Partial Network Coding: Concept, Performance, and Application for Continuous Data Collection in Sensor Networks

DAN WANG
The Hong Kong Polytechnic University
QIAN ZHANG
Hong Kong University of Science and Technology
and
JIANGCHUAN LIU
Simon Fraser University

Wireless sensor networks have been widely used for surveillance in harsh environments. In many such applications, the environmental data are continuously sensed, and data collection by a server is only performed occasionally. Hence, the sensor nodes have to temporarily store the data, and provide easy and on-hand access for the most updated data when the server approaches. Given the expensive server-to-sensor communications, the large amount of sensors and the limited storage space at each tiny sensor, continuous data collection becomes a challenging problem.

In this paper, we present *partial network coding* (PNC) as a generic tool for the above applications. PNC generalizes the existing *network coding* (NC) paradigm, an elegant solution for ubiquitous data distribution and collection. Yet, PNC allows efficient storage replacement for continuous data, which is a deficiency of the conventional NC. We prove that the performance of PNC is quite close to NC, except for a sub-linear overhead on storage and communications. We then address a set of practical concerns toward PNC-based continuous data collection in sensor networks. Its feasibility and superiority are further demonstrated through simulation results.

Categories and Subject Descriptors: E.4 [**Coding and Information Theory**]: Nonsecret Encoding Schemes; H.3.2 [**Information Storage**]: File Organization

General Terms: Theory, Design

Additional Key Words and Phrases: Sensor networks, Random linear coding, Network coding

Author's addresses: D. Wang, Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, Email: csdwang@comp.polyu.edu.hk; J. Liu, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, V5A1S6, Email: jcliu@cs.sfu.ca; Q. Zhang, Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, Email: qianzh@cs.ust.hk.

A preliminary version of this paper is in Proc. 14th IEEE International Workshop on Quality of Service (IWQoS'06), New Haven, CT, June, 2006.

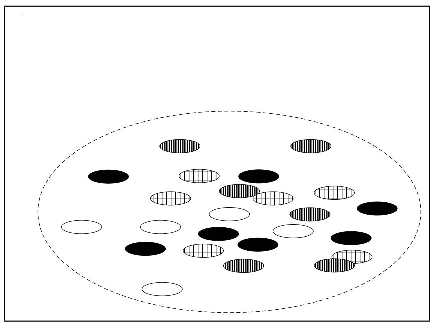
1. INTRODUCTION

A wireless sensor network consists of a large collection of sensor nodes, and is often deployed in an open area with no traditional wired or wireless network support. Each sensor node is a small device. It is not only short of battery power, but also restrained by memory storage. As a result, one sensor can store only a small amount of data collected from its surroundings, and a large quantity of sensor nodes have to work collaboratively for data gathering, storing, and replicating. To collect the data from the sensor nodes, an agent or base station (referred to as a *server* in this paper) functions as an intermediate gateway between a sensor network and the remote world.

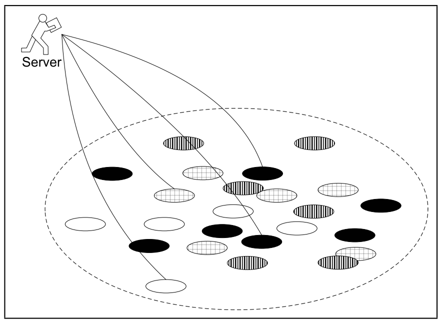
Many recent studies are interested in data collection from harsh and extreme environments [Dimakis et al. 2005][Widmer and Boudec 2005]. In these environments, the communications between the sensor nodes and the server can be expensive and scarce, and the data are collected occasionally. In each data collection, a fast data retrieval is usually desired [Dimakis et al. 2005]. Typical examples include the habitat monitoring system in Great Duck Island [Mainwaring et al. 2002]; some birds are notoriously sensitive to human intervention, and thus, data collection are done occasionally. In each collection, the presence of human being should be minimized and, hopefully, far away from the habitat center. Applications of monitoring systems in chemical plants also share similar characteristics, where the technicians occasionally approach the sensing area to collect data and each data collection should be performed quickly for safety purposes.

In the current popular data collection techniques, a server sends out a query to a root sensor and the root sensor spreads the query to the sensor network. The data are then routed from the source sensors to the root sensor. This collection technique, however, is not suitable for the applications described above. First, this technique can introduce a long delay in each data collection due to data searching and aggregation [Krishnamachari et al. 2002][Wang et al. 2005]. Second, this technique is beneficial if data can be aggregated so that the payload will be reduced in the intermediate nodes. If raw data are required, then the root sensor will be burdened by uploading all data from the sensor network to the server. Third, in many situations, some part of the sensor network may not be accessible due to failure. A ubiquitous access is thus suggested in [Dimakis et al. 2005]. In this scenario, data are redundantly stored in the sensor network and the server just randomly contacts a few sensor nodes to retrieve data. This server accessing (also known as *blind access*) is easy to implement. If the data can be retrieved accurately, the scheme is also much faster. In addition, it inherently distributes the communication cost from the root sensor to multiple sensor nodes, which balances the load.

Unfortunately, as illustrated in Fig. 1, this straight forward approach may introduce large replication. Redundancy management have been studied in many known coding algorithms, e.g., different types of erasure codes [Blahut 1983]. Most of these codes, however, are generated at a central entity and then distributed to different storage locations. This is not realistic in our application, because no sensor node is capable to store all the data, let alone to perform complicated encoding operations. A potential solution rises from *random network coding* [Dimakis et al. 2005][Medard et al. 2005][Widmer and Boudec 2005], which distributively manipulates the data



(a)



(b)

Fig. 1. An illustrative example of blind access. There are 4 different data segments distributed in the network. Each sensor (represented by a small circle) can store only one data segment (represented by the texture). (a) Server is absent. (b) Server access. The server randomly contacts several sensors to upload data. In this example, the server contacts 4 sensors, and unfortunately obtains only 3 different data segments.

in each node. Such operations combine all (say, N) data segments, making the coded data segments being equivalent to each other in decodeability. Thus, each sensor can store a small number of data segments and the server can decode all the original data as long as N combined data segments are collected. A fast and load-balanced data collection is then realized.

A key deficiency of this conventional network coding is the lack of support for removing obsolete data. Removing obsolete data is crucial for the sensor network. First, the resource of the sensor storage has a limit. If the sensor network gets unattended, e.g., because the weather prohibits the server to perform the data collection for a long time; the total data to get stored may exceed the total storage space of the entire sensor network. In many practical applications, new data has higher value than old ones. Thus, the sensor network needs to have the ability to remove old data in order to accommodate newly collected ones. Second, collecting all the data may consume too much energy; and the more data collected, the larger number of sensors need to be queried. To remove data in conventional network coding scheme, the coded segments have to be first decoded; and then re-encode into new combined data. This set of operations is time- and resource-consuming

[Gkantsidis and Rodriguez 2005]. Even worse, given that a sensor can only store a partial set of the combined data, it is impossible to carry out decoding operations in each individual sensor.

In this paper, we present *Partial Network Coding* (PNC), which effectively solves the above problems. PNC inherits the blind access capability of NC, and yet achieves the following salient features: 1) It allows a higher degree of freedom in coded data management, in particular, decoding-free data removal; 2) Its computation overhead for encoding and decoding is almost identical to the conventional network coding scheme; and 3) We proved that its performance is quite close to the conventional network coding scheme as well, except for a sub-linear overhead on storage and communications. We also address a set of practical concerns toward building and maintaining a PNC-based sensor network for continuous data collection and replacement. The feasibility and superiority of PNC are further demonstrated through simulation results.

The remainder of this paper is organized as follows. In section 2, we present the related work. We briefly introduce the system model and explain the motivations in section 3. The theoretical foundations for partial network coding are established in section 4. In section 5, we discuss the practical concerns toward using the partial network coding in sensor networks and show a distributed protocol design. The performance evaluation of PNC is presented in section 6. Finally, we conclude the paper and discuss future directions in section 7.

2. RELATED WORK

Wireless sensor networks have been extensively studied recently and surveys can be found in [Akyildiz et al. 2002][Al-Karaki and Kamal 2004][Estrin et al. 1999]. In many applications, a sensor network is query based [Madden et al. 2002], where a server queries the sensors, and the latter cooperatively response as a single entity. For such queries as Maximum, Minimum, and Average [Wang et al. 2005], a popular scheme is to construct a tree among the sensor nodes with the root being responsible for collecting data. This scheme works well if the data can be aggregated in the intermediate sensors [Heinzelman et al. 2000][Intanagonwiwat et al. 2000]. In our application, we are interested in blindly collecting the up-to-date raw data from the sensor network, which calls for different solutions.

Coding is a powerful tool for randomized data storage and collection. A typical coding scheme is the *erasure codes* [Blahut 1983][Lin and Costello 2004], in which a centralized server gathers all N data segments and builds C coded segments, $C \geq N$. If any N out of C coded segments are collected, the original data segments can be decoded [Gallager 1963][Luby 2002]. A practical investigation of these codes can be found in [Plank and Thomason 2004]. As mentioned before, these centralized operations are not suitable for our application environment that involves a large quantity of tiny sensors. An alternative is network coding [Ahlswede et al. 2000][Zhu et al. 2004], which distributes the encoding operations to multiple nodes.

Network coding was first introduced in [Ahlswede et al. 2000] to improve the performance of multicast routing. In the proposal, a router in the network can not only relay the data packets, but also code the data packets. The throughput of the network can thus be significantly improved. To guarantee decoding of the the coded data packets by the receivers, a centralized management of the co-efficient matrix is needed. *Random linear coding* [Medard et al. 2005] was proposed thereafter to relax this central control feature. The co-efficient of the codes are picked randomly from a finite field by each individual node. It has been shown [Medard et al. 2005] that if the finite field is chosen appropriately (large enough), then the linear dependency of the co-efficient is negligible and the probability of successful decoding of the code words is high. Our work also incorporates this feature.

Network coding was later suggested for efficient data storage and distribution [Deb and Medard 2004]. Using network coding for data distribution is further theoretically studied in [Medard et al. 2005], and a practical system for random file distribution is presented in [Gkantsidis and Rodriguez 2005]. Recently, network coding and its related extensions have been introduced in wireless sensor networks for ubiquitous data collection [Dimakis et al. 2005][Widmer and Boudec 2005]. In these studies, the data segments to be collected are static and fixed. On the contrary, we focus on continuous data, where obsolete data have to be evicted from a limited buffer. The proposed partial network coding complements the previous studies by demonstrating a fully localized algorithm that allows the removal of the obsolete data.

Our application scenario is also closely related to the *Delay Tolerant Network* (DTN) or extreme network architecture [Cerf et al. 2004][Fall 2003][Jain et al. 2004]. A typical example is the ZebraNet in Africa [Juang et al. 2002], where researchers have to travel to the sensor network in person to collect data. Other recent examples can be found in [Dimakis et al. 2005][Ho and Fall 2004][Wang et al. 2005][Widmer and Boudec 2005]. One important feature of these networks is that each node needs to store data temporarily and submits data when needed, but, again, they generally assume that the data are never obsolete, which is different from our focus.

3. MOTIVATIONS AND PRELIMINARIES

3.1 Model and Notations

We now give the a formal description of the system. We assume that the total number of up-to-date events to be recorded in the whole system is N. Each event (e.g., the temperature goes beyond 100°C), whenever generated, is recorded by all the sensors in the sensor networks. The event is represented by one data segment, denoted by c_j, and $c_{j'}$ is fresher than c_j if $j' > j$[1]. Similar to existing studies on linear network coding, we use $\sum_{j=0}^{N-1} \beta_j \times c_j$ to generate a coded data segment f_i, where $\overrightarrow{\beta} = [\beta_0, \beta_1, \cdots, \beta_{N-1}]$ is a co-efficient vector. Each item β_i is randomly generated from a finite field F_q. Since the coding can be viewed as a combination process, f_i is also referred to as a *combined* data segment, and c_j as an *original* data segment. Notice that the size of f_i remains equal to c_j. We define the *cardinality*

[1]We emphasize that the data should be considered as a generalized data, besides just specific numerical readings. For example, it can be a time mark, e.g., all sensors record the time when certain event happens. The time granularity should be reasonably discrete so that all the sensors is able to record the same data. The data can also be some predefined interesting events, e.g., observation of wild animals, etc. In general, if the "distance" between different events is sufficient large then there will be no misunderstandings among the sensors.

Table I. List of Notations

Notation	**Definition**
N	Number of original data segments to be collected
B	Buffer size of each sensor
c_j	Original data segment with index j
β_j	Coding coefficient for c_j
f_i	Combined data segment with index i
f_i^k	f_i, with the oldest original segment k
$\hat{k}$	Cardinality of the combined data segments
W	Number of sensors queried by the server for each access
q	Size of finite field for coefficients

of f_i to be the number of original data segments it contains, and the *full cardinality* of the system is the highest possible number, i.e., N.

The total number of sensors in the network is M, and each has a buffer of size B ($< N$) for storing the data segments. For each server access, W sensors are to be contacted and, without loss of generality, each sensor will upload one data segment from its buffer.

A summary of the notations can be found in Table I. Clearly, to obtain all the N original data segments, we must have $W \geq N$, and even so, not all the segments are necessarily obtained in one access. Consider a naive data storage and collection scheme without any coding. Assume that B segments (out of the N up-to-date original segments) are stored in each sensor's buffer and the distribution is uniform. The server blindly collects data from N sensors. The success ratio for this naive scheme is $\prod_{i=0}^{N-1} \frac{N-i}{N}$[2]. Here, the success ratio serves as the major evaluation criterion in our study, and is defined as follows:

Definition 1. *(Success Ratio) The success ratio is the probability that a scheme successfully collects all the N original data segments. The default settings of W and B are $W = N$ and $B = 1$, which are their lower bounds for valid schemes.*

For the naive scheme, its success ratio is a decreasing function of N. As shown in Table II, even for $N = 2$, the probability is barely 50%, and the performance is extremely poor for larger N.

3.2 Network Coding based Collection: Superiority and Problems

We now show that network coding can significantly increase the success ratio. With network coding, all data segments are stored in a combined fashion, and the N original data segments can be decoded by solving a set of linear equations after collecting any N combined data segments. A necessary condition here is that the

[2]One may suspect that if the server specifically requests data segments from each sensor node, the performance may improve. Unfortunately, this is not always the case. Consider that the buffer size of each sensors to be $B = 1$. Assume there are two pieces of information X and Y to be collected. The server contacts two nodes n_1 and n_2. The probability without server specification is 50% (The possible combinations from n_1 and n_2 are $(n_1, n_2) = (X, X), (Y, Y), (X, Y), (Y, X)$; and the last two succeed.). The probability with server specification is only 25% (wlog, the server specifically requests X from n_1 and Y from n_2. The possible combinations from n_1 and n_2 are still $(X, X), (Y, Y), (X, Y), (Y, X)$; and only the third one succeeds.) Intuitively, as the server does not know which sensor has what information, it can not request "correctly".

Table II. Sucess ratio of the naive scheme ($W = N, B = 1$)

N	Success Ratio	N	Success Ratio
2	0.5	6	0.0154321
3	0.222222	7	0.0061199
4	0.09375	8	0.00240326
5	0.0384	9	0.000936657

Table III. Probability of Linear Independency as a Function of Finite Field Size (q).

q	Probability	q	Probability	q	Probability
2^1	0.288788	2^5	0.967773	2^9	0.998043
2^2	0.688538	2^6	0.984131	2^{10}	0.999022
2^3	0.859406	2^7	0.992126	2^{11}	0.999511
2^4	0.933595	2^8	0.996078	2^{12}	0.999756

coefficient vectors must be linearly independent. This is generally true for a large enough field size q [Medard et al. 2005]. As shown in Table III, the probability of linear independency is over 99.6% for $q = 2^8$, and this is almost independent of N. As such, for the network coding based data storage and collection scheme, the success ratio with $W = N$ and $B = 1$ is close to 100%.

So far we have focused on static data only. In network coding, it is easy to combine new data segments to existing data segments, which increases the cardinality. The reverse operation is difficult, however. Specifically, to remove a data segment, we have to first decode the data segments and combine the remaining data segments to a new one. This is time- and resource-consuming for power limited sensors. Even worse, it is often impossible for $B < N$, because decoding requires N combined data segments. On the other hand, for continuously arrived data, if we keep obsolete data segments in the system, we have to upload more data segments than necessary to the server for a successful decoding of N needed data segments. Eventually, the buffer will overflow, and the system simply crashes. This becomes a key deficiency for applying network coding in continuous data collection.

4. PARTIAL NETWORK CODING BASED DATA STORAGE AND REPLACEMENT

In this section, we show a new coding scheme that conveniently solve the problem of data removal, thus facilitating continuous data management. Our coding scheme enables the combination of only part of the original data segments, and we refer to it as *Partial Network Coding (PNC)*, cf. *network coding* (*NC*) and no network coding at all *(Non-NC)*.

4.1 Overview of Partial Network Coding

In PNC, instead of having full cardinality of each combined data segment, we have varied cardinalities from 1 to N. Formally, for original data segments $c_0, c_1, \dots, c_{N-1}$, we have a coding base $\mathcal{B} = \{f^k | f^k = \sum_{j=k}^{N-1} \beta_j \times c_j, k \in [0, \dots, N-1], \beta_j \in F_q\}$. We use $\overrightarrow{\beta} = [\beta_{N-1}, \beta_{N-2}, \dots, \beta_k]$ to denote a co-efficient vector. We omit this $\overrightarrow{\beta}$, however, and use $f^k = [c_{N-1}, c_{N-2}, \dots, c_k]$ for ease of exposition if the context is clear. Notice that if $\hat{k}$ denote the cardinality of a combined data segment, then the

$$\begin{aligned} f^0 &= [c_3, c_2, c_1, c_0] \\ f^1 &= [c_3, c_2, c_1] \\ f^2 &= [c_3, c_2] \\ f^3 &= [c_3] \end{aligned}$$

Fig. 2. An example of the coding base for PNC with $N = 4$.

cardinality of f^k can be calculated by $\hat{k} = N - k$. The coding base for $N = 4$ is illustrated in Fig. 2. We may further drop the superscript k if the cardinality of the combined data segment is clear in its context.

In our application scenario, each sensor stores only a subset of these combined data segments given buffer size $B < N$. The storage for each sensor is $\mathcal{S} = \{f_i^k | f_i^k \in \mathcal{B}, 0 \le i \le B - 1\}$. Again, we may use f_i provided that k is clear in the context to represent the ith combined data segment in this sensor. An illustrative example is shown in Fig. 3, which also includes the corresponding NC and Non-NC schemes. From 3(c), we can see that, when a new c_4 is generated and c_0 becomes obsolete, sensors s_0 and s_4 can simply drop the longest combined data f_0 in their respective buffers. The buffers of s_0 and s_4 then become $\{f_0 = [c_4, c_3, c_2], f_1 = [c_4]\}$ and $\{f_0 = [c_4, c_3], f_1 = [c_4]\}$, respectively. This simple example demonstrates the salient feature of PNC, that is, removing the obsolete data without decoding.

4.2 Encoding and Decoding

We now give a formal description for the encoding and decoding process of partial network coding.

The encoding of PNC is an incremental process and to be performed by each sensor node. Every time one original data segment c is encoded with a combined data segment f^k. Formally,

```
**Encoding**$(f^k, c)$
  Randomly generate $\beta_i$ from $F_q$.
  $f^{k+1} = f^k + \beta_i c$. (This addition is performed in $F_q$).
```

The encoding process is a sub-function of the data replacement algorithm performed at each sensor. In Section 4.4, we will show how each sensor performs data replacement for the combined data segments in its buffer.

The decoding of PNC is performed by the server. The server collects W combined data segments $\mathbf{f} = [f_0, f_1, \ldots, f_{W-1}]$. The server also collects the co-efficient vectors $[\overrightarrow{\beta_0}, \overrightarrow{\beta_1}, \ldots, \overrightarrow{\beta_{W-1}}]$ which forms a $W \times N$ co-efficient matrix $\mathbf{A} = [a_{ij}] = [\overrightarrow{\beta_i}]$. The decoding process is basically to solve a set of linear equations given the N original data segments $\mathbf{c} = [c_0, c_1, \ldots, c_{N-1}]$ to be the variables. Formally, we look for the solutions of $\mathbf{f} = \mathbf{Ac}$. We apply Gaussian Elimination as follows.

```
**Decoding**$(\mathbf{f}, \mathbf{A}, \mathbf{c})$
  Perform Gaussian Elimination on $\mathbf{A}$ and $\mathbf{f}$ to obtain an echelon form.
  If $a_{N-1,N-1} \neq 0$ then $\mathbf{c} = \mathbf{A}^{-1}\mathbf{f}$,
    else the rank of $\mathbf{A}$ is less than $N$ and the set of equations is not decodeable.
```

We will discuss how the server collects the combined data segments and co-

$$s_0 : \{c_3, c_1\},\ s_1 : \{c_1, c_0\}$$
$$s_2 : \{c_3, c_0\},\ s_3 : \{c_3, c_1\}$$
$$s_4 : \{c_3, c_2\},\ s_5 : \{c_2, c_0\}$$

(a) Non-NC

$$s_0 : \{f_0 = 5c_3 + 2c_2 + 3c_1 + 4c_0,\ f_1 = 7c_3 + 2c_2 + 3c_1 + 4c_0\}$$
$$s_1 : \{f_0 = 3c_3 + 2c_2 + 10c_1 + c_0,\ f_1 = 10c_3 + 2c_2 + 5c_1 + c_0\}$$
$$s_2 : \{f_0 = 2c_3 + 5c_2 + 2c_1 + 4c_0,\ f_1 = c_3 + 15c_2 + 6c_1 + 3c_0\}$$
$$s_3 : \{f_0 = c_3 + 18c_2 + 9c_1 + 4c_0,\ f_1 = c_3 + 8c_2 + 9c_1 + 14c_0\}$$
$$s_4 : \{f_0 = 5c_3 + 2c_2 + 3c_1 + 4c_0,\ f_1 = 2c_3 + 6c_2 + 3c_1 + 4c_0\}$$
$$s_5 : \{f_0 = 7c_3 + 7c_2 + 9c_1 + 5c_0,\ f_1 = 8c_3 + 8c_2 + 8c_1 + 4c_0\}$$

(b) NC

$$\begin{array}{lcll}
s_0 & : & \{f_0 = [c_3, c_2, c_1, c_0], & f_1 = [c_3, c_2]\} \\
s_1 & : & \{f_0 = [c_3, c_2, c_1], & f_1 = [c_3]\} \\
s_2 & : & \{f_0 = [c_3, c_2, c_1], & f_1 = [c_3, c_2]\} \\
s_3 & : & \{f_0 = [c_3, c_2], & f_1 = [c_3]\} \\
s_4 & : & \{f_0 = [c_3, c_2, c_1, c_0], & f_1 = [c_3]\} \\
s_5 & : & \{f_0 = [c_3, c_2, c_1], & f_1 = [c_3, c_2]\}
\end{array}$$

(c) PNC

Fig. 3. Data distribution in 6 sensors (s_0 through s_5) each with two storage units. (a) Non-NC, only original data segments are stored, (b) NC, combined data segments are stored where the cardinality of each segment is N, (c) PNC, data are stored in a combined fashion where the cardinality are arbitrary. We omit the coefficients for each combined data segment in PNC.

efficient vectors in practice in Section 5.1. Again, we emphasize that W ($\geq N$) indicates the number of transmission needed between the server and the sensors. Therefore, it is directly related to the efficiency of the PNC scheme. We will analyze in Section 4.5 their relationship and show that a sublinear overhead ($W = N + \sqrt{N}$) is sufficient enough to guarantee successful decoding.

4.3 Distribution of Cardinality

Partial network coding intrinsically manages the cardinality of the combined data segments by setting some of the co-efficient to zeros. It is not difficult to see that, for a server collection, if the contacted nodes provide combined data segments with high cardinalities, then the success ratio will be higher[3]. We summarize this in the following two observations.

Observation 1. *The success ratio is maximized if every node provides the client the combined data segment with the highest cardinality from its buffer.*

Observation 2. *Consider time instances t_1 and t_2. If at t_1, the probability for each node to provide high cardinality data segments is greater than t_2, then success*

[3]On the other hand, if the sensor network does not have, say, the combined data segment with the highest cardinality (N), no matter how many data segments the server collects, it is impossible for the server to decode. This is because in the co-efficient matrix, the co-efficient associated with variable c_0 (the oldest non-obsolete original data segment) is 0.

ratio for a data retrieval at t_1 is higher than that at t_2.

Generally speaking, in each particular time instance, it is ideal for the system to have combined data segments of cardinalities as high as possible. In an extreme case, if all the combined data segments in the system have cardinality N, the success ratio is 100% but the system is essentially reduced to the conventional network coding based. Once a new data segment arrives, all the combined data segments will have to be deleted to make room for the new segment. For subsequent server collections, no data segment but the newest one can be answered.

Since the data arrival and the server collection times are application-specific, an optimal choice of the cardinality distribution thus depends on the requirement of each particular application and how the overall profit function is defined. We now consider the performance of PNC with a uniform cardinality distribution throughout the lifetime of the system, i.e., for each collected data segment, the probability of encountering any cardinality should be $\frac{1}{N}$. This distribution does not favor data arrivals or server collection at any particular time, and is thus applicable to a broad spectrum of applications that have no specific arrival/collection patterns. It also serves as a baseline for further application-specific optimizations. We show in the next sub-section the storage and data replacement scheme to maintain the uniform cardinality of the entire sensor network.

4.4 Data Storage and Replacement in PNC

In our network, a sensor can not store data segments of all the different cardinalities in its limited buffer. As shown in Fig. 3, a sensor with a buffer size of two segments will never see all the four possible data segments. Besides, it is impossible for the sensors to know exactly what other sensors store. As such, maintaining the uniformity of cardinalities becomes a great challenge in partial network coding.

We solve this problem by a *Data Replacement* algorithm locally executed at each sensor (Fig. 4). It translates the uniformity maintenance problem to a uniform configuration for the initial distribution; the latter is much easier to achieve.

```
Algorithm Data Replacement($c_n$)
  $c_n$: newly recorded data
  for $i = 1 \dots B$
    if cardinality($f_i$) $< N$,
      Encoding($f_i, c_n$)
    else
      $f_i = \beta_n c_n$
```

Fig. 4. Data Replacement Algorithm.

THEOREM 1. *If the cardinality is uniformly distributed, then after executing the Data Replacement algorithm (Fig. 4), the distribution of the cardinality remains uniform.*

PROOF. If the distribution of the cardinality is uniform, then the probability that a combined data has cardinality $\hat{k}$ is $\frac{1}{N}$ for all $\hat{k} = 1 \dots N$. After executing Data Replacement, the probability that a combined data segment has cardinality

$\hat{k}$ is equal to the probability that this segment previously has cardinality $\hat{k} - 1$, $\hat{k} = 1 \ldots N - 1$, and the probability for a combined data segment has cardinality 1 is equal to the probability it previously has cardinality N . Hence, the probability is still $\frac{1}{N}$, and the distribution remains uniform. □

The above theorem suggests that a uniformity is inherently maintained in data replacement, and the algorithm is fully distributed and localized. Therefore, before network depolyment, we can uniformly assign the cardinalities to the sensors. Assume $B = 1$; after the deployment, the sensor assigned with cardinality $\hat{k}$ can wait for $N - \hat{k}$ events and record and combine the $\hat{k}$ following events only. The initial cardinality distribution of the combined data in the sensors is then uniform. The above configuration can be easily generalized to larger buffer sizes.

4.5 Performance Analysis of PNC and Enhancements

We now analyze the performance of PNC, and identify its weakness. We then present two effective enhancements to improve its performance.

THEOREM 2. *The success ratio of PNC based data collection is no worse than the naive collection (Non-NC).*

PROOF. The only possibility for Non-NC to collect all the N original data segments is to collect each of them exactly once. On the contrary, for PNC, if we can collect N combined data segments with every cardinality presents, then we can decode all the original data segments. This is because, a matrix with N different cardinalities is automatically an echelon form and thus decodeable. Since the probability of collecting a specific data segment and that of encountering a cardinality are both $\frac{1}{N}$, the expected success ratio of PNC is no worse than Non-NC. Note that, some combinations without all the cardinalities can be decodable as well; hence, PNC could achieve a higher success ratio. □

It is also worth noting that, when the buffer size of a sensor increases, it can upload a data segment of higher cardinality when queried. The success ratio of PNC will thus be improved. On the contrary, for Non-NC, since each sensor can only randomly picks the data segment from its buffer for uploading, its performance remains unchanged.

We go on to compare PNC and NC. We know that, by ignoring the linear dependency of coefficients and data removal, NC achieves 100% success ratio when $W = N$ combined data segments are collected. An interesting question is thus whether PNC can achieve the same performance, or, if not, what is the overhead. To give some intuition, we see that in PNC, the chances for encountering c_{N-1} and c_0 are not identical: the most up-to-date data segment c_{N-1} is easier to collect because every combined data segment contains c_{N-1}; on the contrary, the oldest data segment c_0 (but not obsolete) exists in the combined data segment with cardinality of N only. As such, the decoding ratio with PNC after blindly accessing a subset of sensors could be lower than that with NC.

To address the above problem, we make two enhancements to the original PNC scheme. First, we extend the cardinality of the system from N to $N + \sqrt{N}$; that is, in addition to N required data segments, we store another $\sqrt{N}$ obsolete data segments in the system. These obsolete data segments makes the originally oldest

$$\begin{aligned}
f_0 &= [c_{N-1}, \dots, c_{N-\sqrt{N}-1}, \dots, c_{2\sqrt{N}-1}, \dots, c_0, \dots, c_{-\sqrt{N}}] \\
f_1 &= [c_{N-1}, \dots, c_{N-\sqrt{N}-1}, \dots, c_{2\sqrt{N}-1}, \dots, c_0] \\
f_2 &= [c_{N-1}, \dots, c_{N-\sqrt{N}-1}, \dots, c_{2\sqrt{N}-1}] \\
&\dots \\
f_{\sqrt{N}} &= [c_{N-1}, \dots, c_{N-\sqrt{N}-1}]
\end{aligned}$$

Fig. 5. A snapshot of the buffer at a sensor. We can see that f_0 has a cardinality of $N + \sqrt{N}$ (with $\sqrt{N}$ obsolete data segments combined). When a new c_N is generated, according to Data Replacement algorithm, it will be combined to all f_i, and f_0 will be discarded. f_1 however will have a cardinality of $N + 1$ (combined with one obsolete data segment c_0). We can guarantee that, at any given time, each sensor will have a data segment of cardinality at least N.

data segment relatively "younger" and therefore more likely to be collected; Second, we expand the buffer size of a sensor to $B = \sqrt{N} + 1$, which facilitates the first enhancement. With these two modifications, the following lemma shows that there is a scheme such that each sensor can upload a data segment with cardinality at least N when queried.

LEMMA 3. *By extending the cardinality of the system to $N + \sqrt{N}$ and the buffer size to $\sqrt{N} + 1$, each sensor can have a combined data segment with cardinality at least N in its buffer.*

PROOF. Consider the following storage scheme for each sensor: A sensor picks a random number $k \in [-\sqrt{N}, 0]$ (we use a negative index to denote an obsolete data segment) and stores combined data segments $f_0 = [c_{N-1}, \dots, c_k]$, $f_1 = [c_{N-1}, \dots, c_{k+\sqrt{N}}]$, $f_2 = [c_{N-1}, \dots, c_{k+2\sqrt{N}}]$, $\dots, f_{\sqrt{N}} = [c_{N-1}, \dots, c_{N-\sqrt{N}-k}]$. The difference of the cardinality between f_i and f_{i+1} is $\sqrt{N}$ for all $0 \le i \le (B - 1)$. The buffer requirement of this scheme is $\sqrt{N} + 1$, and for any k the sensor chooses, the cardinality of f_0 is greater than N. In addition, after executing the Data Replacement algorithm, the cardinality of f_0 remains greater than N until it is discarded upon the arrivals of $\sqrt{N}$ new data segments. After that, the cardinality of f_1 will be greater than N, and the iteration continues. □

A concrete example of a snapshot of the buffer at a sensor node after these two modifications is shown in Fig. 5, where we denote the $\sqrt{N}$ obsolete data with negative indices. We then have the following observation on the performance of PNC as compared to NC.

THEOREM 4. *The success ratio of PNC with $B = \sqrt{N} + 1$ and $W = N + \sqrt{N}$ is 100% (neglecting linear dependency of the coefficients).*

PROOF. From Lemma 3, the server can collect $\sqrt{N} + N$ combined data segments with cardinality at least N. For decoding, we are trying to solve a set of linear equations, of which the coefficients form a $(N + \sqrt{N}) \times (N + \sqrt{N})$ matrix. Since the cardinality of each coefficient vector is at least N, then the rank of this matrix is at least N. Therefore, we can solve the first N variables (which contributes to the rank). □

Corollary 5. *The success ratio of PNC with $B = \sqrt{N} + 1$ and $W = N + \sqrt{N}$ is identical to the success ratio of NC with $B = 1$ and $W = N$.*

In other words, after sacrificing a sublinear buffer overhead ($\sqrt{N}$) at each sensor and a sub-linear communication overhead ($\sqrt{N}$), the PNC will be able to decode all the N original data segments in a blind access as NC does.

5. PROTOCOL DESIGN AND PRACTICAL ISSUES

In this section, we address some major practical concerns, and present a collaborative and distributed protocol for continuous data collection with PNC.

5.1 Computation and Communication Overheads

Since the sensors are small and power constrained entities, the PNC operations must be light-weighted. It is known that the computational overhead for network coding lies mainly in the decoding process. This is however performed in the powerful servers. Each sensor just needs to randomly generate a set of coefficients, combine newly arrived data with those in the buffer, or remove an obsolete data segment. All of these operations are relatively simple with low costs. Another overhead is the transmission cost. For network coding based application, besides the combined data, the coefficient vectors have to be uploaded for decoding. Such overheads are generally much lower than the data volume, and our simulation results have shown that the benefits of PNC generally overcome these overheads.

An alternative way to reduce the communication overhead is using polynomial interpolation [Suli and Mayers 2003]. We change the coding base as $\mathcal{B} = \{f^k | f^k = \sum_{j=k}^{N-1} \beta^j \times c_j, k \in [0, \ldots, N-1], \beta \in F_q\}$. Notice that the major difference in this setting is that an integer β is chosen and the coefficient vector is $(1, \beta, \beta^2, \ldots, \beta^{N-1})$ (for a combined data segment of cardinality N). Therefore, only β, a constant in respect to N, is needed to be uploaded. If all β uploaded are different from each other, then the coefficient vectors are guaranteed to be linearly independent [Suli and Mayers 2003]. Therefore, the overhead is reduced substantially. This, however, does not come for free. If two sensors choose the same β, then these two combined data segments are linear dependent. The linear dependency is thus higher than the case where all items of the coefficient vectors are chosen randomly. Formally, the linear independency probability is $p = \prod_{i=0}^{N-1} \frac{q-i}{q}$, where q is the size of the finite field F_q. Fig. 6 illustrates the relationship of p, N and q. To migrate this effect, we can use larger finite field F_q. Comparing $q = 2^8$ and $q = 2^{16}$, the probability of linear independency has been remarkably improved. For different applications, this is worthwhile as the increase of F_q costs only logarithmic additional bits.

5.2 Multiple Data Pattern

In many applications, the sensor network is required to collect multiple data ranges or patterns. For example, the sensor network may need to track the temperature of multiple critical levels. Therefore, the sensors need to be invoked at different times to record different data sets. The problem here is whether to use a mixed storage with each sensor splitting its buffer to store different temperature levels, or just assign different subset of sensors to record different levels. The tradeoff is obvious: the former might record certain temperature levels incompletely if the

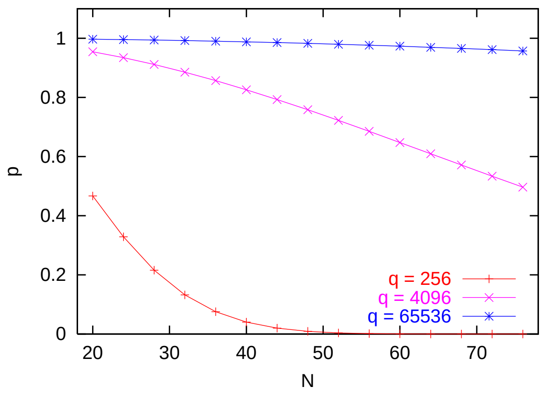


Fig. 6. Probability of linear independency as a function of the number of data segments.

buffer is too small, i.e., smaller than N; the latter, while fully recording certain levels of temperature, will risk the incapability of decoding an entire level.

The above *all or nothing* effect is also considered in [Chou et al. 2003]. Yet, for PNC based collection, we can see that a larger buffer might provide data with higher cardinalities, and it is easy to add an importance parameter in our system. That is, for important data patterns, we can use more sensors to maintain them, and thus have higher probability to successfully collect them. We will further investigate the impact of the importance parameter in the next section through simulations.

5.3 Collaborative and Distributed Implementation

To guarantee success, our PNC suffers only a sublinear overhead ($\sqrt{N}$) in buffer storage and communication cost. In practice, if N is too big, even a buffer of size $\sqrt{N}+1$ might not be available at a tiny sensor. In addition, the buffer sizes of the sensors might not be identical. To overcome these problems, the sensors can work collaboratively to provide combined data segments when queried. Specifically, they can form clusters in advance, where the members of a cluster maintain different cardinalities. A cluster can then upload one highest cardinality data segment upon accessing.

We thus suggest the following collaborative and distributed implementation. We assume that the server is interested in m data patterns and, for each pattern, N_i recent data segments, $1 \leq i \leq m$. After deployment, each sensor will send a probe message to its surrounding area to form a cluster, where the number of sensors in this cluster, n, is greater than $\sum_{i=1}^{m} N_i$. Presumably, the sensors in one cluster can reside in 1 or 2 hops from each other. If n is too large, a two tier structure can be built, where each cluster in the first tier stores the data for a single pattern. A cluster head is then selected for each cluster, which distributes a storage schedule to the sensors in its cluster. When a sensor receives a server query, it first forwards this message to the cluster head; the cluster head checks whether this query is to search a data pattern associated with its own cluster. If so, the head will notify the sensor that currently has the combined data segment of the highest cardinality to upload the data; otherwise, it will forward the query to an appropriate head that

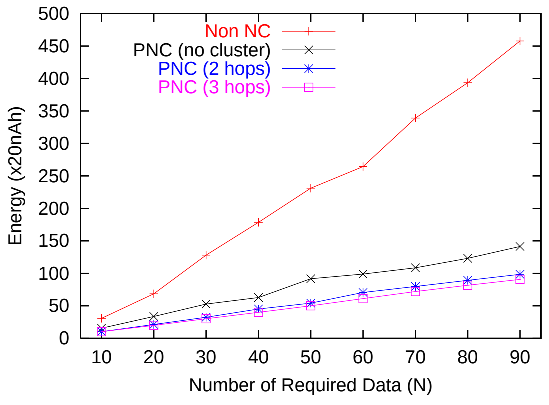


Fig. 7. Energy consumption as a function of N for different cluster radiuses.

is associated with the pattern for further processing.

6. PERFORMANCE EVALUATION

6.1 Simulation Settings

In this section, we present our simulation results for PNC-based sensor data collection. We deploy 1000 sensors randomly into a field of 10m $\times$ 10m. The distance between the server and the sensor nodes is much larger than the distance between the sensors, and, as suggested in [Lindsey and Raghavendra], we assume that there is a 10-fold difference. The server can thus access the data without necessarily entering deep into the sensor field, which is useful for data collection from a dangerous area. The default number of data segments that the server collects is the most recent 50 data segments ($N = 50$) and the default buffer size B is 1. We examine other possible values in our simulation as well. The linear equations in network coding are solved using the Gaussian Elimination [Gentle 1998], and the coefficient field is $q = 2^8$, which can be efficiently implemented in a 8-bit or more advanced microprocessor [Widmer and Boudec 2005]. To mitigate randomness, each data point in a figure is an average of 1000 independent experiments.

6.2 Comparison of Energy Consumption

Since NC does not have the capability of data removal, it will eventually lead to a crash of the system in continuous data collection. Therefore, in our simulations, we only study the performance of PNC and compare PNC with Non-NC.

We first compare the energy consumption of PNC with Non-NC. We use the energy model by [Mainwaring et al. 2002]. The energy consumption of transmitting one packet is 20nAh (10^{-9} Ampere hour). The field will generate events which are of interest in an hourly basis and the sensors will record these events. Server will randomly and occasionally approach with an expected interval of 20 hours. The server is interested in the most recent $N = 50$ data pieces. It will first randomly collect 50 data segments and if some original data segments are missing (for Non-NC) or the combined data segments can not be decoded (for PNC), then the server will send additional requests one by one, until all 50 data segments are obtained.

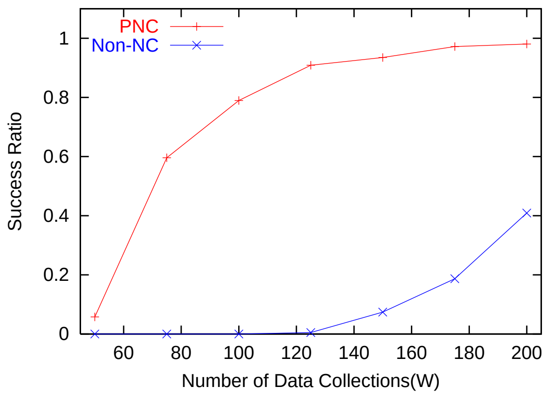


Fig. 8. Success ratio as a function of W for PNC and Non-NC.

The results are shown in Fig. 7.

It is clear that PNC performs better than Non-NC for different N. It can be seen that the energy consumptions are linear with respect to the number of required data (N), but the slope for PNC is much smaller. As a result, when N is greater than 50, the energy consumption with Non-NC is 3 to 4 times higher than that with PNC. The energy consumption with PNC is further reduced when clusters are employed (the cluster radius is set to 2 or 3 hops, respectively), because data segments with higher cardinalities could be uploaded from a larger aggregated buffer.

6.3 Performance of PNC

In our applications, when server approaches, a fast data collection is always desired. Therefore, it is better for the server to estimate the number of sensors it should query and send the query simultaneously, instead of in an incrementally fashion as previous section. Success ratio is thus a good indicator of how many sensors should be queried at once. Therefore, we use success ratio to evaluate the performance of PNC starting from this section.

6.3.1 *PNC vs Non-NC.* We revisit the performance of PNC and Non-NC using success ratio. Fig. 8 shows the success ratio as a function of the number of data segments collected (W). Not surprisingly, the success ratio increases when W increases for both PNC and Non-NC, but the improvement PNC is more substantial. For example, if 100 data segments are collected, the success ratio is about 80% for PNC; for Non-NC, after collecting 200 data segments, the success ratio is still 40% only.

6.3.2 *Effect of Buffer Size.* We then increase the buffer size from $B = 1$ to 2 and 3 to investigate its impact. We require the sensors to upload the data segment of the highest cardinality for each server access. The results are shown in Fig. 9, where a buffer increase from 1 to 2 has a notice improvement in success ratio, and a buffer of 3 segments delivers almost optimal performance. This is not surprising because there is a higher degree of freedom for storing and uploading data in a larger buffer space. We emphasize again the performance of Non-NC will not be

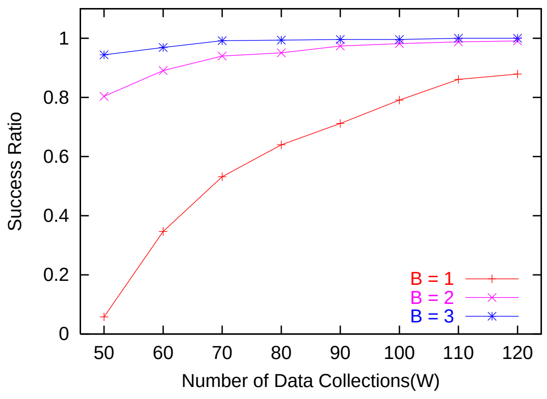


Fig. 9. Success ratio as a function of W with different buffer size.

improved by increasing the buffer size, as the sensor node can only randomly pick a data segment to upload.

In our analysis, we show that given $W = N + \sqrt{N}$, and $B = \sqrt{N} + 1$ the system guaranteed to decode N original data segments (ignoring linear dependency). In this case, the server has to decode $N + \sqrt{N}$ data segments, among which $\sqrt{N}$ are obsolete. An interesting question is thus: *Can we reduce the overhead for W but still guarantee an optimal success ratio?* Unfortunately, from Fig. 10, we can see that this unlikely happens. Among the 1000 experiments, only in 4 experiments the server successfully decodes before collecting all the $N+\sqrt{N}$ data segments. We conjecture that $\sqrt{N}$ could be a lower bound of the overhead, though it has yet to be proved.

The above two sets of results suggest that PNC works quite well for a reasonable buffer size even without extension to include obsolete data segments. Therefore, unless a guarantee is desired, the straightforward PNC is enough for most applications.

6.3.3 *Impact of N.* We then explore the impact of the cardinality N. In Fig. 11, we depict the decoding ratio for different number of original data segments (N=20, 50, and 100). The x-axis denotes the ratio between the number of data collected and the cardinality, i.e. $\lambda = \frac{W}{N}$. We can see from Fig. 11 that their differences are insignificant, and general reduce when W increases. Recall that, the performance of Non-NC decreases sharply when N increases, as shown in Table II, while NC is marginally affected by N only. These simulation results thus reaffirm that PNC inherits the good scalability of NC.

6.4 Effect of Clustering

As discussed in section 5, to surpass the limited buffer size, the sensors can form clusters to achieve a larger aggregated buffer space. In Fig. 12, we show the success ratios for a buffer limited sensor network with different cluster radiuses, i.e., the number of hops to reach the farthest sensors, ranging from 0 to 3. When the radius is 0, there is basically no cluster and a contacted sensor has to respond to the

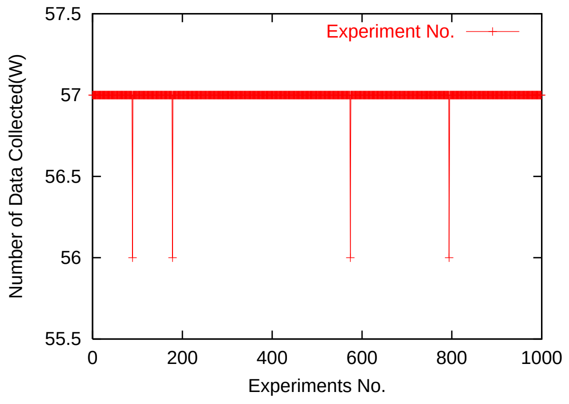


Fig. 10. Number of communication needed (W) to successfully decode N original data segments. $N = 50$ and $N + \sqrt{N} = 57$.

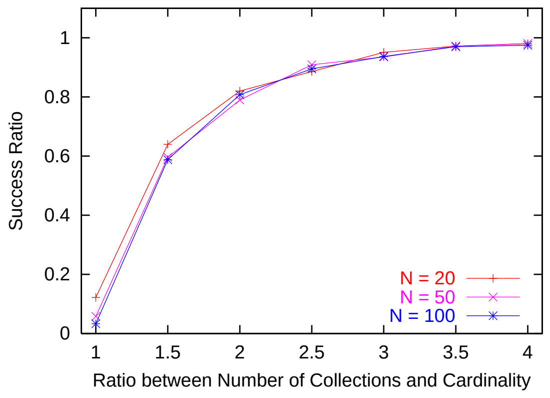


Fig. 11. Success ratio as a function of $\lambda = \frac{W}{N}$.

server immediately using data from its local buffer. We can see that the success ratio significantly increase when the clustering algorithm is enabled. For a cluster radius of 2, it is already quite close to 100%.

6.5 Impact of Multiple Pattern

We next investigate the impact of requiring the sensor network to maintain multiple data patterns, e.g., to record more than one event of interest. Fig. 13 shows the success ratio for a 4-pattern scenario. To differentiate the importance of the patterns, we have assigned different number of sensors to each pattern (in this example, 12.5%, 12.5%, 25%, and 50% of the total number of sensors).

Not surprisingly, the success ratio favors data pattern with more sensors assigned. An interesting observation is that the improvement is not uniform for all the four patterns, either. It favors first for the data pattern with the largest number of assigned sensors (50%), then the pattern with the second largest number of as-

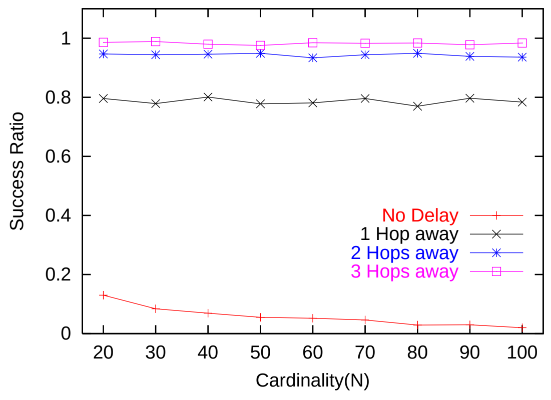


Fig. 12. Success ratio as a function of cardinality for different cluster radiuses.

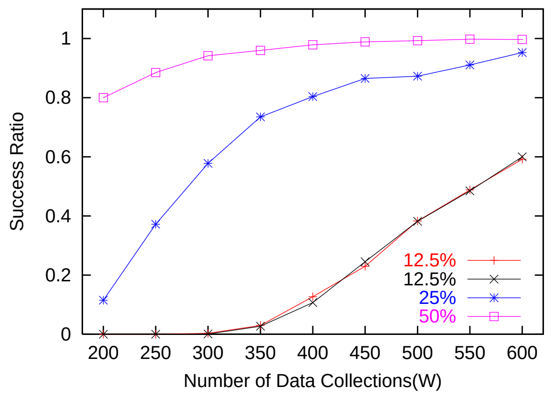


Fig. 13. Success ratio as a function of W for multiple patterns.

signed sensors (25%), and so forth. This is clearly desirable given that we want to differentiate the patterns.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we introduced a novel coding scheme, Partial Network Coding (PNC), which effectively solves the problem of removing obsolete information in coded data segments. The problem is a major deficiency of the conventional network coding. We proved that the success ratio of PNC in data collection is generally better than a non-coding scheme and is close to the conventional network coding, and this is achieved with only a sub-linear overhead on storage and communications. We then addressed several practical concerns toward implementing PNC in resource-constrained sensors, and demonstrated a collaborative and fully distributed protocol for continuous data collection in large sensor networks.

In network coding research, it is known that the higher the cardinality is, the more the benefits we could expect. Therefore, many existing schemes have focused on achieving a full cardinality in data combination; For example, the proposals in [Deb and Medard 2004][Dimakis et al. 2005][Gkantsidis and Rodriguez 2005][Medard et al. 2005] generally increase the cardinality by combining as much data as possible in intermediate nodes and then forward to others. Our work on partial network coding, however, shows that the opposite direction is worth consideration as well.

Nevertheless, there are still many unsolved issues for PNC. Beyond the practical issues toward implementing a real PNC-based sensor network, we are interested in the following two questions: First, based on our simulations, we observe that the performance of PNC is very close to NC. We therefore suspect whether the overhead of $\sqrt{N}$ reaches the potential limit of PNC? Second, our PNC is only used for data collection. We expect that an enhancement could facilitate more complicated queries, such as Max/Min, Quantile, and Average/Summation. Given its flexibility in data management, we believe that PNC can be applied in many other applications, and the solutions to the practical and theoretical issues in PNC is thus urged, especially considering the recent flourish of data streaming in numerous fields.

ACKNOWLEDGMENT

J. Liu's work was supported in part by a Canadian NSERC Discovery Grant 288325, an NSERC Research Tools and Instruments Grant, a Canada Foundation for Innovation (CFI) New Opportunities Grant, and an SFU President's Research Grant. Q. Zhang's research was supported in part by the National Basic Research Program of China (973 Program) under Grant No. 2006CB303000, the NSFC oversea Young Investigator grant under grant no. 60629203, National 863 Program of China under Grant No. 2006AA01Z228. The authors thank Dr. Wenjie Wang for valuable discussions.

REFERENCES

Ahlswede, R., Cai, N., Li, S., and Yeung, R. 2000. Network information flow. *IEEE Transactions on Information Theory 46,* 4 (July).

Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. 2002. A survey on sensor networks. *IEEE Communications Magazine 40,* 8 (Aug.), 102–114.

Al-Karaki, J. and Kamal, A. 2004. Routing techniques in wireless sensor networks: A survey. *IEEE Wireless Communications 11,* 6 (Dec.), 6–28.

Blahut, R. 1983. *Theory and Practice of Error Control Codes.* Addison-Wesley, Reading, MA.

Cerf, V. et al. 2004. Delay tolerant network architecture. Internet draft, http://www.ipnsig.org/reports/draft-irtf-ipnrg-arch-01.txt.

Chou, P., Wu, Y., and Jain, K. 2003. Practical network coding. In *Proc. Allerton Conference on Communication, Control and Computing 2003.* Monticello, IL.

Deb, S. and Medard, M. 2004. Algebraic gossip: A network coding approach to optimal multiple rumor mongering. In *Proc. Allerton Conference on Communication, Control and Computing 2004.* Urbana, IL.

Dimakis, A., Prabhakaran, V., and Ramchandran, K. 2005. Ubiquitous access to distributed data in large-scale sensor networks through decentralized erasure codes.

Estrin, D., Govindan, R., Heidemann, J., and Kumar, S. 1999. Next century challenges: Scalable coordination in sensor networks. In *Proc. ACM MOBICOM'99.* Seattle, WA.

Fall, K. 2003. A delay-tolerant network architecture for challenged internets. In *Proc. ACM SIGCOMM'03.* Karlsruhe, Germany.

Gallager, R. 1963. *Low-Density Parity-Check Codes.* MIT Press, Cambridge, MA.

GENTLE, J. 1998. *Numerical Linear Algebra for Applications in Statistics.* Springer.

GKANTSIDIS, C. AND RODRIGUEZ, P. 2005. Network coding for large scale content distribution. In *Proc. IEEE INFOCOM'05.* Miami, FL.

HEINZELMAN, W., CHANDRAKASAN, A., AND BALAKRISHNAN, H. 2000. Energy-efficient communication protocol for wireless microsensor networks. In *Proc. Hawaaian International Conference on Systems Science (HICSS'00).* Wailea Maui, HI.

HO, M. AND FALL, K. 2004. Poster: Delay tolerant networking for sensor networks. In *Proc. IEEE SECON'04.* Santa Clara, CA.

INTANAGONWIWAT, C., GOVINDAN, R., AND ESTRIN, D. 2000. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc. ACM MOBICOM'00.* Boston, MA.

JAIN, S., FALL, K., AND PATRA, R. 2004. Routing in a delay tolerant network. In *Proc. ACM SIGCOMM'04.* Portland, OR.

JUANG, P., OKI, H., WANG, Y., MARTONOSI, M., PEH, L., AND RUBENSTEIN, D. 2002. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *Proc. ACM ASPLOS'02.* San Jose, CA.

KRISHNAMACHARI, B., ESTRIN, D., AND WICKER, S. 2002. The impact of data aggregation in wireless sensor networks. In *ICDCS Workshop on Distributed Event-based System (DEBS'02).* Vienna, Austria.

LIN, S. AND COSTELLO, D. 2004. *Error Control Coding: Fundamentals and Applications.* Printice Hall, Upper Saddle River, NJ.

LINDSEY, S. AND RAGHAVENDRA, C. Pegasis: Power-efficient gathering in sensor networks. In *IEEE Aerospace Conference Proceedings.* Vol. 3.

LUBY, M. 2002. Lt codes. In *Proc. IEEE FOCS'02.* Vancouver, Canada.

MADDEN, S., SZEWCZYK, R., FRANKLIN, M., AND HONG, W. 2002. Supporting aggregate queries over ad-hoc wireless sensor networks. In *Proc. IEEE International Workshop on Mobile Computing Systems and Application (WMCSA'02).* Callicon, NY.

MAINWARING, A., POLASTRE, J., SZEWCZYK, R., CULLER, D., AND ANDERSON, J. 2002. Wireless sensor networks for habitat monitoring. In *Proc. ACM WSNA'02.* Atlanta, GA.

MEDARD, M., ACEDANSKI, S., DEB, S., AND KOETTER, R. 2005. How good is random linear coding based distributed networked storage? In *Proc. NETCOD'05.* Riva del Garda, Italy.

PLANK, J. AND THOMASON, M. 2004. A practical analysis of low-density parity-check erasure codes for wide area storage applications. In *Proc. International Conference on Dependable Systems and Networks (DSN)'04.* Florence, Italy.

SULI, E. AND MAYERS, D. 2003. *An Introduction to Numerical Analysis.* Cambridge University Press.

WANG, D., LONG, Y., AND ERGUN, F. 2005. A layered architecture for delay sensitive sensor networks. In *Proc. IEEE SECON'05.* Santa Clara, CA.

WANG, D., ZHANG, Q., AND LIU, J. 2006. Partial network coding: Theory and application for continuous sensor data collection. In *Proc. IEEE IWQoS'06.* New Haven, CT.

WANG, Y., JAIN, S., MARTONOSI, M., AND FALL, K. 2005. Erasure-coding based routing for opportunistic networks. In *Proc. ACM SIGCOMM Workshop WTDN'05.* Philadelphia, PN.

WIDMER, J. AND BOUDEC, J. 2005. Network coding for efficient communication in extreme networks. In *Proc. ACM SIGCOMM WorkShop WTDN'05.* Philadelphia, PN.

ZHU, Y., LI, B., AND GUO, J. 2004. Multicast with network coding in application layer overlay networks. *IEEE Journal on Selected Areas in Communications 22,* 1 (Jan.), 107–120.