Practical Evaluation of Partial Network Coding in Wireless Sensor Networks

Rasmus M. Jacobsen
Department of Electronic Systems
Aalborg University, Denmark
raller@es.aau.dk

Kasper L. Jakobsen
Department of Electronic Systems
Aalborg University, Denmark
kalund@es.aau.dk

Peter J. Ingtoft
Department of Electronic Systems
Aalborg University, Denmark
ingtoft@es.aau.dk

Tatiana K. Madsen
Department of Electronic Systems
Aalborg University, Denmark
tatiana@es.aau.dk

Frank H.P. Fitzek
Department of Electronic Systems
Aalborg University, Denmark
ff@es.aau.dk

ABSTRACT

Network Coding (NC) is a hot topic nowadays. Even though it has proven its applicability to wireless networks, there are still some questions whether network coding is feasible for small devices. To answer this question an implementation of the Partial Network Coding (PNC) buffer scheme is done on a wireless sensor and the results are described within this paper. Originally PNC was presented in [4] and this paper validates the findings of the original work by implementing it on the OpenSensor Board (OSB) [3]. The OSB is developed at Aalborg University in collaboration with Technical University of Berlin. On this simple sensor board PNC is implemented and tested.

Categories and Subject Descriptors

C.2.1 [**Computer Systems Organization**]: Computer-Communication Networks—*Network Architecture and Design*; C.4 [**Computer Systems Organization**]: Performance of Systems

General Terms

Measurement, Performance, Reliability

Keywords

Partial Netork Coding, Network Coding, Wireless Sensor Networks

1. INTRODUCTION

A Wireless Sensor Network (WSN) consists of spatially distributed autonomous devices using sensors to cooperatively monitor physical or environmental conditions at different locations. A WSN can be used with advantage to collect data in large areas where it is impossible to use traditional wired or wireless networks e.g. consider a sensor network deployment in a remote and inaccessible environment where sensors are measuring and storing data in the network over long time periods. A Data Collector (DC) may appear at any location and time in the network and try to retrieve as much data as possible [1].

To make WSN deployments economically and technologically feasible, it is necessary to minimize the cost, size, memory and battery energy consumption of the sensors. Due to these constraints, one sensor can store only a small amount of data collected from its surroundings, making the data extraction process difficult for DC, as DC has to contact more sensors to get the required amount of data.

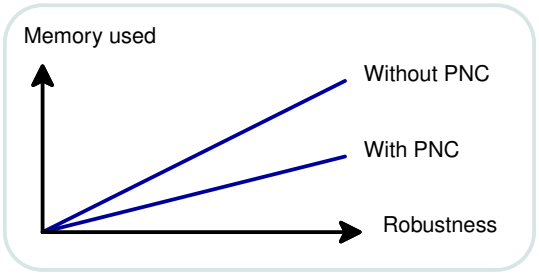
In current WSNs, the process of extracting data is as follows: DC contacts a sensor in the border of the WSN, which in turn routes the query into the network. The sensors in the network then reply the query with sensed data, which is routed out from the network through the sensor contacted by DC. This method has a lot of disadvantages, namely a big routing overhead will occur on the sensors near the contacted sensor, which will increase the number of transmissions. Furthermore if some sensors in the network have stopped functioning since last data retrieval, it is no longer possible to guarantee that the query is received by all sensors, which will increase the probability of lost data.

To overcome the described drawbacks the goal should be to make as few transmissions as possible decreasing the battery energy consumption for each sensor. This requirement brings NC into the picture, where sensed data, when queried by DC, is coded. Each sensor listens to the send packets, and encodes this content into its own packets. The sensors can then use these coded packets, to decode the information relevant for them. This will in general decrease the number of internal transmissions, because the necessity for retransmissions decreases.

PNC is a special case of NC, where the sensed data is distributed within the WSN before data collection. This increases the number of internal transmissions in the network, but reduces the number of transmissions when DC extracts

Mobimedia 2008 July 7-9, 2008, Oulu, Finland.
 ISBN 978-963-9799-25-7/08/07
DOI 10.4108/ICST.MOBIMEDIA2008. 4025

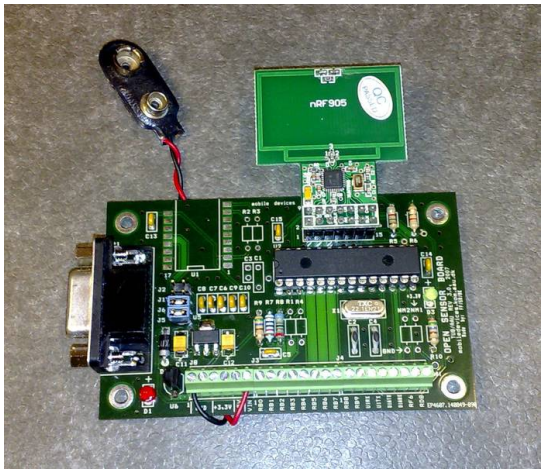
data. Furthermore it also creates redundancy, because of the distribution of data. The saliently feature in PNC is, that the sensors in the network does not have to decode packets while they are distributed in the network, e.g. to remove deprecated data, which unnecessarily would increase energy consumption.



1: The gain when using PNC, where robustness is e.g. the amount of redundancy in the WSN.

In Figure 1, the gain when using PNC is illustrated, by means of memory required for storing data, but other factors exists, e.g. number of transmissions required for data retrieval from DC, energy consumed in the WSN when contacted by DC, etc.

In the following the paper introduces the details in PNC and it explains the main motivation to use it in a WSN. The paper gives a short introduction to NC and PNC and how this have been implemented on the OSB, see Figure 2, to test the performance of PNC. The tests performed with the developed system are described with the corresponding results, followed by a conclusion on whether PNC can be used in a real environment.



2: The OpenSensor Board.

2. PARTIAL NETWORK CODING

A key deficiency of the conventional NC is the lack of support for removing obsolete data. Conventional NC has first to decode and then re-encode buffer elements which is time and energy consuming [2]. As a result, PNC is introduced. PNC introduces a method for easy removal of old packets.

Current research in PNC described in [4] and [5] focuses *only* on the data collection part, and not the distribution of packets among sensors before data collection. It therefore *assumes that every packet generated by a sensor in a WSN has arrived (without NC) to each sensor before data collection*. PNC for data collection then benefits from the fact that it is assumed that the newest packets are always the ones which are of interest by DC. This is not ideal, and other schemes are needed to distribute packets between sensors before data extraction.

A small example should help to understand PNC. In the following it is assumed that the buffers in the two sensors are given as follows:

$$s_1 : \left\{ \begin{array}{l} f_1 = \beta_{1,3} \times c_3 + \beta_{1,2} \times c_2 + \beta_{1,1} \times c_1, \\ f_2 = \beta_{2,3} \times c_3 + \beta_{2,2} \times c_2 \end{array} \right\}$$
$$s_2 : \left\{ \begin{array}{l} f_1 = \beta_{1,3} \times c_3 + \beta_{1,2} \times c_2 + \beta_{1,1} \times c_1, \\ f_2 = \beta_{2,3} \times c_3 \end{array} \right\}.$$

Here c_js are packets, and $\beta_{i,j}$s are random coefficients. They together form a buffer element denoted f_i. Note, the amount of memory used to represent a buffer element is independent on the number of packets encoded, because all operations resides in a Galois field. The packets with the same indices are the same on both sensors, where the random coefficients and buffer elements are different, e.g. $\beta_{1,1}$ on s_1 is not equal $\beta_{1,1}$ on s_2.

To continue the example, if the maximum number of packets in a buffer element N, i.e. the *full cardinality*, is $N = 3$, then, when a new packet c_4 arrives at the two sensors, the buffers become

$$\begin{array}{lll} \text{before} & s_1 & : \{f_1 = [c_3, c_2, c_1], \quad f_2 = [c_3, c_2]\} \\ & s_2 & : \{f_1 = [c_3, c_2, c_1], \quad f_2 = [c_3]\} \\ \text{after} & s_1 & : \{f_1 = [c_4], \quad f_2 = [c_4, c_3, c_2]\} \\ & s_2 & : \{f_1 = [c_4], \quad f_2 = [c_4, c_3]\}. \end{array}$$

This demonstrates the salient feature of PNC, that is, removing obsolete data without decoding.

To be able to decode N packets, the rank of the decoding matrix (further explained in Section 3) should be at least N. Therefore, whenever DC is querying at least N sensors it should be guaranteed to receive N buffer elements which all contains N encoded packets. Then, if (with high probability [4]) the coefficient vectors are linearly independent, DC is able to decode. In the example this clearly does not hold since none of the buffer elements in s_2 has the cardinality N after the arrival of c_4.

To achieve that a given sensor *always have a buffer element with cardinality at least* N, a distributed scheme is required on each sensor to manage the buffer elements; this maintaining problem is the challenge in PNC. In the following section this maintenance problem is translated into an initial distribution problem which is simpler and easier to achieve.

Buffer Scheme

To achieve the ability to ensure that each sensor always are able to transmit a buffer element containing the linear combination of at least the N newest packets, it is proved in [4], that by using a full cardinality of $N + \sqrt{N}$ and a buffer of size $B = \sqrt{N} + 1$, being the number of buffer elements in each sensor; then a sensor can *always* provide a buffer element with cardinality at least N when queried. These values are described in the following.

In conventional NC, a full cardinality equal N means that N packets can be decoded after collecting N buffer elements. In PNC, this value is increased by a sub-linear overhead $\sqrt{N}$; resulting in, that at least $N + \sqrt{N}$ buffer elements have to be received to guarantee that N packets can be recovered.

As mentioned, by initializing the buffer elements in a clever way, it can be ensured that (at least) one buffer element in a sensor have a cardinality of at least N. This initialization is performed in each sensor by selecting cardinalities, in a way such that each cardinality is picked exactly once, from the set

$$\hat{k} \in [\underbrace{\sqrt{N}, \ldots, N - 2\sqrt{N}, N - \sqrt{N}, N, N + \sqrt{N}}_{\sqrt{N}+1}].$$

Now, a buffer element given initial cardinality $\hat{k}$ should not encode the first $N+\sqrt{N}-\hat{k}$ packets, but encode the following $\hat{k}$ packets. This ensures that, after the arrival of $N + \sqrt{N}$ packets, the distribution of cardinalities in the buffers still satisfy the uniform property.

Consider the following example where $N = 4$. This gives the full cardinality 6, a buffer of size $B = 3$, and that each sensor will pick its initial cardinalities from the set $\hat{k} \in [2, 4, 6]$. If buffer element f_1 picks $\hat{k} = 2$, f_2 picks $\hat{k} = 4$, etc., then after the arrival of the first six packets, the buffer in a sensor will look like

$$s_1 \quad : \left\{ \begin{array}{lll} f_1 & = [c_6, c_5], & 4 + \sqrt{4} - 2 = 4 \text{ p.s.} \\ \mathbf{f_2} & = [c_6, c_5, c_4, c_3], & 4 + \sqrt{4} - 4 = 2 \text{ p.s.} \\ \mathbf{f_3} & = [c_6, c_5, c_4, c_3, c_2, c_1], & 4 + \sqrt{4} - 6 = 0 \text{ p.s.} \end{array} \right\},$$

where p.s. is the number of packets skipped until packets are encoded into the buffer element. The bold buffer elements are the ones which satisfy the property that at least N packets are coded in the buffer element. Now, for every new packet arriving to the sensor, the property of having at least one buffer element with cardinality greater then N still holds. This is illustrated by continuing the example. Assume that three more packets arrives; then the buffer elements become as follows:

$$c_7 \text{ arrives} \longrightarrow \quad s_1 \quad : \left\{ \begin{array}{ll} f_1 & = [c_7, c_6, c_5], \\ \mathbf{f_2} & = [c_7, c_6, c_5, c_4, c_3], \\ f_3 & = [c_7] \end{array} \right\}$$

$$c_8 \text{ arrives} \longrightarrow \quad s_1 \quad : \left\{ \begin{array}{ll} \mathbf{f_1} & = [c_8, c_7, c_6, c_5], \\ \mathbf{f_2} & = [c_8, c_7, c_6, c_5, c_4, c_3], \\ f_3 & = [c_8, c_7] \end{array} \right\}$$

$$c_9 \text{ arrives} \longrightarrow \quad s_1 \quad : \left\{ \begin{array}{ll} \mathbf{f_1} & = [c_9, c_8, c_7, c_6, c_5], \\ f_2 & = [c_9], \\ f_3 & = [c_9, c_8, c_7] \end{array} \right\}$$

This behavior can be generalized by the following dataReplacement() algorithm which can be used to maintain the buffer elements after the initial configuration. The algorithm gets the packet to be encoded, and either encodes with the already encoded packets, or copies the coded element replacing the existing buffer element.

```
1  dataReplacement(c_new)
2  {
3    for (i = 1; i <= B; i++)
4    {
5      β = GetRandomCoefficient();
6      if (GetBufferElementCardinality(f_i) < N + √N)
7        f_i = β × c_new + f_i;
8      else
9        f_i = β × c_new;
10   }
11 }
```

3. MAP BUFFER ELEMENTS TO DECODING MATRIX

If it is assumed that all sensors coefficient vectors are *linearly independent*, then the $N+\sqrt{N}$ packets can be restored by solving a set of linear equations, after collecting $N+\sqrt{N}$ buffer elements, given that all buffer elements contains a linear combination of all $N + \sqrt{N}$ packets. This is not always the case as it must be assumed that not all sensors have all packets. If the number of buffers to retrieve is denoted W, clearly $W \geq N + \sqrt{N}$.

A *decoding matrix* $\overline{\overline{G}}$ is given as

$$\overline{\overline{G}} = \begin{bmatrix} \beta_{1,1} & \cdots & \beta_{1,N+\sqrt{N}} \\ \vdots & \ddots & \vdots \\ \beta_{W,1} & \cdots & \beta_{W,N+\sqrt{N}} \end{bmatrix}$$

If a row is added to the matrix, and the rank is increased, the buffer element retrieved is said to be innovative.

When sufficient buffer elements are retrieved, i.e. when $\text{rank}(\overline{\overline{G}}) = N+\sqrt{N}$, the extraction of packets can be solved, because the number of unknowns are equal to the number of equations. The packets can be extracted by solving the equation

$$\overline{F} = \overline{\overline{G}}\overline{m},$$

where $\overline{F}$ is the *information vector* containing the buffer elements, i.e. $\overline{F} = [f_1, \ldots, f_W]$, and $\overline{m}$ is the vector containing the original packets.

The following mapping of buffer elements into the decoding matrix, tweaks this approach, as it is only guaranteed that the buffer elements received have the cardinality N (and not $N + \sqrt{N}$ as above). A mechanism is therefore needed to map the buffer elements with varying cardinality into the decoding matrix and still ensure decoding.

It is possible to receive more buffer elements than the actual size of the matrix. Therefore a mechanism is needed to determine which of the buffer elements to choose, to make it possible to decode the N elements in the buffer elements with highest index.

Assume that the buffer elements contains the indices marked in Table 1, and that $N = 4$. Here the buffer elements to put into the decoding matrix would be 1, 2, 4, 6, and 7. The reason for this is the following; there cannot be found four ($N = 4$) buffer elements which only contains the newest four indices (only 2, 4, and 6 holds this property). Therefore the number of indices included in the decoding matrix is increased, and the new dimension of the decoding matrix is now 5. A linear system can now be created with five equations and five unknowns, which is given in the matrix next to the table.

The reason why the matrix is wanted as small as possible (not lesser than N), is to increase the probability of decoding. Assume that all buffer elements received only contains the newest N indices, then decoding is not possible if the matrix have dimensions $N + \sqrt{N}$, because the rank cannot be greater then N. Opposite; if all buffer elements contains the $N+\sqrt{N}$ newest packets, then the decoding matrix should keep this dimension.

Another issue is the complete lack of an index in the buffer elements. An example is in Table 2 ($N = 4$), here the packet indices 4 and 5 are not included in any of the buffer elements. A packet loss is unavoidable, but some of the indices can be decoded with success. If the buffer elements 1, 3, 5, and 6 are

Indices:	1	2	3	4	5	6	7
Buffer elm. 1:			x	x	x		x
Buffer elm. 2:				x	x	x	x
Buffer elm. 3:		x	x	x	x		
Buffer elm. 4:				x		x	x
Buffer elm. 5:	x	x			x	x	
Buffer elm. 6:						x	x
Buffer elm. 7:			x		x	x	

(a)

$$\overline{\overline{G}} = \begin{bmatrix} x & x & x & & x \\ & x & x & x & x \\ & x & & x & x \\ & & & x & x \\ x & & x & x & \end{bmatrix}$$

(b)

1: Example of mapping indices from buffer elements into the decoding matrix. (a) The table illustrates which indices are in which buffer element. (b) The buffer elements containing the newest indices which can construct a quadratic decoding matrix are chosen.

Indices:	1	2	3	4	5	6	7
Buffer elm. 1:		x				x	x
Buffer elm. 2:	x		x				x
Buffer elm. 3:		x					x
Buffer elm. 4:	x	x				x	x
Buffer elm. 5:		x	x				
Buffer elm. 6:		x				x	

(a)

$$\overline{\overline{G}} = \begin{bmatrix} x & & x & x \\ x & & & x \\ x & x & & \\ x & & x & \end{bmatrix}$$

(b)

2: Example of packet loss in the received buffer elements.

chosen, then the indices 2, 3, 6, and 7 can be decoded, which is considered better then nothing. The resulting matrix is shown next to the table.

The following algorithm takes both issues into account, and fulfills the task;

1. RemoveDuplicates()
   If the same buffer element is received more then once, remove the duplicated buffer elements.
2. DetermineNumIndicesInEachBufferElement()
   Determine the number of indices in each buffer element, and place them in an array.
3. MapIndicesToMatrixColumns()
   Run through each buffer element and locate the newest indices; map these indices to the matrix.
4. DetermineMatrixDimension()
   Determine the matrix dimension to decode and store the new matrix dimension.
5. RemoveUnusedPackets()
   Remove all buffer elements which does not comply with the new dimension.
6. Insert the sufficient number of buffer elements which are not removed.

The decoding matrix is now created, and it is possible to decode at least the N newest packets (assuming no packet losses).

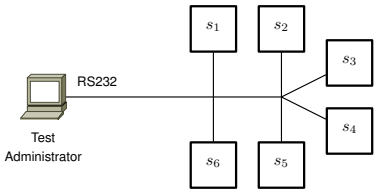
4. TEST SETUP

The system developed consists of several entities; these are

- **Test Administrator (TA):** Maintains overall control of the test setup.
- **DC:** Collects packets in the WSN.
- **OSB:** A sensor in the WSN.

The OSB is a wireless sensor board developed at Aalborg Univerity in collaboration with Technical university Berlin. The main idea behind this sensor board is to create a fully open platform in terms of hardware, software and teaching material. It features two external connections; the Radio Frequency (RF) interface, and Serial Communication using RS232 standard (RS232) or Bluetooth interface. The RF interface is used for communication between sensors and between WSN and DC. Administration of the sensors are performed using the RS232 interface, where each sensor in the WSN is connected to TA, from where the test is administered, see Figure 3.

The TA can e.g. describe, in detail, which packets are generated where and at which time. When TA informs a sensor to generate a packet, it is encoded into the PNC administered buffer on the respective sensor, and send to a subset of listening sensors within its range which will drop the packet according to a set Drop Packet Probability (DPP).



3: TA connected to sensors s_1 – s_6 using serial communication; RS232.

The DC is given as the union of a PC and an OSB:

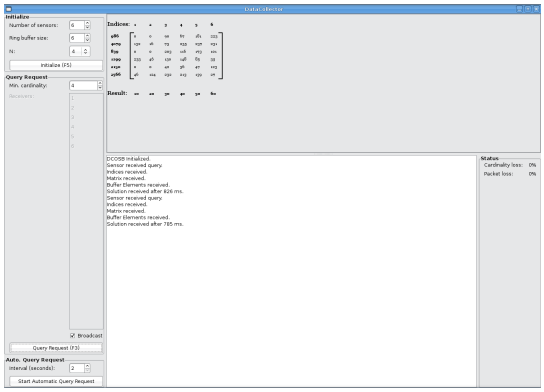
- OSB in Data Collector (DCOSB), because of its radio interface and its ability to perform Gaussian Elimination (GE) solving the linear system.
- PC in Data Collector (DCPC), because it can display measurements live as seen in Figure 4.

5. ENERGY CONSUMPTION TESTS

The energy measurements evaluates which parts of the system that can be optimized to decrease the energy consumption. This can be used to determine the energy consumption when e.g. the WSN is scaled into a larger network.

The results are found having all entities set with $N = 4$. The tests performed are made by separating each functionality in both OSB and DCOSB, which are:

1. Energy consumed when OSB transmits a data packet.



4: The Data Collector application showing the decoding matrix and the result of the decoding.

2. Energy consumed when OSB receives a data packet and encodes it into its buffer.

3. Energy consumed when OSB receives a query request from DC and transmits the buffer elements having the minimum cardinality 4.

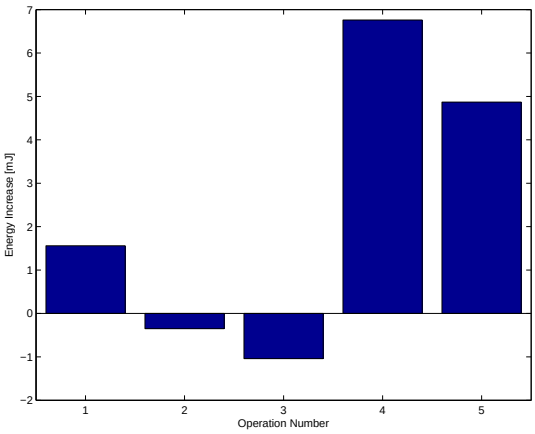
4. Energy consumed when DCOSB sends a query request packet and decodes the buffer elements received.

5. Energy consumed when DCOSB decodes the buffer elements received.

When performing the decoding on DCOSB, six buffer elements are received, which creates a 6×6 decoding matrix.

A comparison of the energy consumed are shown in Figure 5. Note that the first three tests are energy consumed on OSB, and test four, and five are on DCOSB.

The results are discussed in the following.



5: Energy increase for each operation in PNC compared to idle operation.

It is clear, that the generation of a data packets consumes energy. It is assumed that this is primarily due to the transmission of the packet using the RF module. More interesting are the two tests where the OSB receives a packet from the RF module, because the energy consumed decreases below idle operation when receiving.

On DCOSB the energy consumed for decoding is significant. This can be seen, as test four and five illustrates the decoding with and without the query request sequence. The decoding is performed on a 6×6 matrix, and it is valid to assume that the energy will increase as the matrix dimensions grows, making the operation the most energy consuming.

It is worth noting, that the decoding operation is the most energy consuming, and that this is one of the reasons why PNC benifits over traditional NC, as the decoding is omitted on the power limited sensors, and instead placed on the more powerfull DC.

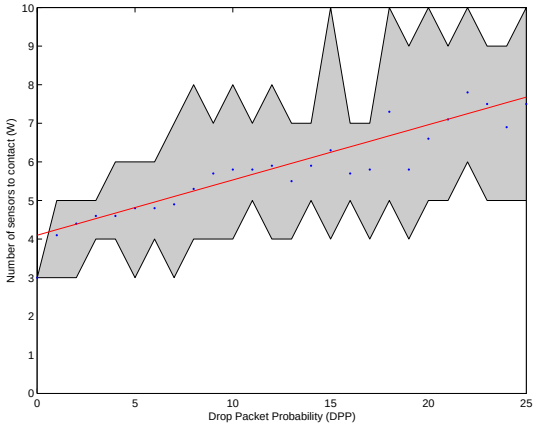
6. PACKET LOSS TEST

This test evaluates the performance of PNC in an environment with packet losses.

To do this test, the packets are distributed uniformly throughout the WSN, by having the packets distributed from a sensor that is not getting a query request from DC.

The test is performed with ten test runs for each DPP. Ten runs is not enough to get reliable results, however it give some tendency of the results. The test is performed for DPP in the interval $0 < DPP < 25\%$, because an increase of DPP will require additional (not available) sensors.

Figure 6 shows a graph of the result. The dotted points are the average values of the number of sensors to query before decoding is possible, and the grey area shows the highest and smallest number of sensors contacted. Finally, the line is a first order regression of the dotted points to estimate the tendency.



6: Number of sensors to contact versus packet loss.

It is seen from the test that PNC can be used to extract the newest data from a WSN with packets loss probabilities up to 25%.

Comparing the results to a WSN that does not use PNC, it would be harder to gather the six newest data packets. DC would have to contact all the sensors within the entire WSN to determine the newest packets in the network.

7. CONCLUSION

The test shows linear tendency in terms of the probability to lose packets and the number of sensors to contact required to decode on the DC. The ideal PNC scheme states that at most $N + \sqrt{N}$ sensors should be contacted to decode the packets; the evaluation shows that in average, if $N + \sqrt{N}$ sensors are contacted, then decoding is possible if the packet loss is under 15%.

The ideal scheme where DPP is zero shows the minimum number of sensors to be contacted for decoding. In this work it is assumed that each sensor contacted delivers two buffer elements, which the sensors can if the number of packets received or generated are even (because $\sqrt{4} = 2$).

The test suite created to test a WSN still makes assumptions in some points to make the setup simple. The main problem when creating a real implementation, is the identification of which sensed value is the newest. Currently TA administers this by informing each OSB which packet index it should use for the generated packet, but in a real scenario this centralized approach is not possible. Instead a distributed packet index creation is needed, which introduces further problems.

The sensors can of cause estimate which packets are newest relative to when they generate a packet and receive a packet from neighbours, but problems may arise when the DC arrives, because DC have the ability to probe sensors not within range, making the relative indices problematic.

If each sensor have their own index source, the mapping of sensor id and this index can be used to uniquely identify a packet, but no mechanism controls which packets are newest, only the newest packet per sensor can be identified.

Nevertheless, this work has shown the feasibility to implement PNC on very simple communication platform such as the OpenSensor Board, which main computation entity is a five dollar DSP. Furthermore a PNC testbed has been establish which will be used in the future to look into the indexing problem of the newest sensor data.

8. REFERENCES

[1] A. Dimakis, P. Godfrey, M. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 2000–2008, May 2007.

[2] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 4:2235–2245 vol. 4, 13-17 March 2005.

[3] Mobile devices at Aalborg University. OpenSensor, 2008. `http://mobiledevices.kom.aau.dk/opensensor`.

[4] D. Wang, Q. Zhang, and J. Liu. Partial network coding: Theory and application for continuous sensor data collection. *Quality of Service, 2006. IWQoS 2006. 14th IEEE International Workshop on*, pages 93–101, June 2006.

[5] J. Y. Wang, G. Yang, X. H. Lin, Z. Q. He, and J. R. Lin. Continuous data collection in wireless sensor networks through pnc and distributed storage. *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, pages 2568–2571, 21-25 Sept. 2007.