A Network-Coding Based Event Diffusion Protocol for Wireless Mesh Networks

Roberto Beraldi[1] and Hussein Alnuweiri[2]

[1] "La Sapienza" University of Rome, Rome, Italy
beraldi@dis.uniroma1.it
[2] Electrical & Computer Engineering Texas A&M University at Qatar
hussein.alnuweiri@qatar.tamu.edu

Abstract. Publish/subscribe is a well know and powerful distributed programming paradigm with many potential applications. In this paper we consider the central problem of any pub/sub implementation, namely the problem of event dissemination, in the case of a Wireless Mesh Network. We propose a protocol based on non-trivial forwarding mechanisms that employ network coding as a central tool for supporting adaptive event dissemination while exploiting the broadcast nature of wireless transmissions. Our results show that network coding provides significant improvements to event diffusion compared to standard blind dissemination solutions, namely flooding and gossiping.

Keywords: Network coding, publish/subscribe, wireless.

1 Introduction

This paper investigates the problem of event diffusion over a wireless mesh network (WMN) by leveraging a recent information dissemination technique called Network Coding; see [8] for a tutorial. The Wireless Mesh Network (WMN) is an emerging communication architecture with many practical applications in such areas as self-organizing community networks, industrial plant automation, wireless sensor networks, etc., [1]. A WMN can be considered as a two-tier architecture. The first tier is a wireless backbone composed of mesh routers capable of packet routing and optionally providing gateway functionality. The second tier is composed of mobile and/or portable wireless devices (e.g. WiFi-enabled smart phones, mobile TV devices, etc.) which can act as clients. A WMN is a self-organizing network with a certain degree of variability in terms of participants and topology. For example, clients can move, new clients can join a network, mesh routers can be occasionally switched off, or some clients can at times act as wireless routers. Having a suitable application level abstraction that can face with such a changes is thus very appealing. In this regards, publish/subscribe (pub/sub) is a mature interaction paradigm that fits such requirements, since it allows for reference-decoupled and asynchronous interactions among the participants [7]. In a pub/sub communication system publishers produce information in form of events and subscribers receive the subset of events that match their interests, expressed as a filter. Pub/sub

[1] This paper was supported by the EU STREP SM4ALL FP7-224332.

A.V. Vasilakos et al. (Eds.): AUTONOMICS 2009, LNICST 23, pp. 17–31, 2010.

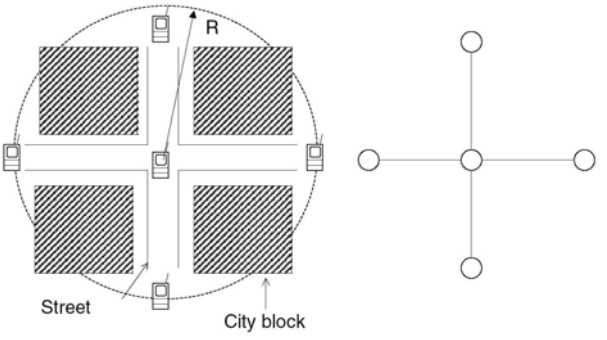

Fig. 1. The grid topology arising from a metropolitan deployment of a WMN

systems have been widely studied in wired a setting, e.g., SIENA [6],Gryphon [11], LeSubscribe [16]. However, while some papers have also focused on pub/sub systems running over networks exploiting wireless technology, e.g. [3], [13], only a very few of them have considered WMNs, [10], [21].

We consider a WMN deployed over a Manhattan like city model, see [1], in which mesh routers can be considered as approximately placed at the intersection of two streets. Since the streets are running est-west and north-south, mesh routers form a regular grid topology, Figure 1. We assume that mesh routers are used as a dispatching structure for supporting event diffusion. This solution is borrowed from the proposal presented in [10]. We assume that each mesh client can communicate with only one mesh router (called its local mesh router), and mesh routers are equipped with additional software appliances that clients interact with. Essentially, when the publisher needs to publish a new event, it contacts its local mesh router and then sends the event to it. The mesh router diffuses the newly event to all the other routers in the network, on behalf of the publisher. A subscriber periodically renews its subscription to its current local mesh router for a specific period of time, thus implementing a lease mechanism. Filtering is done at the mesh router, and filters are not propagated into the network. A router notifies the client as soon as it receives an event matching the filter, given that the client subscription has not expired. In the rest of the paper we refer to a mesh router as a node.

1.1 Contribution of the Work

The contribution of the paper is the proposal of an event dissemination protocol suitable for dynamic environments. The protocol is self tuning in that (i) the behavior of a single node depends on the amount of information is being received as well as on the number of neighbors of a node (node density), (ii) the protocol runs efficiently independently of how many targets there are in the system and where they are located.

The rest of the paper is organized as follows. Section 2 presents a brief tutorial on the main concept of network coding and discusses basic alternatives to implement event diffusion in a wireless mesh setting. Section 3 presents the details of our network-coding

based protocol, and Section 4 provides several evaluation results. Finally, conclusions are given in Section 5.

2 Background

Network coding is a relatively recent technique for end-to-end information delivery in communication networks, introduced in the seminal paper of Ahlswede et al., [2] and advanced by others [12] for many applications. Network coding marks a clear departure from the basic network role as a passive relay of data packets or frames, to a more active model in which network nodes can perform algebraic operations on the data before sending it out. With network coding the intermediate nodes between source of information and the destination(s) do not simply relay the received packets. Rather, they are allowed to combine (encode) incoming data in order to generate the data output to be forwarded. The original key advantage of this intermediate combination is for data broadcasting and multicasting. With network coding a source node can always send data at the network's broadcasting rate, while without network coding this is not possible in general. Some concrete examples of network coding based multicast protocols can be found in [14], [15], [5] and [4]. Additional references can be found in [19].

In the following we adopt a linear network coding approach in which operations on packets are confined to algebraic operations over a finite field. More precisely, we confine ourselves to the Galois Field $GF(2^w)$ and interpret each data packet as being composed from a set elements of the field, each of size w bits. We restrict ourselves to apply linear network coding to the problem of broadcasting an original data packet, X, from a source node (e.g. the mesh router on behalf of the publisher) to all the other nodes of a wireless network. The problem solution can be easily generalized to multi-source multicast under reasonable additional constraints. The main symbols used throughout the paper are listed in Table 1.

In linear network coding, the basic operation performed by each network node is generating linear combinations of incoming packets, and transmitting the new "coded" packet. A linear combination is carried over a fixed set of original data *chunks*, called a *generation* of the original packet. More precisely, we assume that special designated nodes split an original data packet X of length l into m chunks, x_i, each of length l/m,

Table 1. Definition of the main symbols

E	Event to be diffused
m	Generation size
x	Original chunk of data
X	Vector of the original m chunks
y	an encoded chunk
Y	Vector of encoded chunks
α	Random coefficient
A	$m \times m$ matrix of random coefficients (decoding matrix)
EV	Encoding vector, coefficients used to create a linear comb.
IV	Information vector, an encoded chunk sent into a packet

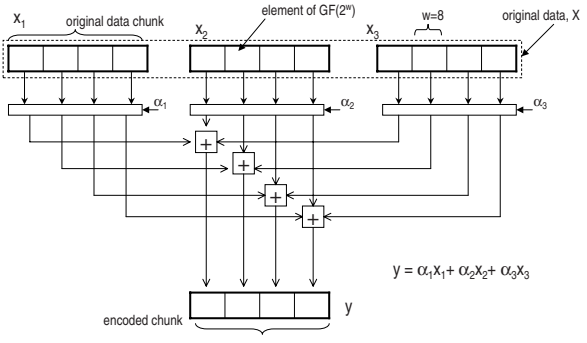

Fig. 2. The operation of linear combination over the original chunks of data

to form the generation $\{x_1, x_2, \ldots x_m\}$. Each chunk of data is composed of k elements of $GF(2^w)$. Hence, $l = k \times m \times w$ (0s are padded if required). The value m is called the *generation size*.

Consider for sake of example only, a data packet X of size 12 bytes, generation size $m = 3$ and element size $w = 8$ bits, or one byte (see Figure 2). The packet is divided into 3 chunks, x_1, x_2, x_3, each composed of 4 elements (bytes). A linear combination is achieved by choosing 3 coefficients of length w bits, $\alpha_1, \alpha_2, \alpha_3$ and computing a new *encoded* chunk

$$y = \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3$$

Because all operators are defined over the finite field $GF(2^w)$, the above computation is performed element-wise, i.e., if x_j^i is the i-th element of chunk j and y_i is the i-th element of the linear combination we have

$$y_i = \alpha_1 x_1^i + \alpha_2 x_2^i + \alpha_3^i x_3 \quad i = 1, 2, 3$$

However, for the sake of avoiding cumbersome notation, we do not make this replication explicit, and a linear combination is expressed as

$$y = \sum_{i=1}^{m} \alpha_i x_i$$

The result y is called an *encoded chunk* or, when it is sent in a packet, an *Information Vector*, IV. The set $EV = [\alpha_1, \ldots, \alpha_m]$ of coefficients used in the combination is called the *Encoding Vector*. As common with other network coding schemes, we assume that a node sends both the information vector IV and the associated encoding vector EV. Moreover, the coefficients used in a linear combination are generated randomly by the node.

Note that the overhead due to sending the above encoding vector is $m \times w$. Assuming typical values of $m = 16$ and $w = 8$ bits when sending a 1 KB data packet, the overhead is only $16/1024 \approx 1.5\%$.

An important aspect of linear network coding is that encoded chunks can themselves be combined to generate new encoded chunks at an intermediate node. For example,

to combine n encoded chunks, $y_1 \ldots y_n$, a node uses n coefficients, say $\alpha'_1 \ldots \alpha'_n$ with which it generates

$$y' = \sum_{j=1}^{n} \alpha'_j y_j$$

Note that the newly encoded chunk, y', also represents a linear combination with respect to the original data chunks. Therefore, since

$$y' = \sum_{j=1}^{n} \alpha'_j (\sum_{i=1}^{m} \alpha_{ji} x_i) = \sum_{i=1}^{m} (\sum_{j=1}^{n} \alpha'_j \alpha_{ji}) x_i$$

where α_{ji} is the i-th coefficient used to calculate y_j, the encoding vector the intermediate nodes is in effect

$$EV = [\sum_{j=1}^{n} \alpha'_j \alpha_{j1}, \ldots, \sum_{j=1}^{n} \alpha'_j \alpha_{jm}]$$

2.1 Event Dissemination with Network Coding

To elucidate the advantage of network coding, we present several alternatives for disseminating (diffusing) an event over a portion of a wireless grid in which each node can reach four neighbor nodes to the north, south, east and west. For this purpose, we consider the 2-dimensional grid topology of Figure 3 and explain several alternatives for disseminating an event E generated by the *source* node S (located at the center of the grid) using the smallest amount of data transmissions. The problem we solve is how to ensure that the event reaches the four *destination* nodes at the four corners of the grid. Nodes 1, 2, 3, 4 function as *relay* nodes in this example.

For the sake of simplicity, we assume here an idealized collision-free broadcast communication channel and that event E fits the size of a single packet (a realistic channel is used in simulations). Our aim is to compare the performance/cost tradeoff of different principle design, where the cost is the total amount of data sent over the network, T, and the performance is measured through the probability P_F that *all* nodes receive E. In the following the cost of sending one packet (containing E) is counted as one.

In general, there are two different approaches for event diffusion: *informed* dissemination and *blind dissemination*. Informed Dissemination requires the source node S to know the topology and coordinate transmissions to the destination nodes. For example, in the case of Figure 3, S may declare, in the packet header, two destinations which have to rebroadcast the packet. For example, S can specify nodes 1 and 3 as destinations. This means relay nodes 2 and 4 discard the packet, while nodes 1 and 3 will rebroadcast their packet to the corner nodes. It is obvious that the cost of this solution is $T = 3$, and $P_F = 1$ (ignoring the small cost of sending the additional destination IDs in the packet header). Clearly, this is the smallest amount of data that must be sent for disseminating the event.

Blind dissemination is an attractive alternative which is more suited to the distributed dynamic nature of wireless mesh networks. In such environment, it may not be easy (or it may be very costly) to maintain the required topology or link-state information for

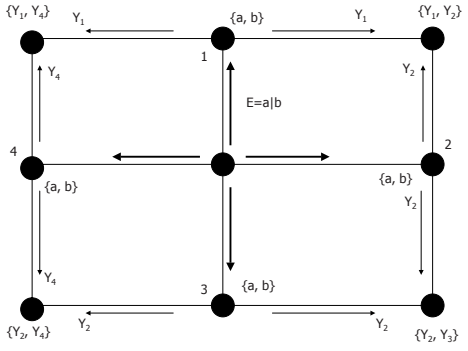

Fig. 3. Event diffusion over a square grid. The source node S is placed at the center.

disseminating events in the presence of node mobility, or some nodes turning off and new nodes turning on arbitrarily.

In this paper, we are interested mainly in blind dissemination. Next, we review several alternatives that avoid the use of coordination among nodes while still requiring the same amount of data transmissions as in informed dissemination. Normally, this comes at the cost of a lower value for P_F. In order to assess the benefit of applying network coding for event diffusion, we will explore and compare the performance of network coding to other blind dissemination techniques, namely flooding, naive dissemination, and gossip. We will briefly explain these techniques with the help of the wireless grid example of Figure 3.

NAIVE RANDOM DISSEMINATION. This technique exploits a very simple random forwarding approach in which coordination is no longer required. Consider the following variation of the approach. Initially, the source S transmits the event to its four neighbors (relay nodes). Then each relay node splits the packet (event E) into two parts, say a and b, of equal size. Now, each relay node chooses only one of these parts, i.e. either a or b, with equal probability, then transmits it to all destination it can reach. Since each part contains only half the information carried in E, the cost of sending one part of the event by a relay node is $\frac{1}{2}$. Thus, the total cost of dissemination is still $T = 3$.

To calculate P_F, let $\{y_1, y_2, y_3, y_4\}$ denote the scheduled parts chosen by the four relay nodes $1, 2, 3$ and 4, respectively. For example, the schedule $\{a, a, a, b\}$ refers to the schedule where relay nodes 1, 2, 3 each choose to transmit a, and 4 chooses to transmit b. For this example, there are 2^4 different schedules, but only two of them, namely $\{a, b, a, b\}$ and $\{b, a, b, a\}$ allow all the receiver nodes reconstruct the event. Thus, $p_F = 2^{-3} = 1/8 = 0.125$. This result shows the weakness of naive random dissemination.

RANDOM DISSEMINATION WITH NETWORK CODING. The second random solution leverages linear network coding, where we can show that we can achieve forwarding probability $p_F \approx 1$ while maintaining the cost of forwarding practically the same as with the informed dissemination. Instead of just sending one half of the event E randomly, a relay node now sends a random linear combination of the two parts a and b.

To compute the linear combination, the relay node picks two coefficients, α_1 and α_2, uniformly at random from a finite field of size q then broadcasts $y = \alpha_1 a + \alpha_2 b$ together with the coefficient's vector (α_1, α_1) (all Operations are defined over the field F_q). Note that the size of the linear combination $|y| = |a| = |b| = |E/2|$. The transmitted packet will have a size slightly larger than $|y|$ because we must include the coefficients in the packet. However, for most practical cases, the overhead due to including the coefficients in the transmission is small and can be neglected to simplify analysis.

Since a receiver node at one of the corners gets two linear combinations from its neighbors, say $y_1 = \alpha_{11} a + \alpha_{12} b$ and $y_2 = \alpha_{21} a + \alpha_{22} b$ with the corresponding 4 coefficients, it will be able to retrieve (i.e. decode) the original event E if it can solve the following linear system of equations:

$$\alpha_{11} y_1 + \alpha_{12} y_2 = x_1$$

$$\alpha_{21} y_1 + \alpha_{22} y_2 = x_2$$

There are q^4 different possible 2×2 matrices whose coefficients are picked randomly from the field F_q. Considering that the number of linearly independent matrices is $(q^2 - 1)(q^2 - q)$, the probability that the matrix above is non-singular, thus allowing a node to retrieve the event (by inverting the matrix, or using Gaussian elimination), is

$$\frac{(q^2 - 1)(q^2 - q)}{q^4};$$

from which the probability that all of the 4 receivers at the corners of the grid get the event is $P_F = \left[(1 - \frac{1}{q^2})(1 - \frac{1}{q})\right]^4$. For example, if we choose with $q = 256$, i.e. the field of 8-bit coefficients, then $P_F \approx 0.98$. Note, also that the overhead for sending two coefficients is just two bytes.

GOSSIP. In order to illustrate the powerful utility of network coding in random event dissemination, we compare it against a probabilistic flooding technique commonly referred to as *gossiping*. Using a basic gossip protocol, each relay node transmits the complete event E to its neighbors with probability p. Let

$$[tx_1, tx_1, tx_3, tx_4]$$

be the transmit decision of the four relay nodes such that $tx_i = 1$ means that node i decided to transmit the packet, and $tx_i = 0$ means it decided to discard the packet. There are two sets of decisions that allow for all four receivers at the corners of the grid to get the event. They are

$$TX = [1, *, 1, *] \quad TX' = [*, 1, *, 1]$$

where $*$ means any decision. In other words, all receivers will get E when either (at least) both relay nodes 1 and 3 transmit E, or both nodes 2 and 4 transmit E. A decision belongs to one of these 2 sets with probability p^2. Note that the decision $1, 1, 1, 1$ is common to both sets and occurs with probability p^4. Therefore, the probability that all receivers get the event is the probability to observe a retransmission pattern of type TX or TX' on the relay nodes, or

$$P_F = 2p^2 - p^4$$

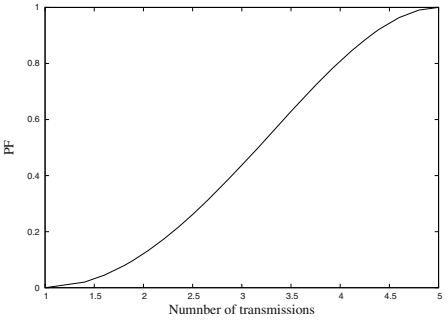

Fig. 4. Probability that all nodes receive the event, P_F vs total number of transmissions, T

To compare gossiping to RDNC, we first observe that in gossiping, the average number of transmissions is $T = 1 + 4p$, giving us $p = (T - 1)/4$, where T is the cost of dissemination. Figure 4 shows the complete reception probability P_F as a function of T for the gossip technique. The above result shows that if we are to maintain the cost at $T = 3$ transmissions, then we get $p = 0.5$; from which we determine $P_F \approx 0.43$. On the other hand, in order for gossip to achieve the same dissemination reliability as network coding, i.e. $P_F \approx 0.98$, we need to set $p \approx 0.9$, which means the cost will increase to $T \approx 4.6$ transmissions.

3 Proposed Protocol

The prosed protocol utilizes a push-pull method whereby the event is diffused in two phases. During the first phase, the information content of the event is partially pushed throughput all the network. During the second phase the fraction of nodes that need to fully decode the event pull the missed information from their neighbor nodes.

The key idea of the protocol is to assure that although a single node doesn't have the whole information required to detect the event, the remaining part can be retrieved from nearby nodes. Roughly speaking, the pushing phase is such that any group of nodes, which is composed by a node and its neighbors, has the full information required to decode the event.

In the first phase of the protocol a node adapts its behavior according to the amount of information being received. A kind of self adaptation is also incorporated in the second phase, where a feedback mechanism is used to adapt the reaction of each node according to the current node density.

3.1 Basic Data Structure and Assumption

Each node manages the following data structures

- $m \times m$ matrix, A, called Decoding Matrix containing the elements of the encoding vector
- $1 \times m$ vector, Y, called the local encoded data vector, which contains the encoded chunks

- Operation bit mode, op (normal or collecting)
- Transmission counter, C

A node has direct access to the local broadcast link layer primitive, `bcast(P)`, with which it can send a packet P to all its neighbor nodes, namely, the ones that are within its transmission range. A packet P contains the following fields

- Information Vector, $P.IV$
- Encoding Vector, $P.EV$
- Generation ID, $P.gen$
- Topic ID, $P.topic$

We assume that an event E fits the size of a packet (a simple variation allows to overcome this limitation). Each event E is uniquely identified through the concatenation of a generation number, managed by the publisher, and the publisher's ID.

3.2 Protocol Description

In our algorithm the nodes are classified according to their roles as:

- Source node (the node that sends the packet carrying the event to be diffused)
- Bootstrap nodes (the neighbors of the source)
- Intermediate nodes (all the of other nodes)

An intermediate node may operate into two different modes: *normal* ($op = 0$) mode and *collecting* mode ($op = 1$). The nodes initially operate in the normal mode.

The proposed protocol uses the following four key parameters to define a flexible network-coding specific forwarding policy. These parameters can be tuned to optimize the performance of the forwarding policy on a wireless mesh. The parameters are:

- BF, Bootstrap Factor
- ΔT_C, Collecting Time
- FF, Forwarding Factor
- $MaxTx$, Maximum number of allowed transmissions

Push phase. The actions performed by each node in the first push phase are the following ones.

PUBLISHER. When the publisher needs to send a new event E, it issues the publish primitive on the local node (router), say node S. Node S acts as a source of the event on behalf of the publisher, and sends E by a local broadcast indicating the generation ID.

BOOTSTRAP NODES. When a bootstrap node, B, receives event E from S, it splits the event into m chunks, $x_1, \ldots, x_m$. Then, B executes BF times the following four steps:

1. generate m random coefficients, $EV = [\alpha_1, \ldots, \alpha_m]$
2. compute the linear combination $y = \alpha_1 x_1 + \ldots + \alpha_m x_m$
3. prepare a new packet with $P.IV = y, P.EV = EV$
4. send P via the local broadcast primitive.

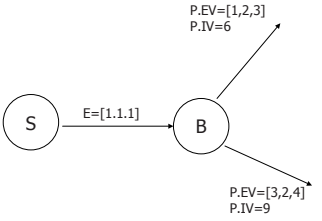

Fig. 5. The initial phase of event diffusion

The Bootstrap Factor parameter, BF, varies in the range $[1..m]$ and determines the number of linear combinations sent by a bootstrap node.

Figure 5 shows an example for $m = 3$. For simplicity, we have assumed that E is composed of 3 bytes, each of value 1. The bootstrap node B generates two linear combinations with the coefficients $[1, 2, 3]$ and $[3, 2, 4]$. Thus, the bootstrap factor is 2.

INTERMEDIATE NODES. When an intermediate node, say I, receives a packet P containing a chunk for a new event, it creates a new decoding matrix, A, associated with the event. The matrix contains all 0s, except for the first row which contains the coefficients carried in the EV. The node also creates a new local encoded data vector, Y, containing all 0s in all entries except the first one, which stores the IV of packet P. Finally, it sets the transmissions counter C to $MaxTx$.

After these steps, I enters into the collecting operation mode ($op = 1$) and starts a new *collecting phase*. A new collecting phase is initiated each time the node enters this mode. During a collecting phase I waits for other possible new innovative chunks if order to decide how many linear combinations to send.

The Collecting time parameter, ΔT_C, determines the minimum duration of the collecting phase. If a new innovative packet is received while the node operates in the collecting mode, say at time t, then the duration of collecting phase will be postponed until time $t + \Delta T_C$. Should a new innovative packet arrive before the new deadline, the collecting phase will be again deferred by ΔT_C.

The application of collecting-time is sketched in Figure 6. Node I has received two packets, $P1$ and $P2$, carrying two linearly independent combinations over the original chunks. The decoding matrix thus contains the corresponding encoding vectors. Packet $P1$ triggers the collecting phase, while $P2$ prolonged the phase.

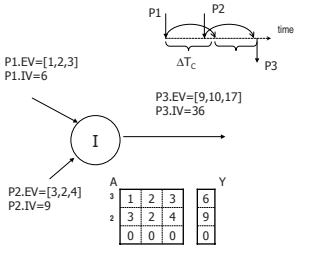

Fig. 6. Generating linear combination at an intermediate node

After ΔT_C from when $P2$ was received, the node does the following: create a new linear combination using the coefficients 3 and 2 (they are shown to the left of the matrix in figure 6), create a new information vector whose value is $3 \times 6 + 2 \times 9 = 36$, create the new encoding vector, $3[1, 2, 3] + 2[3, 2, 4] = [9, 10, 17]$, then send the new packet P3.

The reason for not immediately sending new linear combinations is that we have empirically seen by simulations that innovative packets tend to arrive as burst[1]. As the generation of a new linear combination is more useful if it is done over a wider number of chunks, deferring the generation time - in the hope of increasing the number of chunks - is a simple pragmatic way for improving the benefit of network coding. In fact, the newly generated combination is more likely to be independent from the ones stored at a larger number of neighbors.

The prolongation of the collecting phase thus acts a very simple adaption mechanism that uses the reception of a new innovative chunk as an indicator that further useful chunks are likely to be received in the near future. Please note that if the rank of the decoding matrix at the beginning of a collecting phase is r, then the node can receive no more than $m - r$ new linear combinations; thus, the overall duration of the collecting phase cannot exceed $(m - r) \times \Delta T_C$.

At the end of the collecting phase the node sends out $k = min\{\lceil FF \times r \rceil, C\}$ new combinations, where r is the total number innovative packets collected during the collecting phase. The parameter FF, called the Forwarding Factor is a real number in the range $(0..1]$. The forwarding factor regulates the "verbosity" of a node, and can be useful in reducing the amount of data flooded in the network, or for conserving energy by reducing the amount of transmissions from a node. Finally the node I decreases the transmission counter C of k. The push phase ends when $C = 0$.

3.3 Pull Phase

The pull phase is initiated by a node A after ΔT_P time unit from when $C = 0$ (recall that each packet carries the topic identification so A triggers this phase only for chunks belonging to events of interest).

To pull the missed chunks, node A broadcasts a requesting message to its neighbors. This message contains the event ID and the number of missed chunks, say c, required for full decoding. Upon receiving such a request, say at time t, a A's neighbor, say B, schedules the transmissions of c linear combinations at time $t + \Delta T$, where ΔT is a random jitter picked in the interval $(0, T_J)$ (T_J is maximum jitter). During ΔT, B listen for possible updates sent by A. A sends an update message each time it receives some innovative chunks from any of its neighbors. An update message contains the number $c' < c$ of innovative packets that A now wishes to receive. In such an update message is received, B adjusts the scheduled number of transmissions to c'. Clearly, the transmissions are cancelled when $c' = 0$. Thus for example, a request message with $c = 3$ triggers the scheduling of 3 linear combinations at each A's neighbors (suppone they are they B and D). Let assume that a neighbor D sends 3 chunks. If after the reception of the new chunks A needs, say $c' = 1$ more new chunks, A will send a new requesting message containing this new value. In this way B decreases its contribution from 3 to 1 chunk.

[1] We have used the nam animation tool shipped with ns-2.

4 Evaluation

This section reports an in-depth performance evaluation study of the proposed protocols carried out by extensive simulations using ns-2.31 simulation tools [18]. We used a 914 MHz Lucent WaveLAN DSSS radio interface model available in the simulator. Each node has a transmission range R. We used a two-ray ground refection model as the wireless propagation model. Each reading of the simulation was taken after 100 independent runs. We have simulated 400 nodes arranged on a grid at distance of 250 meters from one other. The transmission range is either fixed to $R = 250m$ (with connectivity degree = 4 nodes), or to $R = 750m$ (connectivity degree = 20 nodes). The duration of each simulation was 500 seconds. The source of the event is placed at the center of the grid and transmits a new packet of $1KB$ in size, every 1 s. The default collecting time is $\Delta T_C = 50\ ms$ while the maximum jitter is $T_J = 10\ ms$. The target nodes are placed *at random*.

Arithmetic operations are performed on the Galois field 2^8. We have used the library available at [9]. To speed up the simulation, the decoding matrix is managed in the Gaussian triangular form. We say that a node decodes the event when the associated encoding matrix has full rank. Using this simulation environment, we provide estimates of the following performance metrics,

- Percentage of Decoding: percentage of nodes that successfully decode the event.
- Decoding Delay: the amount of time elapsed from when the event is generated until it is decoded
- Cost of Diffusion: total number of bytes sent transmitted in the network for disseminating the event

4.1 Protocol Tuning

Our first set of experiments concerns the effect of collecting time and the generation size on the protocol performance. In these preliminary tests we have considered a push-only protocol, i.e., nodes have no limit on the number of transmissions they are allowed to perform.

The left plot in Figure 7 shows the full decoding probability as a function of the generation size and the collecting time given as a parameter. The decoding probability is highly affected by the collecting time. When the decoding phase is not applied, $\Delta T_C = 0$, the decoding probability is very low, especially for small generation size. The decoding probability increases with the collecting time, meaning that waiting allows a node for collecting a larger amount of innovative information and then generating linear combinations that are useful for a larger number of neighbors.

The right side plot in Figure 7 shows the cost as function of the generation size and the collecting time as a parameter. The cost increases with the collecting time since a longer collecting time allows for more node to full decoding and then sending data.

4.2 Performance on a Grid

The left side plot in Figure 8 shows the total per event diffusion cost as a function of the number of subscribers for a 20×20 grid with connectivity 4. The cost of the proposed protocol has been compared against three not adaptive protocols: flooding, gossip

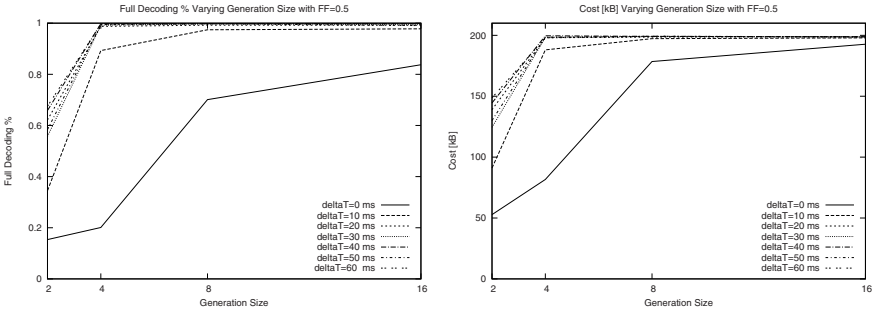

Fig. 7. Full decoding and cost vs generation size, collecting time is a parameter

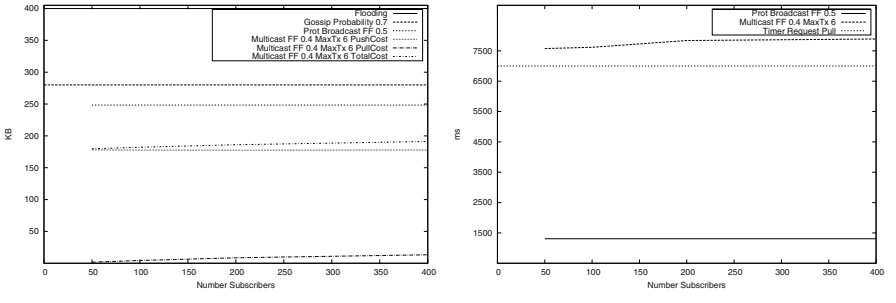

Fig. 8. Cost performance in a 20×20 grid, connectivity 4. Number of bytes sent (left), decoding delay (right).

and a push-only protocol with forwarding factor FF=5 and no limit on the number of transmissions (labelled Prot Broadcast $FF = 0.5$). In the gossip protocol a node sends the packet with probability $p_c = 0.7$. The proposed $multicast$ push-pull runs with $FF = 0.4$, $MaxTx = 6$.

The cost of flooding is always 400 KB as all nodes send the packet whereas the cost of gossip is 280 KB due to the probabilistic transmission. The cost of our protocol is composed of a fixed part related to the pushing phase, and a variable part due to the pulling phase, which increases linearly with the number of subscribers. The cost is always below the gossip's one.

The right side plot in Figure 8 shows the total decoding delay. The push-only protocol is used to estimate an upper bound on the delay of the push-phase. The total decoding delay is dominated by the time after which a node initiates the pulling phase. In our simulations the pull phase starts after $\Delta T_P = 7$ s. This high value was selected for clearly distinguish the pull phase component of the delay. From the plot we can see that the pull phase requires approximatively 1 s to terminate. As the push phase requires no more than 1.5 s., the total decoding delay can be as low as 2.5 sec.

To test the self-adapting mechanism of the pull phase, we have conducted a set of experiments considering the higher connectivity degree of 20 neighbors. The parameters used in this new setting are $FF = 0.3, MaxTX = 3$. These values are determined

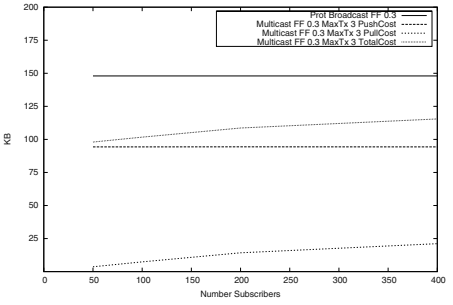

Fig. 9. Cost performance in a 20×20 grid, connectivity 20

empirically from the simulation and provide near optimal performance. The cost, see Figure 9, shows a similar behavior and it always is lower than the cost of a push-only protocol.

5 Conclusions

In this paper we have proposed an adaptive network-coding based event-diffusion protocol for wireless mesh networks. The proposed partial diffusion protocol uses a push-pull method to reduce the initial dissemination cost (amount of data sent), but adds a recovery cost incurred by the subscriber nodes. Compared to other blind dissemination protocols, most notably probabilistic flooding or gossip, our protocol has the advantage that it requires significantly smaller amount of data to be transmitted into the network. This advantage can be especially important for situations when nodes have limited energy and/or bandwidth, such as sensor networks, or battery-powered wireless nodes. Even in wireless meshes harnessed with higher-bandwidth IEEE 802.11 wireless routers, reducing the amount of data transmitted is necessary in broadcast environments to reduce contention on channels, since collisions can lead to excessive waiting delays or dropping connections.

Wireless mesh networks can especially benefit from our protocols because the nodes are not only transmitting and receiving their own data, but are also involved in relaying data between close and distant neighbors.

References

1. Akyildiz, I.F., Wang, X., Wang, W.: Wireless Mesh Networks: A Survey. Computer Networks Journal (March 2005)
2. Ahlswede, R., Cai, N., Li, S.-Y.R., Yeung, R.W.: Network Information Flow. IEEE Transactions on Information Theory, IT-46, 1204–1216 (2000)
3. Baldoni, R., Beraldi, R., Cugola, G., Migliavacca, M., Querzoni, L.: Structure-less content-based routing in mobile ad hoc networks. In: Costa, P., Picco, G. (eds.) IEEE International Conference on Pervasive Services, ICPS (2005)

4. Deb, S., Medard, M., Chote, C.: Algebraic Gossip: A network coding approach to optimal multiple rumor mongering. IEEE/ACM Transactions on Networking, 2486–2507 (June 2006)
5. Mosk-Aoyama, D., Shah, D.: Information Dissemination via Network Coding. In: IEEE Intl. Symp. on Info. Theory (2006)
6. Carzaniga, A., Rosenblum, D., Wolf, A.: Design and evaluation of a wide-area event notification service. ACM Trans. Comput. 19(3), 332–383 (2001)
7. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.M.: The many faces of publish/ subscribe. ACM Computing Surveys 35(2), 114–131 (2003)
8. Fragouli, C., Le Boudec, J.-Y., Widmer, J.: Network Coding: An Instant Primer. In: ACM SIGCOMM 2006 (2006)
9. http://www.partow.net/projects/galois/
10. Gavidia, D., Voulgaris, S., van Steen, M.: A Gossip-based Distributed News Service for Wireless Mesh Networks. In: Proc. Third Int'l Conf. Wireless On-demand Network Systems and Services (WONS) (January 2006)
11. Gryphon Web Site, http://www.research.ibm.com/gryphon/
12. Ho, T., Mdard, M., Effros, M., Karger, D.: The benefits of coding over routing in a randomized setting. In: Proc. IEEE Symp. Information Theory, Yokohama, Japan (June/July 2003)
13. Ionescu, M., Marsic, I.: Stateful Publish-Subscribe for Mobile Environments. In: ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots, WMASH (2004)
14. Katti, S., et al.: XORs in The Air: PracticalWireless Network Coding. In: ACM Sigcom (2006)
15. Lin, Y., Li, B., Liang, B.: Stochastic Analysis of Network Coding in Epidemic Routing. JSAC 26(5) (June 2008)
16. Preotiuc-Pietro, R., et al.: Publish/subscribe on the web at extreme speed. In: Proc. of ACM SIGMOD Conf. on Management of Data, Cairo, Egypt (2000)
17. Mansouri, M., Shah Pakravan, M.R.: Network Coding Based Reliable Broadcasting in Wireless Ad-hoc Networks. In: 15th IEEE International Conference on Publication Networks (ICON 2007), November 2007, pp. 525–530 (2007)
18. http://www.isi.edu/nsnam/ns/
19. http://tesla.csl.uiuc.edu/koetter/NWC/
20. Wu, Y., Chou, P.A., Kung, S.-Y.: Minimum-Energy Multicast in Mobile Ad Hoc Networks Using Network Coding. IEEE Transaction on Communications 53(11) (November 2005)
21. Zheng, Y., Cao, J., Liu, M., Wang, J.: Dept., Efficient Event Delivery Publish/Subscribe Systems forWireless Mesh Networks. In: Proc. of IEEE Wireless Communications and Networking Conference, WCNC (2007)