# Scalable Ubiquitous Data Access in Clustered Sensor Networks

Yueh-Hua Lee, Alex Thomo, Kui Wu, and Valerie King

University of Victoria, Victoria BC, Canada
yhl@uvic.ca,{thomo,wkui,val}@cs.uvic.edu

**Abstract.** Wireless sensor networks have drawn much attention due to their ability to monitor ecosystems and wildlife habitats. In such systems, the data should be intelligently collected to avoid human intervention. For this, we propose a network infrastructure in which the sensor nodes are designated as "data-generating" or "data-storage" nodes. Data-generating nodes take measurements, whereas data-storage nodes make themselves available to compute and store checksums of data received from nearby data-generating nodes.
We propose a spatially-clustered architecture for our storage nodes and a coding scheme that allows a data collector to recover all original data by querying only a small random subset of storage nodes from each cluster. The size of such a subset is equal to the number of data-generating nodes that the cluster serves.
When the clustering structure of the storage nodes is unknown, we show that recovering of the original data is still possible if a random subset *of the right size* of storage nodes is selected for querying. We determine this right size so as to have a successful decoding with a probability exceeding a given threshold.

## 1 Introduction

A wireless sensor network (WSN) consists of a large number of cheap, low-power sensors with strictly limited resources. Most WSN applications collect and process sensor readings such as temperature and humidity. In applications such as the detection of fire and pollution, live data streams are delivered to a data processing center through a connected network [7, 8, 1, 5]. In some other applications; however, the network may not be connected all the time or access to live data streams may be costly and undesirable. To mention a few examples, in ZebraNet [6] that tracks wild zebras in Africa, it is very hard to access live data from zebras due to their spontaneous movement; in the habitat monitoring system in Great Duck Island [9], some birds are notoriously sensitive to human intervention, and as such, data are collected only occasionally. In these examples, the data are stored temporarily (at storage nodes) for later data access.

A system infrastructure to achieve the above goal is to deploy sensors in the field to form a distributed data storage network. The infrastructure contains three different nodes: "data-generating nodes" (or data nodes for short), "data

storage nodes" (or storage nodes for short), and "data collectors." The data nodes (e.g., the sensors on animals) takes measurements. When the data nodes are close to storage nodes, they upload their readings. The duty of the storage nodes is to encode and store the incoming data. Data collectors, which may be interested researchers using the system, will access the storage nodes at a later time. The storage nodes are usually stationary. They form an auto-configuration network, which is managed by an underlying network management scheme (e.g., clustering). The operations of the management scheme may be invisible to data nodes (zebras) and data collectors (human people), who may not be always on the monitored field.

It has been pointed out that naïve data storage without coding cannot achieve efficient data collection (cf. [3, 12]). Suppose that the number of data nodes and storage nodes is $K$ and $N$, respectively. The *ubiquitous access* property is

*to recover all $K$ data items by querying any $K$ storage nodes.*

This is exactly the goal that we want to achieve with small system overhead.

Dimakis et al. in [3] proposed Decentralized Erasure Codes (DEC) for ubiquitous access to sensor data. Dimakis et al. showed that if each data node transmits the data to at least $5\frac{N}{K}\ln K$ storage nodes, then the ubiquitous access can be fulfilled. They assumed, however, a flat underlying network structure, and thus, the communication cost for the data distribution is prohibitive when $N$ becomes large. Unfortunately, in many applications, we must deploy a great number of storage nodes to provide good coverage. In addition, the $K$ data items are decodable only with a probability of $1-\frac{K}{q}$, where $q$ is the order of the Galois field used for encoding. To achieve a high decoding probability, $q$ cannot be small, and this implies a significant overhead in calculations.

We remark that one could also achieve ubiquitous access by using Reed-Solomon (RS) codes. In such a case, the decoding is 100% certain. However, the RS code matrix is in general "denser" (i.e. with more non-zero elements) than the DEC's code-matrix, and this translates into more data messaging from the data nodes to storage nodes.

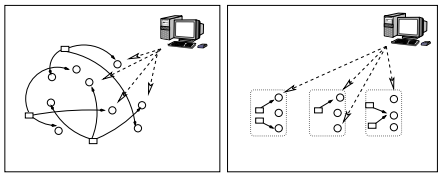

**Fig. 1.** A flat WSN (left) requires long-distance communication. Our scheme has intra-cluster communication only (right). Squares are data nodes; circles are storage nodes.

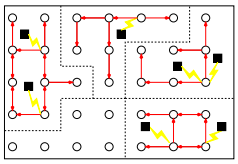

**Fig. 2.** A sensor netork of 5 clusters: The data nodes (squares) upload data to storage nodes (circles). The storage nodes are forwarding (indicated by arrows) the data among its cluster.

In this paper, we solve this problem by clustering storage nodes. Data items are propagated only within each cluster. As shown in Figure 1 (right), data distribution with our method only involves short-range intra-cluster communications. On the other hand, with a flat architecture, Figure 1 (left), the messages may be sent to faraway storage nodes. Notably, we have the "luxury" to adopt a deterministic code (such as RS codes) in a cluster because the number of storage nodes in a cluster is much smaller than their total number.

Next, we study the case when the data collectors are not fully aware of the clustering structure of storage nodes. This is motivated by the fact that the clusters may change dynamically and data collectors may not be always on the field. Most existing clustering algorithms (cf. [11]) dynamically adjust cluster heads or structure to balance energy consumption and prolong network lifetime. Therefore, for full generality, we assume that the cluster assignment is unknown to data collectors, who would like to decode all the data segments by simply querying a random set of storage nodes. With a theoretical analysis and numerical results, we show that the size of the random set can be close to $K$, the number of data nodes.

In summary, our contributions are:
1. We propose an architecture that reduces the cost to achieve ubiquitous access.
2. We propose a coding scheme which imposes no memory overhead.
3. We investigate the possibility of ubiquitous access in the case when a data collector does not know the cluster structure of the network. For this, we present a mathematical model for the decoding probability and show that, in practical cases, this probability is sufficiently high for moderate sample sizes.
4. We demonstrate that our coding scheme is more cost-efficient than DEC.

## 2 System Architecture and Encoding/Decoding scheme

Our architecture is a clustered WSN (See Figure 2). In total, there are $K$ data nodes and $N$ storage nodes. We assume that the storage nodes are stationary but data nodes may move with animals as for example in the ZebraNet. An underlying clustering algorithm partitions the storage nodes into $M$ clusters. Within a cluster, a cluster-head node maintains the full view of the cluster, including the number of data and storage nodes, node IDs, etc. Each data node takes measurements and send its readings to its nearest storage node. As the data nodes move, they transmit their data to different storage nodes belonging to different clusters. Suppose that the number of storage nodes in the $m^{th}$ cluster is $N_m$ for $1 \leq m \leq M$. We assume that $K \leq N_m$ for each $1 \leq m \leq M$. This can be easily achieved by the underlying clustering algorithm as the common assumption is that $K \ll N$.

We denote by $d_k$, for $1 \leq k \leq K$, the content of such a data-message generated by data node $k$. When a storage node receives $d_k$, it propagates $d_k$ in its cluster and updates its checksum and as follows. We denote by $s_{m,i}$, where $1 \leq m \leq M$ and $1 \leq i \leq N_m$, the code-checksums of the storage nodes of cluster $m$. Let $H$ be the code-matrix of a *systematic* $(N_m + K, K)$ RS code over a

$GF(2^w)$ field, where $2^w > N_m + K$. The encoding state of cluster $m$ adheres to the following equation

$$H \cdot \begin{bmatrix} d_1 \\ \vdots \\ d_K \end{bmatrix} = \begin{bmatrix} 1 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & 1 \\ a_{1,1} & \dots & a_{1,K} \\ \vdots & & \vdots \\ a_{N_m,1} & \dots & a_{N_m,K} \end{bmatrix} \begin{bmatrix} d_1 \\ \vdots \\ d_K \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ d_K \\ s_{m,1} \\ \vdots \\ s_{m,N_m} \end{bmatrix}.$$

Based on the above equation, a storage node checksum, say $s_{m,i}$, will be

$$s_{m,i} = a_{i,1}d_1 + a_{i,2}d_2 + \ldots + a_{i,K}d_K.$$

Of course, some data nodes may be served by other clusters. For those data nodes, we consider that they send the value of zero, which we assume is not a valid content. This coding scheme tolerates up to $N_m$ *erasures*, which means that we can find out all of $d_1, \ldots, d_K$ values by selecting $K$ storage nodes only.

Let $K_m$ be the number of non-zero $d_k$'s in the $m^{th}$ cluster. Clearly, $K_m \leq K$. The question is:

> Can we recover the $K_m$ non-zero $d_k$'s by selecting only $K_m$ (as opposed to $K$) storage nodes?

I.e. whether the $K_m$ non-zero $d_k$'s are calculated from a system of $K_m$ equations. In order to achieve this, we modify our coding state. We observe that our erasure scheme is special in that all $d_k$'s are always "erased." Hence, the problem boils down to solving (with respect to $d_k$'s) a system of linear equations obtained by selecting any $K_m$ equations from the following system of $N_m$ equations represented in matrix form as

$$\begin{bmatrix} a_{1,1} & \dots & a_{1,K} \\ \vdots & & \vdots \\ a_{N_m,1} & \dots & a_{N_m,K} \end{bmatrix} \begin{bmatrix} d_1 \\ \vdots \\ d_K \end{bmatrix} = \begin{bmatrix} s_{m,1} \\ \vdots \\ s_{m,N_m} \end{bmatrix}.$$

The above translates into asking that any submatrix of the coefficients' matrix (on the left), obtained by selecting $K_m$ rows, to be invertible.

In other words, because of our particular erasure model, we only need to find such a "nicely behaved" matrix for computing the $s_{m,i}$'s. Fortunately, an $N_m \times K$ Vandermonde matrix fits our needs. Such a Vandermonde matrix is

$$\begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & b_1 & \dots & b_{K-1} \\ \vdots & & \vdots & \\ 1 & b_1^{N_m-1} & \dots & b_{K-1}^{N_m-1} \end{bmatrix}$$

where $1, b_1, \ldots, b_{K-1}$ are $K$ different elements of the underlying $GF(2^w)$ field.

The desired property of such an $N_m \times K$ Vandermonde matrix is that every subset of $K$ rows is guaranteed to be linearly independent. Furthermore, the west part, $(1 \dots N_m) \times (1 \dots K_m)$, for $1 \leq K_m \leq K$, is also a Vandermonde matrix with the property that any set of $K_m$ rows is linearly independent.

Thus, by using the above $N_m \times K$ Vandermonde matrix as the lower part of our systematic code-matrix, and by consolidating the non-zero $d_k$'s to be always the first in the data node vector, we guarantee that with only $K_m$ equations (for $s_{m,i}$'s) we are able to calculate all the values of the non-zero $d_k$'s. Hence, we need to query only $K_m$ of the storage nodes in order to recover the contents of the $K_m$ data nodes that are in the proximity of cluster $m$.

When using this scheme, we do not need to store the coefficients since they can be computed on the fly as powers of $b_1, \dots b_{N_m}$, whereas these elements can be considered as consecutive powers of a field generator $g$.

**Remark.** Vandermonde matrices are commonly used in creating systematic Reed-Solomon codes for RAID schemes recovering from disk failures (see [10]). For this, one could start with an $(N_m + K) \times K$ Vandermonde matrix and then apply elementary matrix operations to bring it into a systematic form. The final matrix is, of course, not Vandermonde anymore.

We emphasize that, having an $(N_m + K) \times K$ code-matrix which has a $K \times K$ identity matrix as upper part and an $N_m \times K$ Vandermonde matrix as lower part (as we are proposing) would not (in general) allow us to fully decode having only $K$ values out of $d_k$'s and $s_{m,i}$'s. Such a matrix is not good for a RAID scheme as any disk, regular or redundant, can fail, and we need to recover the data using the remaining disks (whose number needs to be $\geq K$).

On the other hand, in our setting, we need instead to recover $K_m$ of $d_k$'s from $s_{m,i}$'s. This is to say that, the $K_m$ data nodes in the vicinity of the cluster "always fail" whereas the storage nodes in the cluster are "always alive." For this, our proposed code-matrix allows us to decode the data of the $K_m$ data nodes, by querying only $K_m$ storage nodes (storing $s_{m,i}$'s).

## 3 Decoding and Sampling

The random sampling procedure is as follows. A data collector chooses a random subset of storage nodes and queries them. The storage nodes reply with their checksums. Then, the data collector is able to decode if it receives at least $K_m$ checksums from cluster $m$, for $1 \leq m \leq M$.

As we mentioned before, the data collector may have no clue of the clustering structure. As such, when a data collector retrieves information from a WSN, it randomly queries a set $S$ of sensors from the whole network. We call this set of sensors a *sample*. Let $S_m$ be the intersection of $S$ and $m$-th cluster. Therefore, there are $|S_m|$ selected storage nodes in the $m$-th cluster. Let $X_m = |S_m| - K_m$. Then, the data collector can fully decode if $|S_m| \geq K_m$ (or $X_m \geq 0$), for $m = 1, \dots, M$. Clearly, the smallest sample size is $K = K_1 + \dots + K_M$. We define $Pr(S)$ to be the probability that the user can decode using sample $S$. We can show that

**Theorem 1.**

$$Pr(S) = \frac{\sum_{X_1} \sum_{X_2} \cdots \sum_{X_M} \binom{N_1}{K_1+X_1} \binom{N_2}{K_2+X_2} \cdots \binom{N_M}{K_M+X_M}}{\binom{N}{|S|}}.$$

To illustrate, suppose that we have 3 clusters of 7 storage nodes each. Also suppose that $K_1 = K_2 = K_3 = 4$; and thus the smallest sample size is 12. If $|S|$ is 13, then the extra selected node can be in any cluster. There are three possible cases for the extra node. Therefore, $Pr(S) = \frac{\binom{7}{4+1}\binom{7}{4}\binom{7}{4}+\binom{7}{4}\binom{7}{4+1}\binom{7}{4}+\binom{7}{4}\binom{7}{4}\binom{7}{4+1}}{\binom{21}{13}}$ =0.379. Formally, we are solving the following problem.

Find minimum $|S|$

Subject to

$Pr(S) \geq 1 - \varepsilon$

$X_m, K_m, N_m \geq 0$, for $1 \leq m \leq M$

$X_1 + X_2 + \ldots + X_M = X$

$K_1 + K_2 + \ldots + K_M = K$

$N_1 + N_2 + \ldots + N_M = N$

$|S| = K + X$

$X \leq N - K$

We solve the above problem numerically by calculating $Pr(S)$ of different sample sizes ranging from $K$ to $N$. This approach is inefficient since the calculation of $Pr(S)$ requires at least $O(\binom{M+X-1}{X})$ time, and the initial value of $|S|$ may be far from the optimum solution. The rest of this section approximates the $Pr(S)$ in linear time and provides a "safe" sample size that can be used directly by data collectors.

For the approximation, suppose that there are $N$ balls in $M$ bins. The balls are numbered from 1 to $N$, and the bins are numbered from 1 to $M$. The number of balls in the bin $m$ is $N_m$. Besides, each bin has a threshold value $K_m$. We randomly pick up a ball, record the ball number and bin number, put it back, and pick up another ball. We consider the following question.

> After we pick $|S|$ times, what is the probability that we pick at least $K_m$ balls from bin $m$, for $m = 1, \ldots M$?

We denote this probability as $Pr'(S)$ and use $Pr'(S)$ to approximate $Pr(S)$. Next we show that $Pr'(S)$ is easy to calculate and provides a good lower bound for $Pr(S)$.

We firstly consider bin 1. Define an indicator random variable

$$Y_i = \begin{cases} 1 \text{ if the } i^{th} \text{ ball is picked from bin 1} \\ 0 \text{ otherwise} \end{cases}$$

Let random variable $Z_1 = \sum Y_i$ be the number of the ball that we pick from bin 1. And then, we denote $E_1$ as the event that $Z_1 \geq K_1$, and $\overline{E}_1$ as the event that $Z_1 < K_1$. Clearly, $Pr'(S) = 1 - Pr(\overline{E}_1 \cup \overline{E}_2 \cup \ldots \cup \overline{E}_M)$. We use the following lemma from [2] to analyze $Pr(\overline{E}_1)$.

**Lemma 1.** *Consider a sequence of n Bernoulli trials, where in the $i^{th}$ trial, success occurs with probability $p_i$ and failure with probability $q_i = 1 - p_i$. Let Z be the random variable describing the total number of successes, and $\mu = E[Z]$. Then, for $r > 0$, $Pr(\mu - Z \geq r) \leq exp[\frac{-r^2}{2n}]$.*

Let $p_1 = N_1/N$ be the probability that we pick a ball from bin 1, and let $r$ be $|S|p_1 - K_1$. Then, $Pr(\mu - Z_1 \geq r) = Pr(Z_1 \leq K_1) = Pr(\overline{E}_1) \leq exp[\frac{-(|S|p_1 - K_1)^2}{2|S|}]$. If we take the union bound for all bins, we can find the probability that we fail to decode. We denote it as $Pr(\text{fail})$. $Pr(\text{fail}) = 1 - Pr'(S) \leq \sum_{m=1}^{M} Pr(\overline{E}_m) = \sum_{m=1}^{M} exp[\frac{-(|S|p_m - K_m)^2}{2|S|}]$. Assume that the $\widetilde{m}$-th bin has the maximum $exp[\frac{-(|S|p_m - K_m)^2}{2|S|}]$ value among all bins. Let $\widetilde{p}$ and $\widetilde{K}$ be the probability that we pick a ball from the $\widetilde{m}$-th bin and the threshold value of the $\widetilde{m}$-th bin respectively. We substitute $exp[\frac{-(|S|\widetilde{p} - \widetilde{K})^2}{2|S|}]$ for each $exp[\frac{-(|S|p_m - K_m)^2}{2|S|}]$ value in $Pr(\text{fail})$. Let the threshold be $\varepsilon$. Then, we have

$$Pr(\text{fail}) \leq M \cdot exp[\frac{-(|S|\widetilde{p} - \widetilde{K})^2}{2|S|}] < \varepsilon$$

In our sampling scheme, we sample the network without replacement. In other words, our scheme has better probability of success than the above ball and bin game. Finally, we solve the above equation and conclude with the following theorems.

**Theorem 2.** *The decoding probability $Pr(S)$ is no less than*

$$1 - \sum_{m=1}^{M} exp[\frac{-(|S|p_m - K_m)^2}{2|S|}].$$

**Theorem 3.** *We can choose a sample size*

$$|S| = \frac{2(\widetilde{p}\widetilde{K} + \ln\frac{M}{\varepsilon}) + \sqrt{4(\widetilde{p}\widetilde{K} + \ln\frac{M}{\varepsilon})^2 - 4\widetilde{p}^2\widetilde{K}^2}}{2\widetilde{p}^2}$$

*to achieve more than $1 - \varepsilon$ probability of decoding.*

## 4 Performance Evaluation and Conclusion

We use a grid network of $N$ storage nodes as an example. The same analytic principle is applicable to other network topology. Suppose that the storage nodes are deployed as a grid in a unit square. Each data node moves around randomly and uploads its sensor reading to its nearest storage node. A clustering algorithm partitions the grid into $M$ small squares. Regarding the data distribution, our coding scheme has a message complexity of $\sum_{m=1}^{M} K_m N_m (2/3)\sqrt{N_m}$, whereas the DEC has a message complexity of $5N \ln K \cdot (2/3)\sqrt{N}$. For an illustration,

if $N_m \approx N/M$ and $K_m \approx K/M$ (for $1 \leq m \leq M$), then the condition for our scheme to be better than DEC is that

$$M\frac{K}{M}\frac{N}{M}\sqrt{\frac{N}{M}} \leq 5N \ln K\sqrt{N} \Rightarrow \left(\frac{K}{5\ln K}\right)^{\frac{2}{3}} \leq M,$$

which is generally true. The experiments also confirm the same conclusion. We use 10,000 storage nodes and 100 data nodes in our simulation. The results show that our decoding probability is sufficiently high, and our data distribution cost is smaller than DEC's cost when $M > 4$. Detailed results and complexity analysis are in the full version of this paper at *http://web.uvic.ca/∼yhl/ssdbm08full.pdf*.

In conclusion, we have proposed a cost-efficient coding scheme that fulfills the ubiquitous access to sensor data. The algorithm is easy to implement on any clustered WSN. Moreover, we give a mathematical model for the probability of decoding as well as analysis of our system cost. The experimental results demonstrate that our coding scheme outperforms DEC.

## References


1. P. Bonnet, J. E. Gehrke, and P. Seshadri: Towards Sensor Database Systems, Proceedings of the Second International Conference on Mobile Data Management, 2001.
2. T. Cormen, C. Leiserson, R. Rivest, and C. Stein: *Introduction to Algorithms*, second edition, 1990.
3. A. G. Dimakis, V. Prabhakaran, and K. Ramchandran: Decentralized erasure codes for distributed networked storage, IEEE/ACM Transactions on Networking, Volume 14 , Issue SI, June 2006.
4. H. Garcia-Molina, J. D. Ullman, J. D. Widom: *Database Systems: The Complete Book*, 2001.
5. C. Intanagonwiwat, R. Govindan and D. Estrin: Directed diffusion: A scalable and robust communication paradigm for sensor networks, Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCOM), 2000.
6. P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein: Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet, Tenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X), 2002
7. S. Madden, M. Franklin, J. Hellerstein, and W. Hong: TAG: a tiny aggregation service for ad-hoc sensor networks, Proceedings of OSDI, 2002.
8. S. Madden, M. Franklin, J. Hellerstein, and W. Hong: The design of an acquisitional query processor for sensor networks, Proceedings of ACM SIGMOD, 2003.
9. A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson: Wireless Sensor Networks for Habitat Monitoring, Proceedings of ACM WSNA02, 2002.
10. J. S. Plank: A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems, *Software – Practice & Experience*, 27(9), pp. 995–1012, 1997.
11. X. Shan and J. Tan: Mobile sensor deployment for a dynamic cluster-based target tracking sensor network, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems,2005.
12. D. Wang, Q. Zhang, and J. Liu: Partial Network Coding: Theory and Application for Continuous Sensor Data Collection Proceedings of IWQOS 2006.