Scalable Redundancy for Sensors-to-Sink Communication

Osameh M. Al-Kofahi Ahmed E. Kamal
Department of Electrical and Computer Engineering, Iowa State University
e-mails:{osameh, kamal}@iastate.edu

***Abstract*—In this paper, we present a new technique that uses deterministic binary network coding in a distributed manner to enhance the resiliency of Sensor-to-Base information flow against packet loss. First, we show how to use network coding to tolerate a single packet loss, and then we extend the solution to tolerate multiple losses. Moreover, we study the coding efficiency issue and introduce the idea of relative indexing to reduce the coding coefficients overhead. Finally, we show through a simulation study that our approach is highly scalable and performs better as the network size and/or number of sources increases.**

I. Introduction

Wireless Sensor Networks (WSNs) [1] are composed of a large number of sensing nodes, which are deployed in a certain area to monitor a phenomenon of interest. WSNs are usually deployed in harsh environments where packet loss and node failure is common, which in turn degrades the quality of the monitoring process. Survivability techniques are proposed to mitigate the effect of these problems, where in all these techniques the improved reliability comes at the price of increased redundancy. Usually this is done by duplicating information on multiple paths, which either can be totally disjoint or braided [2], [3], [4]. Sufficient connectivity is a prerequisite for multipath routing, for example in [5], [6] sensor deployment and topology control algorithms were proposed to guarantee a k-connected network.

In [3], information is sent on multiple paths to enhance the network reliability. A forwarding mesh that is composed of a set of interleaving paths was used in [2] to relay information from a source node to the base station (BS), where each data unit is given a budget that controls the degree of redundancy or the width of the mesh. On the other hand, in [4], the author introduced an efficient algorithm to give each node a set of node-disjoint paths to choose from in the case of a failure on the primary path. The concept of event-to-sink reliability in WSNs was introduced in [7], where a certain event is considered reported if the number of reports reaches a certain threshold. It was shown that the overhead from duplicating the data units can be reduced by using erasure codes in [8] and [9], where a packet is encoded into n smaller sub-packets such that only k of them, where $k < n$, are necessary to recover the original data unit. These sub-packets can be sent on the same path as in [8] or on node-disjoint paths as in [9].

In this paper, we introduce a new technique that will dramatically reduce the redundancy overhead compared with that in duplication-based multipath mechanisms, and yet still achieves the same level of survivability. We accomplish this by using network coding [10], [11]. In [12] we showed that network coding can be used to tolerate e failures in many-to-one flows, by having each set of n sources send at least $n+e$ combinations to the sink, such that any n of them are linearly independent. However, the scheme in [12] requires global information (in the first run at least) and assumes a fixed set of sources at a certain time. In this paper we start by showing how to use deterministic network coding in a distributed manner to protect the data units from k sources against a single packet loss, which is discussed in Section II. Our solution is extended in Section III to tolerate multiple packet losses. Moreover, we discuss some coding-related issues in Section IV, where we introduce the ideas of relative indexing and best effort decoding. Simulation results are presented in Section V. Finally, we conclude the paper in Section VI.

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

II. Operation

Throughout this paper we focus our discussion on tolerating packet losses. Nevertheless, the described methods can be used to tolerate node or link failures if properly combined with a routing algorithm that computes node-disjoint (or link-disjoint) paths, which is out of the scope of this paper.

We consider a dense and uniformly distributed WSN, in which packet losses may occur. Assuming that at most one packet can be lost, our objective is to practically provide proactive protection to the information flow from the sensors to the BS using as few resources as possible. Before discussing the details of node operation, we need to clarify our assumptions and notations. Specifically, we assume the following:

1) We assume a wireless sensor network in which the sensor nodes can be organized into rings around the base station, such that, the minimum hop count between the sensor nodes in ring i and the BS is i hops. An example is shown in Figure 1, where nodes d, e and f are in the first ring, nodes c and b are in the second ring and finally node a is in the third ring around the BS.
2) We assume that there are $\mathcal{R}$ rings in the sensor network, and that the BS is the only node in ring zero (R_0).
3) Routing a packet ensures its progress towards the sink in each transmission, i.e. a packet gets closer to the sink by one hop after each transmission.
4) Sensors generate data periodically and at the same rate.
5) The data units for all sensor nodes are equal in size, and the data unit for sensor node u is denoted by d_u.

978-1-4244-2324-8/08/$25.00 © 2008 IEEE.

6) Throughout the paper we use p to denote a certain packet, and we reserve d_p to only represent the information symbol carried in packet p.

Note that the first three assumptions are not necessary, but they will help in streamlining the discussion.

Assume that sensor node u has a data unit d_u that must be forwarded to the BS. Node u can send two copies of d_u to the sink to tolerate a single packet loss as a proactive alternative to retransmission. However, if there is a large number of sensor nodes that need to forward information to the sink, this solution is not efficient anymore. This is because 50% of the forwarded information is redundant, and thus, at least 50% of the network resources (bandwidth and energy) will be wasted to provide this redundancy.

Assume that node u has sent two copies of d_u to the sink. Naturally, this is done through multi-hop communication for each of these packets. Suppose that as the two packets are forwarded, $k-1$ of the forwarding nodes had data units of their own that also need to be sent to the BS. We show how to protect d_u and the $k-1$ other data units by forwarding only $k+1$ packets, through the use of network coding. In general, the nodes in a wireless sensor network can be divided into the following three types:

1) **Type 1.** A source with no data to relay.
2) **Type 2.** A source with data to relay.
3) **Type 3.** Just a relay with no data of its own.

To tolerate a single loss using network coding in a distributed manner, we need to specify the way a node operates given its local information. We define the following operation for each class of nodes:

1) **Type 1.** Assume node u only has its own data unit d_u, and has no packets from other sensors to relay. Then node u just sends two copies of d_u to be able to tolerate a single loss. For example, node a in Figure 1. Since wireless communication is used, these two copies can be delivered to the next hop nodes in one transmission.
2) **Type 2.** Assume node u has its own data unit d_u, and in addition has received another packet, say p, which needs to be relayed (e.g., node b in Figure 1). Then node u produces the following two packets:
   a) **Packet 1.** Contains the bitwise XOR of d_u and d_p.
   b) **Packet 2.** Contains only d_u.
3) **Type 3.** Assume node u has no data of its own, and has received a packet p that it needs to relay. Then it just forwards p as is, e.g., nodes c, d, e and f in Figure 1.

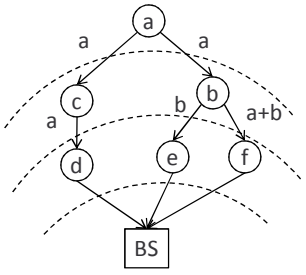

Fig. 1. Protecting data from two sources against a single packet loss

The type of a node changes according to its status, i.e., if node u is of Type 3 at a certain time instant t_i, it might become a Type 1 or Type 2 node at t_{i+1} if a data unit is generated locally. Now recall the scenario of node u and the $k-1$ other sources that forward the data units of u. In this scenario node u is the only node of Type 1, and the $k-1$ sources are of Type 2. Note also, that each of the $k-1$ sources increases the number of total packets by 1, which means that the total number of generated packets is $k+1$. Let us call this operation a *forwarding process initiated by node* u, or for short, an *F-process* initiated by a Type 1 node, u.

We now prove that by following the rules of operation, if k sources are involved in a certain F-process then they will produce $k+1$ packets such that any k of them carry a solvable set of combinations.

Claim 1. *If s is the k^{th} source to participate in an F-process initiated by node u, in which the number of involved sources until now is $k-1$ (including u), then by following the rules of operation described above, s will increase the number of packets to $k+1$, such that any k packets from them carry a solvable set of combinations.*

Proof. We prove this claim by induction. We need to show that if any $k-1$ from the current k combinations (produced by the $k-1$ sources) are solvable, then the participation of the k^{th} source will produce $k+1$ combinations such that any k of them are solvable.

The *basis step* is when the F-process is first initiated at node u. By following the rules of operation, node u will send two copies of d_u. Thus we have $k=1$ source, and $k+1=2$ combinations (trivial combinations in this case), such that any one (k) of them is solvable, which is obvious since each combination contains only one data unit, so if one packet is lost the other is sufficient to recover d_u.

To prove the *inductive step*, assume that $k-1$ sources have transmitted and any $k-1$ from the current k combinations are solvable. Node s can participate only if it is of Type 2, i.e., it received one packet (say p) from the current k packets and it has its own data unit d_s that must be sent to the BS. By following the rules, s will produce two packets: 1) a packet containing the XOR of d_s with d_p (note that the number of total packets is still k since p and d_u are merged into one packet), and 2) a packet containing d_s (i.e., the number of packets is increased by 1). We now need to show that losing either one of the newly created packets will leave us with k solvable combinations. If we lose the first packet, then the second packet will be sufficient to recover d_s, and from our assumptions the remaining $k-1$ combinations will be sufficient to recover the remaining $k-1$ data units. If we lose the second packet, then from our assumptions we can recover all the data units in d_p (and thus d_p itself) using the $k-1$ combinations other than $d_s \oplus d_p$, and then recover d_s by $d_p \oplus (d_s \oplus d_p)$. Finally, if we lose a packet other than those produced by s, we can recover d_s from the second packet produced by s, which leaves us with $k-1$ solvable combinations in $k-1$ unknowns. ■

A node of Type 1, will initiate a process that will involve a large number of nodes of the other two types. To distinguish the packets belonging to the process initiated by a node of Type 1, we add a new field in all the packets generated by

this process. Let us call this field the *"initiator ID"* or *IID* for short. A node of Type 1 will put its ID and a timestamp in this field, and a node of Type 2 (say u) that XORs d_p with its data unit will copy the IID from p and put it in both generated packets. If a Type 2 node receives 2 or more packets with the same IID it must not combine its data with more than one of them, since this may produce dependent combinations.

Let P_i^j be the set of packets leaving level i (i.e., generated by level i, or just forwarded by it) that belong to the process initiated by node j, then $|P_i^j|$ is equal to:

$$|P_i^j| = 1 + \sum_{k=i}^{\mathcal{R}} S_k^j = 1 + N_i^j \quad (1)$$

where S_k^j is the number of source nodes (Type 1 and Type 2) in ring k that are involved in the process initiated by j, and N_i^j is the number of source nodes in the rings from i to $\mathcal{R}$ that are involved in the process initiated by node j. Of course, there is only one node of Type 1 in the process initiated by node j, and that is j itself.

Certainly, there should be a limit on the number of combinations that can be produced by an F-process, and that is the minimum cut between all the participating sources transmitting their data to the BS. In a sufficiently dense and uniformly distributed WSN, the minimum node cut is usually the number of one hop neighbors of the BS. We assume that the min node cut is equal to h, where $h \geq 2$. Therefore, no more than $h-1$ sources can be involved in any process. To insure this, we add a new field in the packet format; Let us call it *"Number of Remaining Combinations"* or *"NRC"* for short. We now redefine the operation of Type 1 and Type 2 nodes as follows (the operation of Type 3 nodes do not change):

1) **Type 1.** Assume node u only has its own data unit d_u, and has no data units from other sensors to relay. Then node u just sends two copies of d_u, such that in both packets the values of $IID = u$ and $NRC = \lfloor h/2 \rfloor$.
2) **Type 2.** Assume node u has its own data unit d_u, and in addition has received another packet, say p, that it needs to relay. The operation of u depends on the NRC value of p as follows:
   a) If $NRC(p) \geq 2$ node u produces the following two packets, such that, in both packets *IID* = $IID(p)$ and NRC = $\lfloor NRC(p)/2 \rfloor$.
      i) **Packet 1.** Contains $d_u \oplus d_p$.
      ii) **Packet 2.** Contains only d_u.
   b) However, if $NRC(p) = 1$, node u acts as a node of Type 3, i.e., just a relay.

Note that by taking the floor when updating the NRC value, h is made equivalent to the largest power of 2 that is less than or equal to h. Note also how the resulting tree resembles a binary tree, where each leaf is a combination in our case. However, our tree is restricted in depth, where it can have at most $d = \log_2 h$ levels. Therefore, the maximum number of combinations (or leaves) is $2^d = 2^{\log_2 h} = h$, which is the case only when we have a complete binary tree of d levels. By this we insure that the maximum number of combinations is less than or equal to h and the maximum number of participating sources in less than or equal to $h-1$. For example, consider the network in Figure 2 where $h = 7$. Node a is the initiator node (Type 1 node), by following the rules of operation a will send two packets carrying d_a to b and c with $IID = a$ and $NRC = \lfloor \frac{7}{2} \rfloor = 3$. Node c is a Type 3 node, i.e., it will relay only and will not change the value of the IID or the NRC. Node b is a Type 2 and according to the rules of operation it will produce two packets with $IID = a$ and $NRC = \lfloor \frac{3}{2} \rfloor = 1$. It is obvious that starting with $h = 7$ we can have at most 4 packets produced, which is equivalent to starting with $h = 4$ (i.e., the closest power of 2 less than 7). Therefore, in the remainder of this paper we will always assume that h is a power of 2.

It is worth mentioning that assigning the NRC value as described in the rules of operation is not optimal in terms of maximizing the number of participating sources in a certain F-process. For example in Figure 2, assume that similar to node c node d is also a Type 3 node and the packet forwarded from d will reach the BS on a path composed only of Type 3 nodes. However, assume that node f has a data unit of its own that it wants to combine with the packet received from b. It will not be able to participate since the NRC value in the received packet is 1. Therefore, if node a knew beforehand the source nodes that will relay its packets, it can set the NRC to 1 in the packet sent to c, and to 6 (which is equivalent to 4) in the packet sent to b. Nevertheless, since we assume that a node operates using only its local information, and the deployed network is dense and uniformly distributed, a realistic assumption that an initiator node can make is that both packets will pass through the same number of sources approximately, and thus divide h by 2. Using simulation, in Section V we verify that using this assumption results in a performance that is not much worse than the optimal.

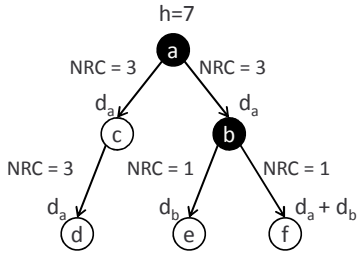

Fig. 2. NRC update for h=7

III. Tolerating Multiple Losses

The operation described in Section II proactively protects each generated data unit against a single loss. Suppose we want to protect each data unit against e losses. We can do this using the same operation described in Section II by allowing a Type 1 (Type 2) node, say u, to initiate (participate in) multiple F-processes for the same data unit. Assuming that at most e packets (i.e., combinations) can be lost, d_u must be inserted by u in at least $e+1$ combinations. Since each time u participates in a process two combinations are produced, node u must participate in (or initiate) $\frac{e+1}{2}$ F-processes. Therefore, if e is even (i.e., $e+1$ is odd) either u participates in $\lceil \frac{e+1}{2} \rceil$ F-processes, or in $\lfloor \frac{e+1}{2} \rfloor$ processes and produces a packet containing d_u only with $NRC = 1$ to complete the $e+1$

insertions ($NRC = 1$ to prevent other sources from combining their data with this combination).

Specifically, we assume that each node of Type 1 or Type 2 keeps a set of counters that are initialized to $e + 1$, each of which is associated with a certain data unit that is generated by the sensor node itself. Each counter keeps track of the remaining number of participations needed to provide protection against e failures for a certain data unit. A counter is decreased by 2 each time the node participates (or initiates) in a certain F-process until it reaches 0, where no further participations are needed. An example is shown in Fig.3, where we want to tolerate 3 packet losses for each of the data units, i.e., $e = 3$. There are three Type 1 nodes (A, B and C), and three Type 2 nodes (D, E and F), each of which has a single data unit to be sent. We only focus on the packets going through nodes D, E and F, where each of these nodes forward traffic from more than one F-process, and we assume that the paths for the remaining packets do no not intersect. For every data unit a counter is initialized to 4, i.e., each Type 1 node can *initiate* 2 F-processes and each Type 2 node can *participate* in 2 F-processes. For example consider node A, the two packets sent from A to nodes D and E represent one process initiated by A, and the remaining two packets represent the other process initiated by A after which the counter will be 0 for d_a. Nodes D and E receive packets from all the three Type 1 nodes, but since they can only participate in 2 processes, they choose to participate in the processes initiated by nodes A and B. Therefore, when nodes D and E receive the packet from C they act as Type 3 nodes and just relay the packet as is. As shown in Figure 3, this operation results in a total of 18 packets to protect the data units from 6 sources against 3 losses.

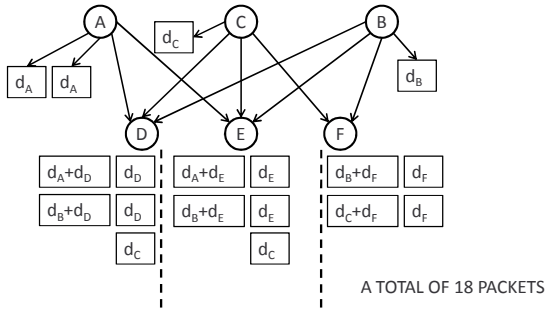

Fig. 3. Protecting against 3 losses

IV. Coding/Decoding Issues

The efficiency of network coding is an important issue that rises in most network coding-based applications. In this section we show that the overhead, which results from the need to send the coding vectors along with the combinations to the BS, can be significantly reduced by using relative indexing. Moreover, we introduce the idea of best effort decoding, which allows us to make use of the combinations received at the BS even if more than one packet was lost.

A. Relative Indexing for Efficient Encoding

For the sink to be able to decode the linear combinations, it needs to have the chosen coefficients or the coding vectors for each one of the combinations. In binary network coding for multicast connections, the length of the coding vector is determined by the minimum max-flow h between the single source and any of the terminals [10], where position i in the vector is reserved for d_i, i.e., if index i = 1 then d_i is present in the combination, and if index i = 0 then d_i is not. This is possible in multicast connections because all the data units originate from the same source, which means that this single source can assign for each data unit a unique index to identify it with. However, in our case the coding process is distributed, and thus the coding vector should be of size N, where N is the total number of nodes in the network, where each node is a possible source at some time. In a sensor network N can be very large, where it can be in the hundreds or even thousands of nodes. It is obvious that this is a waste of bandwidth since at most $h - 1$ sources can participate in a certain F-process.

We now show how to enable the source nodes in an F-process to use an $(h - 1)$-bit coding vector by relatively indexing the sources in that process. To do this, we use the NRC value in addition to a new control bit that we will add to the packet header, which we refer to as I. Assume that both packets generated by an initiator node A will be combined with data units from two other sources B and C. The question is if A is given index 1 in the $(h - 1)$-bit coding vector, how can B and C (and any other Type 2 node in the process) choose distinct indices using the values of NRC and control bit I?

In an F-process there is only one node of Type 1, which will be always assigned index 1 (index 0 will not be used). This leaves us with (h-2) indices, which will be recursively divided into halves. Let us consider the pair of Type 2 nodes in the first coding level after A (i.e., B and C), where each of them receives a packet with $NRC = \frac{h}{2}$, but B receives a packet in which $I = 0$ and C receives a packet in which $I = 1$. We give B the first index in the first half, and C the first index in the second half of the remaining h-2 indices. Half of h-2 is $NRC - 1$ (since $h = 2 * NRC$), which means that relative to index 1 the first half begins at the next position **after** *position 1* (i.e., 1+(1)), and the second half begins **after** the next $NRC-1$ positions after *position 1* (i.e., 1+($NRC-1$)+1). This is shown in Figure 4, where h is 8, the data unit from node B will be given index $1+(1) = 2$, and the data unit from node C will be given the index $1+(NRC-1+1) = 1+NRC = 5$. In the second iteration the process continues but with reference to position $1 + (1)$ for the nodes descending from B, and with reference to position $1 + (NRC - 1) + 1$ for the nodes descending from C.

To generalize for other levels, let $X(p)$ be the coding vector of the combination carried in p, and let X_i be an $(h - 1)$-bit vector with 1 in the i^{th} position and zeros otherwise. In addition, let $I(p)$ and $NRC(p)$ denote the value of bit I and the NRC field in p respectively. If a node u decides to participate in the F-process of some packet p, it calculates its index according to the following equation:

$$Index(u) = \max_{i:X_i \wedge X(p)=X_i} i + I(p) * (NRC(p) - 1) + 1 \quad (2)$$

where the values for $I(p)$ and $NRC(p)$ are set according to following operation (we only redefine the operation for type

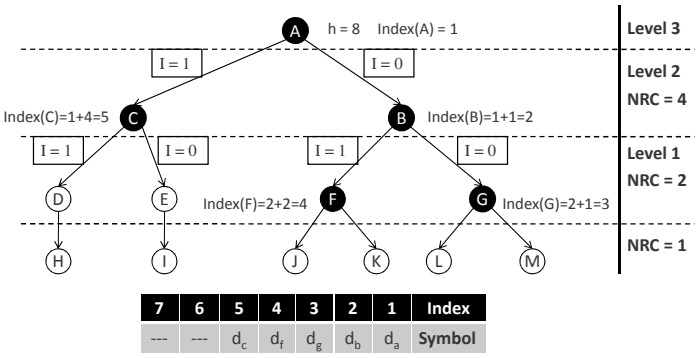

Fig. 4. Relative indexing, for $h = 8$

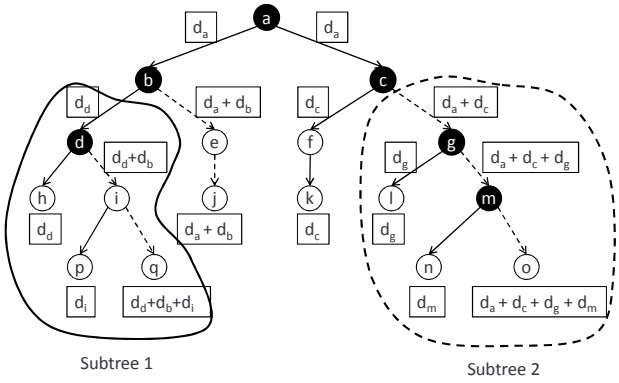

Fig. 5. Best Effort Decoding

1 and 2 nodes):

1) **Type 1.** Assume node u has its own data unit d_u, and has no data units from other sensors to relay. Then node u sends two packets carrying d_u, such that $IID = u$ and $NRC = \lfloor h/2 \rfloor$ in both packets. In addition to the following settings:
   - **Packet 1.** $I(Packet1) = 0$, $X(Packet1) = X_1$.
   - **Packet 2.** $I(Packet2) = 1$, $X(Packet2) = X_1$.
2) **Type 2.** Assume node u has its own data unit d_u and has received another packet p. The operation of u depends on the NRC value of p as follows:
   a) If $NRC(p) \geq 2$ node u calculates its index using equation 2. Then it produces the following two packets, such that, in both packets $IID = IID(p)$ and $NRC = \lfloor NRC(p)/2 \rfloor$.
      i) **Packet 1.** Contains $d_u \oplus d_p$, and has $I(Packet1) = 0$, and $X(Packet1) = X(p) \oplus X_{Index(u)}$.
      ii) **Packet 2.** Contains only d_u, and has $I(Packet2) = 1$, and $X(Packet2) = X_{Index(u)}$.
   b) However, if $NRC(p) < 2$ node u acts as a node of type 3, i.e., just a relay.

For the sake of illustration, we went through one more iteration for the descendents of B, where the NRC is set to 2 in the packets received by nodes F and G, resulting in G taking index 2+(1)=3 and F taking index 2+(2-1 + 1)=4. Lastly, we want to emphasize that relative indexing is valid only for a set of combinations having the same IID.

B. Best Effort Decoding

For the operation described in Section II, if e combinations were lost from the resulting $1 + N_1^j$ combinations in a certain F-process, the sink should not discard the remaining combinations because it may still be able to recover a subset of the encoded symbols. Actually, the remaining combinations received at the BS will still be solvable, if the lost combinations remove $e - 1$ data units, i.e., $1 + N_1^j - e$ equations remains in $N_1^j - (e - 1) = 1 + N_1^j - e$ unknowns. From the operation described in Section II this condition is satisfied for any subtree rooted at a Type 2 node u that combines its data unit with a trivial combination (i.e., a single data unit combination).

Consider a subtree rooted at a Type 2 node u, which encodes its data unit with a single data unit combination containing only d_v. The number of combinations this subtree will produce is equal to the NRC value in the packet carrying d_v. In addition, the number of new data units that will be added in these combinations by the sources in the subtree (including the root node u) is $NRC - 1$. Thus we will have NRC combinations in NRC unknowns, which can be solved. However, if the Type 2 node u combined its data unit with a non-trivial combination, the number of unknowns will be larger than the number of combinations. An example is shown in Figure 5, where Type 2 nodes that receive trivial combinations are the heads of all solid links, and Type 2 nodes that receive non-trivial combinations are the heads of all dashed links. For instance the combinations from Subtree 1 are solvable since node d combines its data unit with a trivial combination, while those from Subtree 2 are not.

In reality the BS does not need to search for solvable subtrees to decode them. Rather, it can use the decoding method in algorithm 1 for a set of combinations having the same IID, where RCV is the set that will contain the recovered data units, and Z is a set for temporary use in decoding. This way the BS will recover data units as much as it can, and will stop when no new single data units are found, hence the name best effort decoding.

Algorithm 1 Best Effort Decoding

1: $RCV = \phi$
2: Z= All single data unit combinations.
3: **while** $Z \neq \phi$ **do**
4: For each element in Z remove it from all combinations containing it, using bitwise XOR.
5: $RCV = RCV \cup Z$
6: Z= All new single data unit combinations.
7: **end while**

V. Simulation Results

We evaluated the performance of our scheme in terms of produced overhead, using simulation on TOSSIM. The effects of the following three parameters were studied in our experiments: 1) S, which represents the number of packets generated by each sensor node, 2) the number of nodes in the network, denoted by N, and 3) h, which controls the degree of coding. As a reference, we compared our results to a theoretical lower bound, which represents the best way in

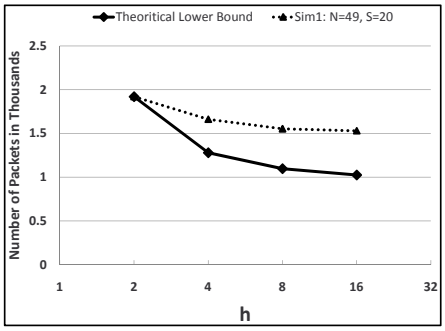

Fig. 6. Simulation results for N=49 and S=20

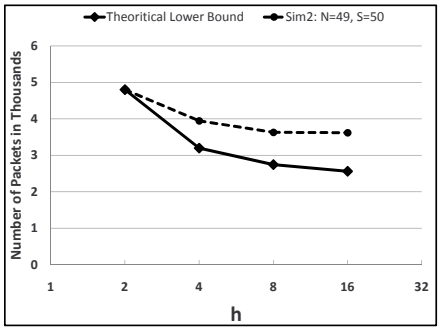

Fig. 7. Simulation results for N=49 and S=50

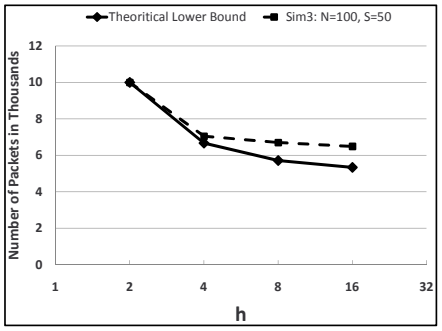

Fig. 8. Simulation results for N=100 and S=50

which packets can be combined. The best case occurs when every $h-1$ nodes belong to a certain F-process in which all of them are either Type 1 or Type 2 nodes. That is, we will have $\frac{N}{h-1}$ groups of nodes each of which produces a total of hS packets. This gives the following lower bound on the total number of packets produced:

$$(\frac{h}{h-1})N * S$$

We experimented with three scenarios, where in all of these scenarios h took the values $\{2, 4, 8, 16\}$. Note that when $h = 2$ no coding will take place and our scheme will be equivalent to traditional duplication based redundancy. The first scenario is a 7x7 grid network where each node produced 20 packets, the result is shown in Figure 6, where at h=16 the duplication overhead is reduced by 42%. The second is also a 7x7 grid, but each node produced 50 packets to see how increasing the load affects the performance. Figure 7 shows that by increasing the number generated packets our approach performs better and decreases the duplication overhead by 50% at h=16. Finally, we increased the size to a 10x10 grid network and fixed the number of produced packets to 50, to see how the performance is affected by increasing the network size in addition to the number of packets. Again, the results assure that the performance gets better with increasing the network size and generated packets, where at h=16 the duplication overhead is reduced by 72%. It is clear that the third scenario is the closest to the lower bound, since the chances for packet combining are enhanced as the network and/or number of generated packets grows larger. Although these scenarios are relatively small in size, they clearly illustrate that our coding-based approach is scalable and performs better as the network size increases.

VI. Conclusions

Deterministic binary network coding was utilized in a distributed manner to add lightweight redundancy to the information flow from the sensor nodes to the base station. Relative indexing was introduced to enable the use of an $(h-1)$-bit coding vector instead of an N-bit coding vector, where N is the number of sensor node (which can be in thousands), and $h << N$. In addition, we presented best effort decoding that allows us to make use the combinations received at the BS even if more than one packet was lost. Finally, simulation results confirmed our theory and proved that our scheme is highly scalable, where it performs better as the network size and/or the number of sources increases.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. Computer Networks 38 (2002) 393422.

[2] F. Ye, G. Zhong, S. Lu, and L. Zhang. Gradient broadcast: A robust data delivery protocol for large scale sensor networks. ACM Wireless Networks (WINET), vol. 11, no. 2, March 2005.

[3] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. ACM SIGMOBILE Mobile Computing and Communications Review archive Volume 5 , Issue 4 (October 2001).

[4] W. Lou. An efficient n-to-1 multipath routing protocol in wireless sensor networks. In Proc. 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2005).

[5] N. Li and J.C. Hou. Flss: A fault-tolerant topology control algorithm for wireless networks. In Proc. 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc 2004).

[6] J.L. Bredin, E.D. Demaine, M. Hajiaghayi, and D. Rus. Deploying sensor networks with guaranteed capacity and fault tolerance. In Proc. 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc 2005).

[7] Y. Sankarasubramaniam O.B. Akan I.F. Akyildiz. Esrt: Event-to-sink reliable transport in wireless sensor networks. In Proc. 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc 2003).

[8] S. Kim, R. Fonseca, and D. Culler. Reliable transfer on wireless sensor networks. In Proc. SECON 2004.

[9] S. Dulman, T. Nieberg, J. Wu, and P. Havinga. Trade-off between traffic overhead and reliability in multipath routing for wireless sensor networks. In Proc. WCNC 2003.

[10] R. Ahlswede, N. Cai, S. R. Li, and R. Yeung. Network information flow. IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 46, NO. 4, JULY 2000.

[11] R. W. Yeung, S. R. Li, N. Cai, and Z. Zhang. *Network Coding Theory*. now Publishers Inc, 2006.

[12] O. Al-Kofahi and A. Kamal. Network coding-based protection of many-to-one flows. In Proc. 4th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2007).