Event Diffusion in Wireless Mesh Networks using Random Linear Network Coding

Roberto Beraldi, Aurelio Forese
DIS - Università di Roma "La Sapienza"
Via Ariosto, 25- 00185 Roma, Italy

Hussein M. Alnuweiri
Texas A&M University at Qatar
Education City, Doha, Qatar

1. INTRODUCTION

In this paper we face the problem of efficient event dissemination in a pub/sub system running on a Wireless Mesh Network (WMN). We assume that filtering is done at the receiving side. We present a diffusion protocol based on a recent technique, called random Linear Network Coding (LNC), which potentially allows for reducing the amount of data in the network per event diffusion at the theoretical minimum. The WMS is modelled as a regular grid topology. For a background network coding and its applications the interested reader can refer to the network coding's home page[1].

2. PROTOCOL DESCRIPTION

We assume that an event E fits the size of a packet and that the event is uniquely identified. When the publisher S needs to send a new event E, it sends E by a local broadcast indicating the generation ID.

A neighbor of the source node is called a bootstrap node. When a bootstrap node, B, receives E from S, it splits the event into m chunks, $x_1, \ldots, x_m$. Then, B repeats BF (Bootstrap Factor) number of times the following actions. It generates m random coefficients, $EV = [\alpha_1, \ldots, \alpha_m]$, computes the linear combination

$$y = \alpha_1 x_1 + \ldots + \alpha_m x_m$$

prepares a new packet P, containing the fields $P.IV = y$ (IV = Information Vector) and $P.EV = EV$ (EV = Encoding Vector), and sends P via the local broadcast primitive.

Figure 1 shows a simple example for $m = 3$. The keep the example simple, we have assumed that E is composed of 3 bytes, of value 1. B generates two linear combinations with the coefficients $[1, 2, 3]$ and $[3, 2, 4]$. The bootstrap factor is then 2.

The other nodes of the network are called intermediate nodes, I, and behave as follows. When a node I receives a packet P for a new event, it creates a new decoding matrix, A, associated to the event. The matrix contains all 0s, except for the first row, which contains the coefficients carried in the encoding vector. The node also creates a new local encoded data vector, Y, containing all 0s in all elements but the first one, which stores the P's information vector.

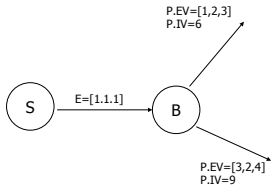

Figure 1: The initial phase of event diffusion.

With probability FP, called forwarding probability, the node schedules the transmission of a new linear combination after a time interval, ΔT_C, called the Collecting Time. If a new packet for the same generation, say P', arrives before the time interval is elapsed, the node tests whenever the encoding vector stored in P' is linearly independent from the rows already stored into its decoding matrix. If such is the case, the coefficients carried into P', are added at the second row of the matrix and, with probability FP, the transmission of linear combinations are scheduled after ΔT_C starting from the reception time. This procedure is repeated for any packet that should arrive within the collecting time.

A packet carrying an information vector which is linearly independent from the encoded chunks stored at a node, is called an *innovative packet* for that node. An innovative packet thus increases the rank of the decoding matrix of one unit.

If P' doesn't provide an independent linear combination, no actions are taken. After ΔT_C time units from when P' is received, the nodes transmits k new linear combination.

Figure 2 shows an example. Node I has received two packets, $P1$ and $P2$, carrying two linearly independent combinations over the original chunks. The decoding matrix contains thus contains the corresponding encoding vectors.

After ΔT_C from when $P2$ was received, the node creates a new linear combination using the coefficients 3 and 2 (they are shown close the rows of the matrix), creates a new information vector whose value is $3 \times 6 + 2 \times 9 = 36$, creates the new encoding vector, $3[1, 2, 3] + 2[3, 2, 4] = [9, 10, 17]$, and then sends the packet P3.

[1] $http://www.ifp.uiuc.edu/\ koetter/NWC$

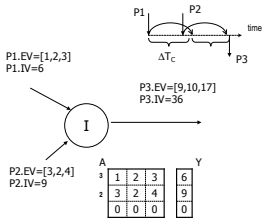

Figure 2: Linear combinations at an intermediate node.

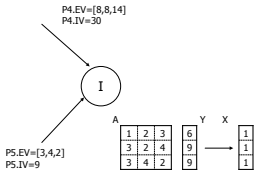

Figure 3: Decoding at an intermediate node.

When a node collect enough encoded chunks, namely when the rank of the decoding matrix becomes m, i.e., it is full, the node can retrive the original data chunks and from here the original event, E. The event is notified to all subscribers whose filters are matched by the event.

After a node has decoded the original data packet it deletes the decoding matrix and the encoded chunks. Any subsequent chunk for the same event is ignored; for this purpose, a node stores the ID of the last decoded events (this list is pruned after a certain amount of time).

For example, Figure 3 shows what happens when the intermediate node I receives two more packets, $P4$ and $P5$. The first packet doesn't increase the matrix's rank, and thus it is simply discarded. On the contrary, $P5$ carries the missing part of the original information. The decoding matrix has now full rank and the node can retrive the original data, i.e, $X = [1.1.1]$.

Our algorithm has four main parameters that allow for regulating the forwarding strategy, i.e, when a node has to send a new linear combination. These parameters are:(i) Collecting Time, ΔT_C; (ii) Bootstrap Factor, BF; (iii) Forwarding Probability, FP; (iv) Forwarding Factor, FF.

The collecting time determines the time interval during which no other innovative packets have to be received in order for the node to send linear combinations.

The Bootstrap Factor, BF, is an integer number which varies in the range $[1..m]$. It determines the number of linear combinations sent by a bootstrap node. If the publisher has n neighbors, then it must be $BF \times n \geq m$, otherwise the number of chunks spread into the network are not enough to decode the original packet.

The Forwarding Probability, FP, is used to determine if an innovative packet will further delay the transmission of a linear combination. When an innovative packet is received, the sending of linear combinations are delayed of ΔT_C with probability FP. Note that even if a node has received h innovative packets it can happen that no transmissions take place at all. This happens with probability $(1 - FP)^h$.

FF is a real number in the range $(0..1]$. If a node has received ip innovative packets, then the node sends $k = \lceil FF \times ip \rceil$ linear combinations of all the encoded chunks it currently has.

3. SOME RESULTS AND FUTURE WORK

Figure 4 shows the percentage of nodes that can decode the event as a function of the forwarding probability, for a grid composed of 400 nodes and connectivity degree 4. The Forwarding Factor is a parameter, while the generation size is $S = 2$. The source of the event is positioned at the center of the grid and emits a new event of $1KB$ in size, every 1 s. The collecting time is $\Delta T_C = 50\ ms$. Arithmetic operations are performed on the Galois field 2^8. A gossip protocol is used for comparison purpose.

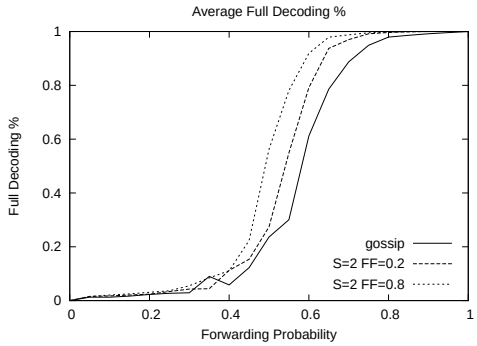

Figure 4: Full decoding vs forwarding probability.

Figure 5 shows the cost (total KB sent in the network) as a function of FP. We can observe how using our protocol the consumed bandwidth is roughly half the one required by a gossip protocol. These results confirm that LNC is a appealing solution for event diffusion in a wireless setting.

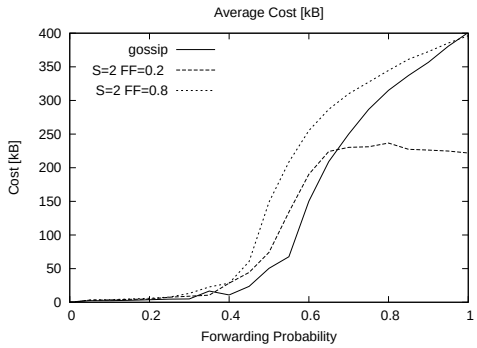

Figure 5: Cost vs forwarding probability.

We are currently investigating how the protocol behaves in a more general scenario, in which the topology varies with the time and events are generated at different sites.