# Effect of Network Coding on Buffer Management in Wireless Sensor Network

Sunil Jardosh, Narmawala Zunnun, Prabhat Ranjan, Sanjay Srivastava
*ICT, DA-IICT*
*Gandhinagar, India,*
*{sunil_jardosh,prabhat_ranjan,narmawala_zunnun,sanjay_srivastava}@da-iict.ac.in*

*Abstract*—**In resource constrained wireless sensor networks (WSN), it is highly desirable to make efficient use of available buffer. Hence for the WSN designed for the monitoring applications, buffer management is a key requirement at sensor nodes. We have proposed an efficient buffer management scheme based on random linear network coding as the in-network processing on data packets. With buffer allocation from source to sink path, our scheme distributes the buffer requirement among the nodes on the path. Further in the case of a packet loss, the proposed scheme recovers packet from available information distributed on the path from source to sink. We have compared our scheme with conventional buffer management scheme. Results show that network coding based buffer management scheme has better buffer availability with less redundancy and reduced loss recovery cost.**

***Keywords-component; Wireless Sensor Network, Random Linear Network Coding, Buffer Management***

## I. INTRODUCTION

Wireless sensor network is application specific data centric network [1-2], One of the application specific classes of WSN is monitoring application class [3-5]. Monitoring applications perform the micro-level monitoring of sensor field. Here network follows the specific data traffic pattern of many to one in which all the sensor nodes send their data towards the base station. Sensor nodes sense sensor field periodically or on some predefined events and generate readings. These readings are forwarded towards the base station using store and forward method.

For this class of applications, buffer is very constrained resource. So effective buffer management and maintaining minimum number of duplicate packets in the network are of prime importance. Further buffer allocation at each node should be fair among different sources. To achieve these goals, we have proposed a buffer management scheme using in-network processing for WSN deployed for monitoring applications like [3-5]. Our objective here is to distribute the buffer load among nodes and provide uniform buffer utilization. That helps in reducing the buffer requirement at source and intermediate nodes. Secondly, we also minimize the number of retransmitted messages required in the case of packet loss. We have used network coding as "in-network processing" on packets from source to sink [6-7]. In normal network, each packet carries its own information while network coding distributes information of packets in a generation to encoded packets of that generation. We have used this property of network coding to distribute the information of source buffer along the path from source to sink.

Rest of the paper is organized as follows. Section 2 covers the network coding mechanism and related work. A detailed example of random linear network coding (RLC) is given in appendix. Section 3 contains our buffer management scheme. Simulation results and analysis are provided in section 4 and we conclude our paper in section 5.

## II. NETWORK CODING AND RELATED WORK

Ahlswede et al. in their pioneering work [8] showed that maximal network flow and sum of cut-edge capacities of any minimal cut that separates sender and receiver are equal and it is the maximum achievable capacity. This capacity is generally not achievable in conventional communication network which considers information as a 'fluid' which can simply be routed or replicated. i.e., independent information streams may share network resources but they are kept separate. In network coding, instead of forwarding packets as it is, nodes may recombine two or more input packets into one or more output packets. Li et al. in [9] showed that linear coding with finite field size suffices to achieve maximum flow from the source to each destination. Among various linear encoding techniques at network nodes, random linear coding [10] is widely used as it is distributed and less computationally intensive and algorithms are well understood. The following explanation is based on [11].

### A. Random Linear Coding

Let each packet contains L bits. If packets to be combined are not of the same size, smaller packets are padded with trailing zeroes. s consecutive bits of a packet can be interpreted as a symbol over the field $F_{2^s}$. i.e., each packet consists of a vector of L/s symbols. Outgoing packet at each node is a linear combination of incoming packets or generated packets at that node where addition and multiplication operations are performed over the field $F_{2^s}$. The encoded packet also contains L bits. So an encoded packet contains information about all original packets and multiple such packets can be generated. In effect, information of an original packet is spread into number of encoded packets.

### B. Encoding

Let original packets $M^1, \ldots, M^n$ be generated by one or more sources. The encoded packet is $X_p = \sum_{i=0}^{n} g_i M_p^i$, where sequence of coefficients $g_1, \ldots, g_n$ are chosen uniformly at random over the field $F_{2^s}$. The summation is to be done for every symbol in a vector of L/s symbols in a packet. i.e.,

$X_p = \sum_{i=0}^{n} g_i M_p^i$ where $M_p^i$ and $X_p$ is the $P^{th}$ symbol of $M_i$ and $X$ respectively. Coefficients vector $g = (g_1, .., g_n)$ is also sent in the encoded packet along with encoded data $X$. Forwarding nodes can also do encoding on already encoded packets. Consider a node that has a set $g = (g^1, X^1), \ldots, (g^m, X^m)$ of encoded packets, where $g^j$ is the encoding vector of the $j^{th}$ packet and $X^j$ is the information vector of $j^{th}$ packet. This node can generate a new encoded packet $(k, Y)$ by selecting uniformly at random a set of coefficients $h_1, \ldots, h_n$ and computing the linear combination $Y = \sum_{j=0}^{m} h_i X^i$. As the coefficients should be with respect to the original packets, the corresponding $k$ vector is given by $k_i = \sum_{j=0}^{m} h_i g_i^j$. Encoding example on field $F_{2^8}$ with $m = 2$ is given in appendix.

### C. Decoding

Consider that a node has received the set $(k^1, Y^1) \ldots (k^m, Y^m)$. To retrieve original packets, it needs to solve the system of m linear equations $Y^j = \sum_{i=1}^{n} k_i^j M^i$ with $M^i$'s as unknowns. If $m = n$, i.e., number of received packets are equal to number of original packets, we can recover all the original packets provided all the encoded packets are linearly independent. But there is a possibility that some of the encoded packets are linearly dependent. Simulation results indicate that even for small field sizes (e.g. $s = 8$), the probability becomes negligible [10]. For the example given in appendix, please note that $m = n = 2$.

### D. Generation Size

As the number of originating packets (n) in a network for given destination increases, the amount of memory needed to store coefficients of encoded packets increases because these coefficients are to be remembered till a node receives at least n packets to decode all original packets. Further, till at least n encoded packets are not received, most of the n original packets can not be decoded. Thus delivery delay of a packet increases with n. To reduce memory requirement and the delay, originating packets can be grouped together into so called 'generation'. Now all the nodes in the network will encode packets of the same generation. If the generation size is k ( $k \ll n$), whenever the receiver receives at least k packets of a generation, it will able to decode k original packets and the corresponding k encoded packets can be discarded, freeing the memory and the delay will also be reduced as receiver will have to now wait only for k encoded packets to be able to decode. But as packets from different generations can not be 'mixed', mixing opportunity reduces with k.

In sensor network literature very limited material is there which has used the powerful tool of network coding. Wang et al. in [12] presented partial network coding (PNC) as a generic tool for continuous data collection. PNC generalizes the existing network coding (NC) paradigm but enables efficient storage replacement for continuous data, which is a major deficiency of the conventional NC. They proved that the performance of PNC is quite close to NC, except for a sub linear overhead on storage and communication.

ShanShan et al. in [13] proposed an energy aware method to combine multipath routing with practical network coding. Through this method, same reliability as conventional multipath mechanisms can be guaranteed with significantly reduced energy consumption by decreasing the number of paths needed to deliver data. The method only needs little metadata overhead and some small scale linear operations. Toledo and Wang in [14] used network coding to achieve an adaptive equivalent solution to the construction of disjoint multipath routes from a source to a destination. It exploits both the low cost mesh topology construction, such as those obtained by diffusion algorithms, and the capacity achieving capability of linear network coding. The solution easily adapts to the changing conditions of the wireless sensor network and it can be used by the sinks to increase or decrease capacity and reliability on demand.

In our work, we have used network coding for better buffer utilization and loss recovery in wireless sensor networks. In normal network, each packet has its own information. On packet loss, lost packet can not be recovered from other packets. Sink can get that packet either from source or from intermediate nodes. With random loss, nodes need to store more packets to reduce message transmission for loss recovery. It creates trade-off between buffer size and recovery cost and both are scarce resources for WSN. Network coding helps here in reducing the buffer requirement. Instead of storing entire generation or bulk of packets at source or intermediate nodes, one can distribute load among nodes by distributing encoded packets of the generation uniformly among the nodes on the path from source to sink (1) because an encoded packet carries information of multiple original packets and is useful to recover any on these lost packets. This helps in reducing the buffer requirement at each node and reduces the buffer size. In normal network, it is hard to decide how many number of packets should be stored and at which nodes because each packet carries only its own information.

We have used this fact and designed the buffer allocation policy. In our scheme, source node applies random linear coding on generated packets. That spread the information of current generation in all the packets belonging to that generation. Intermediate nodes store the encoded packet and for new packet, update their coefficients to add information in stored data. Sink node recovers the entire generation after reversing the process. On packet loss, information that is uniformly distributed on the path helps the sink to recover packet loss with less number of messages.

## III. BUFFER MANAGEMENT WITH NC

Network nodes have limited buffer size and it is desired to use it efficiently. Efficiency of buffer management scheme can be measured by buffer availability, information loss due to limited buffer size and redundancy in stored data. Buffer management scheme should also recover lost packets with minimum retransmitted messages (recovery cost).

In our network model, source node periodically generates the packets, applies network coding and sends them towards the sink node through intermediate nodes. We have assumed that it is the responsibility of routing protocol to forward the

packet towards the sink. We also depend on network layer for unique path between sink node and source node.

*A. Conventional Coding Scheme*

By Conventional Coding Scheme (CCS), we mean buffer management scheme for network without network coding. In CCS, a node stores the arrived packet into its buffer with the purge time. On purge time, it removes the packet from the buffer. In the case when there is no room for new packet, it removers randomly a packet from buffer to make space for arrived packet.

*B. Network Coding Scheme*

In network coding based buffer management scheme (NCS), after generating packet, as explained above, node applies random linear network coding on it along with previously stored packets and then sends the encoded packet to the network layer. Normally network coding is applied on group of packets. Required number of packets depends on the generation size. In the case when packets are less then the generation size, it can be applied with corresponding coefficients as zero. Procedure to encode packet is explained in section 2 under encoding subsection. These generated packets are forwarded to the network layer. Network layer sends them to parent node. Each sent packet contains the generation number and source id with it. With generation number and source id, network nodes uniquely identify the packet generation. Here instead of recognizing each packet, network nodes only need to recognize the unique generation. All the packets belonging to same generation contain some information about entire generation. This helps in reducing the buffer size with group of encoded packets then the bulk of individual packets.

Algorithm I covers the steps for the source node. In conventional network, each packet has its own information so if a packet is lost, it can not be recovered from other packets. Sink can get it either from source or from intermediate nodes. To recover this random packet loss with reduced message transmissions, nodes have to store more packets and require larger buffer size. Network coding helps in reducing the buffer requirement.

Instead of storing entire generation at source or intermediate nodes, we only store limited information. Some part of information is stored on source and other on the path from source to sink uniformly (1) for loss recovery as shown in Line 10-14 where Slevel is number of hops from source to sink and GSize is the generation size .

$$\text{limit} = \begin{cases} \left\lfloor \frac{\text{GSize}}{\text{Slevel}} \right\rfloor + \left\lfloor \left( \frac{\text{GSize}}{\text{Slevel}} - \left\lfloor \frac{\text{GSize}}{\text{Slevel}} \right\rfloor \right) \times 2 \right\rfloor, \\ \text{if Slevel} < Gsize \text{ and even level} \\ \left\lceil \frac{\text{GSize}}{\text{Slevel}} \right\rceil \qquad \text{otherwise} \end{cases} \quad (1)$$

On the way from source to destination, packet passes through intermediate nodes. Nodes identify the packet based on its source id and generation number. On new generation, a node stores the first packet in its buffer and forwards this packet to parent without processing it. If node has packet in its buffer from same generation and source, node applies network coding on stored packets of given generation and source and incoming packet to generate encoded packet. Node updates its buffer with this newly encoded packet. It also increments received packet count for given generation and source. Node then forwards the received packet to its parent. Buffer space is allocated based on sources level number and generation size. Equation 1 states the buffer size for given source level Slevel and generation size GSize. With this we have uniformly distributed the buffer load on network path. This helps in reducing number of messages required for loss recovery with reduced buffer size. Periodically node purges its buffer, removing old packets from it. Purge delay is calculated based on node's hop distance (level) from the sink and expected load in the network. Purge delay is multiple of maximum RTT and generation size. Algorithm II covers the steps for intermediate nodes.

Lines 26 to 37 cover the steps that intermediate node takes on query received from parent. Based on loss count and buffered packets, node sends the reply. If node does not have sufficient packets to recover loss, it forwards the query to child. Until the query is fully resolved, network keeps on sending query towards the source node.

At last packet reaches sink node. On sink node, when packet arrives, sink identifies the source id and generation number. Nodes wait for sufficient number of packets that is equal to generation size. On receiving sufficient packets sink node applies the decoding on received generation. The decoding method is given in section 2 under decoding sub section. Generation is successfully decoded if it ends with identity matrix with rank equal to generation size. If rank is lower or sink receives next generation packet before all the

**Algorithm I:** Source Node

| | |
|---|---|
| 1 | **Line 2-6 packet generation & buffering** |
| 2 | $l \leftarrow 1$ |
| 3 | *Generate packet* $P_l$ |
| 4 | *If* $l <$ *GSize* |
| 5 | $buffer\ P_l$, |
| 6 | $increment\ l$, |
| 7 | **Line 8-10 first level encoding before sending** |
| 8 | $for\ i = 1\ to\ GSize\ do$, |
| 9 | $generate\ encoded\ packet\ EP_i$ |
| 10 | $EP_i = ERLC\ (\ P_1, \dots, P_l.., P_{Gsize})$ |
| 11 | **Line 12-16 second level encoding for buffering** |
| 12 | $for\ i = 1\ to\ limit\ do$, |
| 13 | $generate\ encode\ packet\ EEP_i$ |
| 14 | $EEP_i = ERLC\ (EP_i, \dots.., EP_{gsize})$ |
| 15 | *Store* $EEP_i$ *in buffer with purge time* |
| 16 | *remove unwanted packets* |
| 17 | **Forwarding packet to down layer** |
| 18 | *Forward encoded packet with random delay* |
| 19 | **Purging the packet to make buffer free** |
| 20 | *On purge event remove* $EEP_i$ *from buffer* |

Algorithm II: Intermediate Nodes

1 **Line 2-6 & 14-16 buffers received packet**
2 *On packet receive P*
3 *Get sId, Slevel and Gnumber from P*
4 *If first pkt from Slevel and Gnumber*
5 $buffer\ P$ as $P_{sId,Gnumber,k}$
6 $update\ C_{sId}^{Gnumber}, k$ //**Counters**
7 *go to Line 22*
8 **Line 10-13 & 18-21 Encode received packet with**
9 **buffered packets**
10 *If Slevel > GSize*
11 *Encode P with buffered* $P_{sId,Gnumber,k}$
12 $P_{sId,Gnumber,k} = ERLC\ (P, P_{sId,Gnumber,k})$
13 *Else*
14 *If* $C_{sId}^{Gnumber} < limit_{sId}^{Gnumber}$
15 $buffer\ P$
16 *update* $C_{sId}^{Gnumber}, k$
17 *Else*
18 *Encode P* with buffered $P_{sId,Gnumber,k}$
19 $for\ i = 1\ to\ limit_{sId}^{Gnumber}\ do,$
20 $P_{sId,Gnumber,i} = ERLC\ (P, P_{sId,Gnumber,i})$
21 store $P_{sId,Gnumber,i}$ with purge time
22 **Forward Packet to parent**
23 *Forward P*
24 **Removing packet from buffer**
25 *On purge event remove* $\boldsymbol{P_{sId,Gnumber,k}}$ *from buffer*
26 **Send packets to sink to recover from packet loss**
27 *On Query* $Q(sId, Gnumber, loss)$
28 $t \leftarrow k$
29 *while* $Gnumber - loss < C_{sId}^{Gnumber}, t > 0$ *and loss >0*
30 *Send encoded packet*
31 *Decrement t, loss*
32 *If loss > 0*
33 **If insufficient packets to recover then forward query**
34 **to child**
35 *Forward query* $Q(sId, Gnumber, loss)$

packets of current generation, it generates the query message asking for packets of incomplete generation which contains loss details and the number of independent packets required to successfully decode the generation. The required number of independent packets is the difference between generation size and rank of the identity matrix. Intermediate nodes send the required packet if they have or they forward the query message to the child node. As all the packets have information about generation, it helps to reduce packet recovery cost. Here loss is the difference between generation size and rank of received packets.

Algorithm III: Sink Node

1 **Line 2 -6 receive packet**
2 *On packet receive P*
3 *Get* $sId, slevel$ *and Gnumber from* $P$
4 *if* $C_{sId}^{Gnumber} < Gsize$ *or not new generation*
5 $buffer\ P$ as $P_{sId,Gnumber,k}$
6 *update* $C_{sId}^{Gnumber}, k$
7 **Line 9 to 16 packet decoding and sending query**
8 **to child on information loss.**
9 *else*
10 $Rank = RRLC\ (P_{sId,Gnumber,1}, \ldots\ldots, P_{sId,Gnumber,k})$
11 *If rank < Gsize or On time out*
12 *update* $C_{sId}^{Gnumber}, k$
13 *Send Query*$(sId, Gnumber, Gsize - rank)$
14 *Else* $for\ i = 1$ *to GSize do*
15 $P_i = DRLC\ (P_{sId,Gnumber,1}, \ldots\ldots, P_{sId,Gnumber,k})$
16 *remove generation* $sId, Gnumber\ from\ buffer$

## IV. SIMULATION RESULTS

We have done our simulation on Network Simulator-2 (NS-2). In our simulation 100 nodes are deployed on a 10x10 grid. Nodes allocate fixed buffer B to each passing generation. Boundary nodes are source nodes and the sink node is located at the centre of the grid. We have compared our network coding scheme (NCS) with conventional coding scheme (CCS). Since generation size concept is only applicable to NCS and not to CCS, to make a fair comparison, we have kept the bulk size (number of packet generated at a time) equal to the generation size. For the given generation or bulk, total available buffer $T_{Buffer} = Slevel * B$ is same in both the schemes. Performance parameters of interest are buffer availability, information loss, and cost for packet recovery. Due to limited buffer or network conditions, packets are lost in the network. We have analyzed the cost to recover packet loss and effect of limited buffer on information loss and redundant information in the network.

Figure 1 shows the availability of buffer. Here we have plotted available buffer size as function of generation size. This analysis is of limited buffer case, where each node has fixed buffer for a generation of a source node. The figure shows that

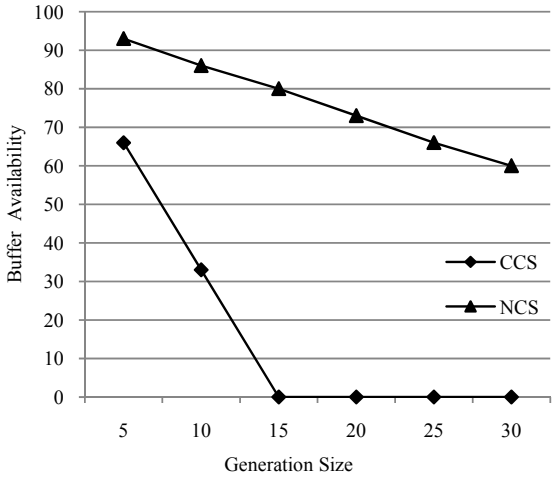


Figure 1: Average un-utilized buffer in network coding scheme and conventional coding scheme

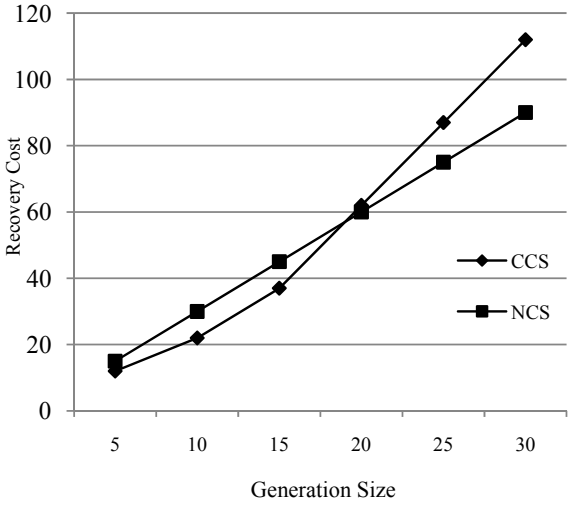


Figure 2: Recovery cost (in # packets) for network coding scheme and conventional coding scheme.

as the generation size (bulk size) increases, free buffer size decreases in both schemes but in NCS, the rate of decrease in much less then in CCS. It means that NCS requires less buffer size for given generation size (bulk size) compared to CCS. Once the bulk size is greater than the buffer size, packets are lost with significant rate in CCS.

From the result, we conclude that CCS has more redundant packets in the network compared to NCS and so more packets get dropped once buffer is full resulting in more information loss. As a result, more messages are required for packets recovery. In NCS, equation 1 and network coding spread the source information on nodes along the path and amount of redundant information is between 0 to 15% which is much lower than CCS.

Figure 2 shows the recovery cost of packet loss as function of generation size. CCS performs better then NCS when the generation size (bulk size) is smaller then the buffer size. It performs better because sink gets required packets nearer to it. This analysis is average case analysis with random packet loss at each level. At higher generation size (bulk size), nodes randomly remove packets from buffer to make room for new packet. But as the generation size (bulk size) increases, CCS has more information loss. To recover packet, it has to travel more hops towards the source. That increases the recovery cost. In case of NCS, packets belonging to same generation have information about all the packets of generation. That reduces the search compared to individual packet search. For recovery, it requires any independent packet of same generation than the specific individual packet. This helps in reducing the recovery cost.

## V. CONCLUSION

In this paper we have proposed buffer management scheme for uniform buffer utilization with minimum redundancy in stored data and reduced loss recovery cost with limited buffer size. Our scheme is designed for many to one data traffic pattern but it can also work when network layer provides bidirectional path between source and destination. We have used in-network processing of packets which is random linear coding. Source node and intermediate nodes apply network coding on generated or forwarded packets and store only required number of packets which depends on loss rate and uniform buffer allocation. Our results show that compared to CCS, NCS requires smaller buffer size and has uniform buffer allocation with minimum redundancy. NCS also has less recovery cost compared to CCS.

## REFERENCES

[1] Kuorilehto, M., Hännikäinen, M., and Hämäläinen, T. D. 2005. "A survey of application distribution in *wireless sensor networks". EURASIP J. Wirel. Commun. Netw. 5, 5 (Oct. 2005), 774-788.*

[2] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, "Energy aware wireless microsensor networks," *IEEE Signal Processing Magazine*, vol. 19, iss. 2, pp. 40--50, March 2002.

[3] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. "Habitat monitoring: Application driver for wireless communications technology." In *Proceedings of the 2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean,* April 2001., 2001.

[4] Edoardo Biagioni and Kent Bridges. "The application of remote sensor technology to assist the recovery of rare and endangered species". *In Special issue on Distributed Sensor Networks for the International Journal of High Performance Computing Applications,* Vol. 16, N. 3, August 2002.

[5] K. Chintalapudi, T. Fu, R. Govindan, E. Johnson, "Structural Damage Detection and Localization using Wireless Sensor Networks with Low Power Consumption", *Proc. of the 5h International Conference on Structural Health Monitoring,* 2005.

[6] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *in IEEE Transactions on Information Theory,* vol. 46, pp. 1204-1216, July 2000.

[7] T. Ho, R. Koetter, M. M`edard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," *in IEEE International Symposium on Information Theory, 2003.*

[8] Rudolf Ahlswede, Ning Cai, ShuoYen Robert Li, and Raymond W. Yeung. "Network information flow". *In IEEE Transactions on Information Theory,* vol. 46, pp. 12041216, July 2000.

[9] ShuoYen Robert Li, Raymond W. Yeung, and Ning Cai. "Linear network coding". *In IEEE Transactions on Information Theory,* vol. 49, pp. 371381, February 2003.

[10] Tracey Ho, Ralf Koetter, Muriel M`edard, David R. Karger, and Michelle Effros. "The benefits of coding over routing in a randomized setting", *In IEEE International Symposium on Information Theory,* 2003.

[11] Christina Fragouli, JeanYves Le Boudec, and Jorg "Widmer. Network coding: An instant primer." *In ACM SIGCOMM Computer Communication Review*, January 2006.

[12] D. Wang, Q. Zhang, and J. Liu, "Partial network coding: Theory and application for continuous sensor data collection," *in 14th IEEE International Workshop on Quality of Service*, June 2006.

[13] L. ShanShan, Z. PeiDong, and L. XiangKe, "Energy efficient multipath routing using network coding in wireless sensor networks," in *ADHOC-NOW*, 2006.

[14] A. L. Toledo and X. Wang, "Efficient multipath in sensor networks using diffusion and network coding," *in 40th Annual Conference on Information Sciences and Systems*, March 2006.

## APPENDIX

Following example explains the working of random linear network coding. In the example we have take $n = 2$, $p = 1$, $m = 2$ and field size $s = 8$, $F_{2^8}$.

*A. Encoding*

M is our original packet generated by one of the source and G is our coefficient vector from field $F_{2^8}$.

$$M = \begin{pmatrix} M^1 \\ M^2 \end{pmatrix} = \begin{pmatrix} 161 \\ 87 \end{pmatrix}$$

$$G = \begin{pmatrix} g_1^1 & g_2^1 \\ g_1^2 & g_1^2 \end{pmatrix} = \begin{pmatrix} 230 & 33 \\ 155 & 85 \end{pmatrix}$$

$$gX = \begin{pmatrix} 230 & 33 \\ 155 & 85 \end{pmatrix} \begin{pmatrix} 161 \\ 87 \end{pmatrix} = \begin{pmatrix} 108 \\ 104 \end{pmatrix} = \begin{pmatrix} X^1 \\ X^2 \end{pmatrix} = X$$

Following steps are for applying encoding on already encoded packets. Node receives $g = (g^1, X^1)$ applies encoding on it and generates new encoded packet $(k, Y)$. Here new coefficient vector k is calculated by $k_i = \sum_{j=0}^{m} h_i g_i^j$ and $Y = \sum_{j=0}^{m} h_i X^i$.

$$g = \begin{pmatrix} 230 & 33 \\ 155 & 85 \end{pmatrix}, X = \begin{pmatrix} 108 \\ 104 \end{pmatrix}$$

$$h = \begin{pmatrix} h_1^1 & h_2^1 \\ h_1^2 & h_1^2 \end{pmatrix} = \begin{pmatrix} 39 & 202 \\ 236 & 141 \end{pmatrix}$$

Getting re-encoded packet using new coefficient $h$ from $F_{2^8}$.

$$hX = \begin{pmatrix} 39 & 202 \\ 236 & 141 \end{pmatrix} \begin{pmatrix} 108 \\ 104 \end{pmatrix} = \begin{pmatrix} 142 \\ 97 \end{pmatrix} = \begin{pmatrix} Y^1 \\ Y^2 \end{pmatrix} = Y$$

Calculation new coefficient vector $k$ from $h$ and $g$

$$hg = \begin{pmatrix} 39 & 202 \\ 236 & 141 \end{pmatrix} \begin{pmatrix} 230 & 33 \\ 155 & 85 \end{pmatrix} = \begin{pmatrix} 133 & 33 \\ 155 & 85 \end{pmatrix}$$

$$hg = \begin{pmatrix} k_1^1 & k_2^1 \\ k_1^2 & k_1^2 \end{pmatrix} = k$$

*B. Decoding*

At decoding node, node receives the encoded packet $(k, Y)$. To get original packets node applies gauss-elimination for the system of $m$ linear equation.

$$k = \begin{pmatrix} 133 & 186 \\ 127 & 123 \end{pmatrix}, Y = \begin{pmatrix} 142 \\ 97 \end{pmatrix}$$

$$\begin{pmatrix} 133 & 186 \\ 127 & 123 \end{pmatrix} \begin{pmatrix} M^1 \\ M^2 \end{pmatrix} = \begin{pmatrix} 142 \\ 97 \end{pmatrix}$$

After applying gauss-elimination we have following results. In which left-hand side matrix is identity matrix with rank equals to $n$. To decode it successfully one need $m$ independent equations, in other words $m$ independent packets. If its rank is less than $n$ then our $m$ linear equations are not independent and generation can not get decoded.

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} M^1 \\ M^2 \end{pmatrix} = \begin{pmatrix} 161 \\ 87 \end{pmatrix}$$