G}^h_{S,T}$, Network Graph G`

for(all shared trees)
    Determine the sources and destinations which need to be a
    part of this tree(V) while ensuring that every source gets the
    required protection by being a part of the required number of
    shared trees
    Find a Steiner tree for V
end for
```

b)Disjoint Tree Generation

In this step, the *k* primary disjoint set of trees required by every source are established. The multicast trees required are generated using the Steiner tree approach.

V. Coefficient Selection And Coding

Throughput improvements are obtained by using network coding on the shared trees. This requires selection of coefficients from the Vandermonde matrix. We exploit the matrix structure in order to reduce the field size for network coding at intermediate nodes and for erasure coding at the source. A Vandermonde matrix of size $x*y$ consists of the coefficients (λ) that are chosen randomly from a finite field (F) of size $v \geq x$. Every group of packets from different sources which are used to form one combination to be delivered to destinations is assigned a unique λ_i . However, data packets belonging to different sources that will be combined together use the $\lambda_{i.}$ assigned to it, albeit with a different power of λ_i for each packet, i.e.,

$\forall \lambda_{i1}^{j1}, \lambda_{i2}^{j2} \in \lambda_i^j$, such that

$i1 = i2$, for all packets coded on the same shared tree,

$j1 \neq j2$, for all packets coded on the same shared tree.

$i1 \neq i2$, for packets coded on different shared tree,

At source node *j,* the following operations are performed:

Let packet matrix entries: $x_{ji}, 0<i\leq k, 0\leq j< s$

Also let coefficient matrix entries: λ_p^{i+k*j}, $0<p\leq n\text{-}k$, $0<i\leq k$, $0\leq j<s$, $\lambda_p^i \in F$

A. Data packets are sent according to the following:

a) On shared Trees, *1,2……n-k*

$$\sum_{i=1}^{k} x_{ji}(\lambda_1)^{i+k*j}, \sum_{i=1}^{k} x_{ji}(\lambda_2)^{i+k*j} \ldots\ldots \sum_{i=1}^{k} x_{ji}(\lambda_{n-k})^{i+k*j},\ 0\leq j\leq s$$

b) On disjoint Trees, *1,2……k*

$x_{j1}, x_{j2}\ldots\ldots x_{jk}, 0\leq j< s$

B. At intermediate node *w*, on shared multicast tree *p* with data packets from source set ~~*S*~~*,* where ~~*S*~~ represents all the sources whose share is present in the data on edges incoming to node *w,* the following is performed:

Network coded data on the *output edge of node w* ($Y^h(w)$) will be formed as:

$$Y^h(w) = \sum_{j\in \cancel{S}} \sum_{i=1}^{k} x_{ji}(\lambda_p)^{i+k*j}$$

C. At each receiver, the following combinations will be received:

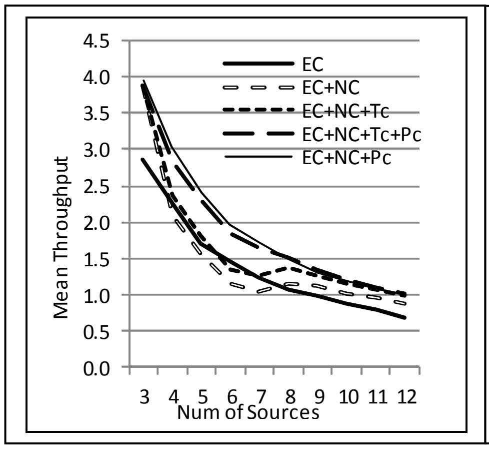
$$\sum_{j=1}^{s}\sum_{i=1}^{k} x_{ji}(\lambda_1)^{i+k*j}, \sum_{j=1}^{s}\sum_{i=1}^{k} x_{ji}(\lambda_2)^{i+k*j} \ldots\ldots\ldots\ldots\ldots\ldots \sum_{j=1}^{s}\sum_{i=1}^{k} x_{ji}(\lambda_{n-k})^{i+k*j}$$

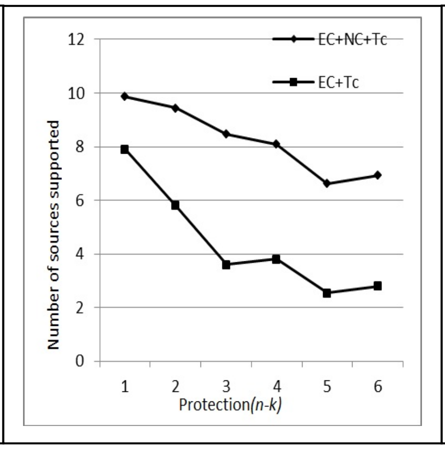
where $x_{j1}, x_{j2}\ldots\ldots x_{jk}$, $0\leq j\leq s$ are the explicit *k* data units from source node *j.*

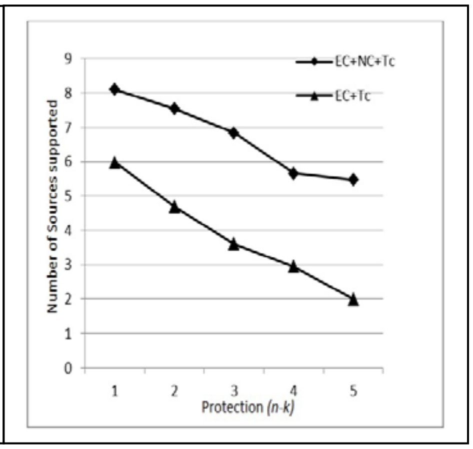
D. The field size required for coefficient selection is established as follows. There are a total of *s* sources with *k* disjoint paths and *n-k* shared paths to a given destination. Since there are *k* unknowns for every source, *k*s* equations are obtained from the various disjoint paths. Additionally, the shared paths will contribute another *n-k* equations in the best case (where all the source packets are combined into one packet for every shared path) or in worst case *(n-k)*s* equations. Either way, this results in *n-k* linearly independent equations at the receiver (because the *(n-k)*s* equations can be always combined at the receiver to form *n-k* equations where every equation includes one share from every source). Hence the field size required would be *n-k*+1. We note here that given the structure of the transmitted data matrix on the shared path, at a common node packet entries from different sources can be combined in an exclusive manner (no coding coefficients are required at the intermediate nodes). Hence network coding at intermediate nodes is an XOR operation and can be accomplished in F^2.

VI. Simulation Results

We compare the proposed scheme against the original *(k,n)* scheme of erasure coding. The performance improvement is demonstrated via throughput, cost of provisioning multicasts sessions and the number of sources that the proposed approach can support in different network conditions.



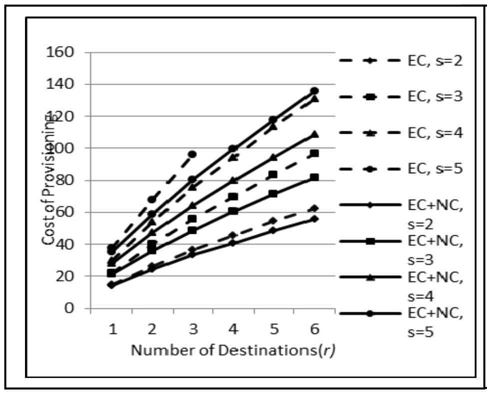


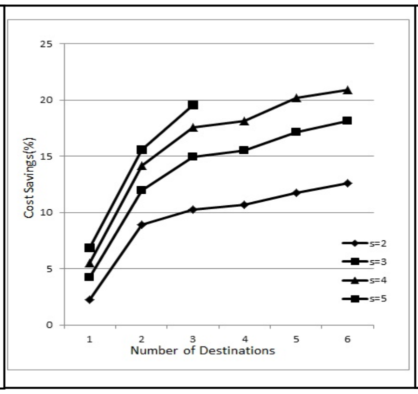


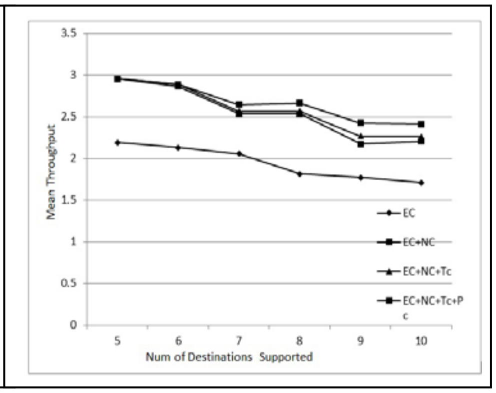
(a)

(b)

(c)







(d)

(e)

(f)

Fig. 2 a) Comparing cost of provisioning multicast session for Erasure Coding (EC) and proposed approach- a combination of Network and Erasure Coding
b) Cost Savings of proposed scheme over erasure coding.
c) Plot of Mean throughput/source with variable number of destinations for EC, EC+NC, EC+ NC+ Tc, EC+ NC+ Tc+ Pc
d) Plot of Mean throughput/source with variable number of sources for EC, EC+NC, EC+ NC+ Tc, EC+ NC+ Tc+ Pc and EC+ NC+ Pc
e) Plot comparing number of sources supported with flexible number of destinations for EC+NC+Tc and EC+Tc
f) Plot comparing number of sources supported with fixed number of destinations for EC+NC+Tc and EC+Tc

D. *Cost of Provisioning Sessions*

Since the number of disjoint trees has decreased because of sharing trees across source sessions, substantial reduction is seen in the cost of provisioning multicast sessions. To compare with erasure coding fairly, neither throughput compromise nor protection compromise is employed in this example. Cost was averaged over 50 randomly generated graphs with $N = 45$ with an average nodal degree equal to 24. The total number of paths supported was six (n=6) and the number of disjoint paths was four (k=4). Fig. 2(a) compares the cost of provisioning multicast sessions using the proposed approach to the erasure coding only method. Fig. 2 (b) shows the savings obtained in terms of cost by the proposed scheme over conventional erasure coding. It can be observed that as the number of sources and destinations in the network increase, the savings also increase. For s=4, the savings for the proposed scheme increases from 5% all the way up to 20% as destinations are added. This is understandable since the difference between the number of edges in the shared trees and the erasure coding's extra disjoint trees will keep increasing with the number of nodes that need to be supported.

E. *Throughput*

The proposed approach allows for compromising throughput and protection. If the number of sources is large and cannot be supported at the requested protection and throughput, the scheme allows:

1) *Reduction in throughput in order to accommodate the source (Tc),*

2) *Reduction in the amount of protection offered to the source (Pc), or*

3) *Reducing throughput and protection (Tc+Pc).*

The benefits are best measured using mean throughput obtained in supporting as many sources and destinations as possible. We generated 50 random graphs with N=40 and average nodal degree of 20, and averaged the throughput over all 50 networks. To simulate real time source throughput requirements K^{th} was randomly generated for every source such that $1 \leq K^{th} \leq k$. Two types of simulations are considered:

a) In the first case, the number of destinations was increased and the variance in throughput was observed. This has four sources and n=5 and k=3.

b) In the second case, the number of sources was changed. There are four destinations to which data has to be multicasted and n=6, k=4.

For case (a), the results are plotted in Fig. 2(c), and the mean throughput for the proposed scheme is shown to be better than the original erasure coding scheme. Our approach involving throughput and protection compromise performs better than all the other approaches.

For case (b), as the results plotted in Fig. 2(d) show, there is a substantial drop in throughput as the number of sources increases, and for all approaches. For the ones which involve decreasing throughput to support more sources, the mean throughput is bound to be less. The approach which performs best here is the one where only protection is compromised to support more sources. EC+NC+Pc does better than EC and EC+NC because it supports more sources than both of these approaches. It does better than EC+NC+Tc and EC+NC+PC+Tc because it does not compromise throughput to support more sources which either approaches do. As is evident from Fig. 2(d), EC+NC+Pc always does better than the other approaches.

F. *Sources supported for multicasting*

While the basic scheme of EC+NC attempts to decrease the cost of provisioning the multicast sessions, other approaches like EC+ NC+ Tc try to maximize the number of sources that could be supported by considering the sources that cannot meet the required constraints of EC+NC. Since throughput compromise can be implemented for the original erasure coding too, it is interesting to see how EC and EC+NC fare with respect to supporting maximum number of sources given both are allowed to compromise their throughput.

Random network graphs with N=40 and average nodal degree is 22 were generated. The results were averaged over 50 iterations with r=3. If the number of destinations is allowed to be reduced, then both approaches would be able to support more sources. Hence the comparison is given for both cases, with and without destination reduction:

a) The focus is on supporting as many sources as possible in spite of reduction in supported destinations.
b) Max number of sources that can be supported such that all the destinations are still part of the scheme.

For case (a), EC+NC+Tc is able to support around 10 sources when the protection n-k is 1, whereas EC+Tc is able to support only 8 sources. As the protection increases, the EC+NC+Tc supports more sources than EC+Tc. For a protection level of n-k =3, EC+NC+Tc is able to support more than 135% of what EC+Pc can support. Fig.2(e) shows these results.

For case (b), EC+ NC+ Tc is able to support an average of around 8 sources when the protection n-k is 1, whereas EC+Tc is able to support only 6 sources. This can be seen in the plot shown in Fig.2(f).As n-k increases, the required disjointedness in the network increases and EC+NC+Tc outperforms EC+Tc since it can leverage on sharing of trees across sources in order to improve performance.

VII. CONCLUSION

We have proposed an approach that enables packets from multiple multicast sources to be network coded and be sent to multicast destinations in multi-source, multi-sink wireless sensor networks. The approach uses the principle of erasure coding where k out of n copies are required at the receiver to be able to decode the data. The scheme therefore supports breakdown of any n-k paths of a source at any given time. The recovery is done by utilizing the packets obtained on the primary and shared paths of other sessions. This necessitates that the primary paths of all sources be disjoint. We also proposed a heuristic algorithm for implementing this approach.

The advantages of using our implementation include cost savings obtained by network coding packets belonging to different sources. Simulations show an average of 20% savings in bandwidth (provisioning costs). The approach is also able to support sources that cannot meet the stricter flow constraints of the basic scheme, by way of reduced throughput or protection. Therefore, certain sources can take part in the multicast sessions in spite of having lower connectivity. Network coding at the intermediate nodes is a simple XOR operation. Additionally field size for generating coded packets at the source is small. The recovery of source packets at the receiver can be instantaneous providing proactive protection to the source sessions.

Future work will include the cases where sessions support different numbers of destinations for every source, as well as develop scheduling algorithms.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, “Network information flow”, IEEE Transactions on Information Theory, Vol. 46, pp. 1204–1216, July 2000
[2] S. Alouneh, A. Agarwal, A. En-Nouaary, “A Novel Approach for Fault Tolerance in MPLS Networks”, Innovations in Information Technology, Nov 2006
[3] R. Koetter and M. Medard, “An algebraic approach to network coding”, IEEE Transactions on Networking, 2003
[4] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, and L. Tolhuizen, “Polynomial time algorithms for multicast network code construction”, IEEE Transactions on Information Theory. Vol. 51, No.6, 2005
[5] T. Ho, M. Medard and R. Koetter, “An Information-Theoretic View of Network Management”, IEEE Trans. on Information Theory, April 2005
[6] S. Aly, A. E. Kamal and O. M. Al-Kofahi, “Network Protection Codes: Providing Self-healing in Autonomic Networks Using Network Coding”, Elsevier Computer Networks, 56 (1), 12 January 2012, Pages 99-111.
[7] Y. Wu, “On Constructive Multi-Source Network Coding”, IEEE Int. Symposium on Information Theory, pp. 1349-1353, July 2006
[8] P.Bao-xing, Y. Lu-ming, W. Wei-ping, X. Xiao, “Linear Network Coding Construction for Multi-Source Multicast Network”, First Int. Workshop on Education Technology and Computer Science, 2009.
[9] A. E. Kamal, A. Ramamoorthy, L. Long, S. Li, “Overlay Protection Against Link Failures Using Network Coding”, IEEE/ACM Transactions on Networking, Vol. 19, No. 4, Aug. 2011, pp. 1071-1084.
[10] M. Mohandespour, A.E Kamal, “1+N Protection in Polynomial Time, A Heuristic Approach”, IEEE Globecom, 2010
[11] Al-Kofahi, O. and A. E. Kamal, "Scalable redundancy for sensors-to-sink communication", in the proceedings of the IEEE Globecom 2008
[12] Al-Kofahi, O. M. and A. E. Kamal, "Network Coding-Based Protection of Many-to-One Wireless Flows", IEEE Journal on Selected Areas in Communications special issue on Network Coding in Wireless Communications, Vol. 27, No. 5, June 2009,pp. 797—813
[13] H.Takahashi and A.Matsuyama, “An approximate solution for the Steiner problem in graphs”,Math.Japonica,24(6):573-577,1980.