Energy-Reliability Tradeoffs in Sensor Network Storage

Neerja Bhatnagar Kevin M. Greenan Rosie Wacha Ethan L. Miller Darrell D. E. Long
Storage Systems Research Center, University of California, Santa Cruz, CA 95064, USA

Abstract

Sensor nodes that store their data locally are increasingly being deployed in hostile and remote environments such as active volcanoes and battlefields. Observations gathered in these environments are often irreplaceable, and must be protected from loss due to node failures. Nodes may fail individually due to power depletion or hardware/software problems, or they may suffer correlated failures from localized destructive events such as fire or rockfall. While many file systems can guard against these events, they do not consider energy usage in their approach to redundancy. We examine tradeoffs between energy and reliability in three contexts: choice of redundancy technique, choice of redundancy nodes, and frequency of verifying correctness of remotely-stored data. By matching the choice of reliability techniques to the failure characteristics of sensor networks in hostile and inaccessible environments, we can build systems that use less energy while providing higher system reliability.

Categories and Subject Descriptors

D.4.5 [**Reliability**]: Backup Procedures, Fault-tolerance—*Distributed File Systems*; C.4 [**Performance of Systems**]: Fault tolerance

General Terms

energy-reliability tradeoffs

Keywords

energy, reliability, sensor network storage

1 Introduction

The availability of inexpensive gigabyte-scale local storage on sensor nodes [13] and the high cost of radio operations relative to storage operations are enabling sensor nodes that store data locally in between data collection events [12]. Storage-based sensor networks are used to monitor volcanoes, battlefields, habitats, seismic events, traffic, and the stability and integrity of engineered structures such as buildings and bridges [2, 20]. However, the difficulty of gathering data from sensor nodes in hostile and inaccessible environments has also made it harder to deploy base stations that accumulate nodes' data. Base stations installed with sensor networks are easily detected in contested land areas such as borders, and are an obvious target for network disruption. Base stations in inaccessible and natural environments are single points of failure because they may suffer from power outages or malfunction, causing data loss; in a volcano-based sensor network, "[f]ailures of the base station infrastructure were a significant source of network downtime" [20]. Some networks try to avoid this problem by deploying multiple base stations or specialized storage nodes [19], increasing the both the likelihood of the detection of the network, and the system cost. Data loss in centrally-controlled sensor networks is likely to be more severe because nodes do not retain the observations they have already uploaded to the base station. Moreover, a base station cannot easily transmit data to a receiver when none is nearby, as is often the case in remote environments. Such environments are better suited to occasional data collection, requiring nodes to reliably maintain their data over long periods of time.

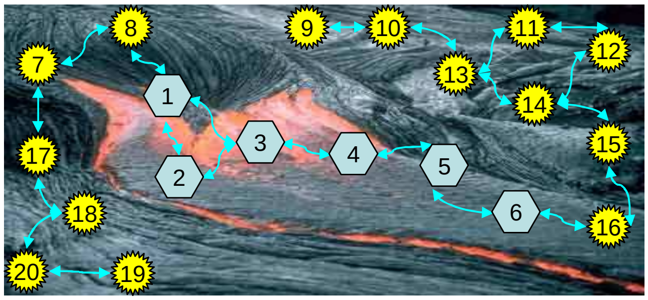

Figure 1. Sensor network on a volcanic lava flow. Nodes 1–6 have been destroyed by the flow.

Individual sensor nodes typically suffer from relatively high failure rates, as compared to traditional storage devices. Moreover, sensor nodes are more likely to suffer *correlated* failures due to environmental dangers. Individual failures may be caused by battery depletion, hardware or software errors, or physical damage. In contrast, correlated node failures may be caused by larger-scale physical damage caused by a destructive event such as flood, rockfall, or fire; for example, the lava flow in Figure 1 has obliterated nodes 1–6. Unfortunately, the latest data from destroyed nodes is the most valuable because it may record details of the event, making it even more important for the observations gathered by the nodes to survive their destruction. However, it is also imperative that sensor nodes create and maintain back-up copies of their data without overwhelming their energy budgets.

We discuss the tradeoffs between energy and reliability in sensor networks that store data for long periods of time: weeks to years. These tradeoffs can be made in three separate areas: redundancy techniques, choice of nodes which store the redundant data, and fre-

Hot EmNets'08 June 2–3, 2008, Charlotesville, Virginia, USA.

quency of integrity checks on the remotely stored redundant data. We do not expect the energy expenditure of reliable storage in sensor networks to be less than the energy expended by nodes to upload their data to a base station; rather, our goal is to make sensor network storage much more reliable by increasing the likelihood that sensor data survive despite individual and correlated node failures. By providing energy-efficient storage operations, sensor networks can more easily provide raw data, instead of aggregated and representative values, to their intended audience, potentially facilitating more robust forecast and analysis models.

We assume that the network is comprised of sensor nodes severely constrained in power, storage, and processing. We also assume that nodes have limited radio range, so communication with distant nodes requires multi-hop routing. Since our research is primarily concerned with energy-reliability tradeoffs, we fold the costs for interference and retransmission into the cost for transmitting data between nodes. We assume that each node has a battery-backed RAM for buffering data and NAND flash memory for persistent storage [12], though new non-volatile memory technologies such as phase change memories may further simplify the architecture [9].

2 Issues in Reliability

Analyzing tradeoffs between energy usage and file system reliability depends on making good choices for redundancy techniques, nodes for remote storage, and frequency of checking integrity of redundant data, while considering the high failure rate of sensor nodes and the likelihood of occurrence of correlated failures [14].

2.1 Redundancy Techniques

As with traditional file systems, sensor nodes may use either mirroring or erasure coding to store data reliably. Transmission costs dominate energy usage when mirroring is used because transmitting data costs two hundred times more energy [12] than storing the same amount of data locally. As a result, due to the relative position of nodes and the base station, the transmission cost of mirroring data to another node may be lower than that of uploading data to a base station. This is specifically the case when the transmitting node is in the center of the network and the base station is installed at the edge of the network, or vice versa. For example, in Figure 1, node 6 will have to transmit its data over five hops if the base station was installed near node 11. The storage overhead of mirroring is also very high: tolerating n failures requires the system to store $n+1$ copies of the data. In contrast, processing costs dominate energy usage for erasure codes.

We compared the performance (energy consumption expressed in mJ and throughput expressed in MB/s) of encoding using Reed-Solomon (RS) codes [15] based on GF(2^8) [5] to XOR-based codes [6, 21] on an ARM9E 400 MHz processor that consumes 94 mJ/s [1]. The first column in Table 1 represents the RS code implementation for parameters (n,m), where n is the number of data nodes, and m is the number of parity nodes. RS codes were implemented as table lookups, where each multiplication requires two lookups. Each lookup table is 256 bytes in size, consuming 512 bytes of memory. The second column in Table 1 represents the most fault-tolerant XOR-based codes for the same parameters. These codes have the storage efficiency of $n/(n+m)$. The last two rows present the performance of highly fault-tolerate XOR-based codes that we developed. The XOR_1 code we designed is an instance of a WEAVER code [6] that tolerates two-node failures.

Reed-Solomon codes consume 3–10 times more energy than XOR-based codes due to more complex finite field calculations [6], but provide higher reliability (*e.g.*, a $(5,3)$ RS code can tolerate *all* three-node failures but an XOR-based $(5,3)$ code may only be able to tolerate at *most* three-node failures). However, it may be possible to tolerate some node failures without losing data because very closely-located sensor nodes may be observing similar phenomena. In order to tolerate correlated failures, closely-located sensor nodes must spread their information over a large physical area. The energy expenditure of XOR_1 and XOR_2 schemes is comparable to most XOR-based codes but better than that for RS codes. We are currently exploring the suitability of several less processor intensive XOR-based codes, based on the research done by Wylie and Swaminathan [21], to sensor networks.

Table 1. Energy Expenditure of Erasure Codes in mJ/s and Throughput in MB/s.

Code Size	Energy Expenditure (mJ)		Throughput (MB/s)	
	RS	**XOR**	**RS**	**XOR**
(5, 3)	3.515	1.205	2.674	7.798
(6, 2)	3.133	0.6	3	15.654
(9, 3)	4.82	0.524	1.95	17.953
(10, 2)	3.92	0.653	2.4	14.4
(17, 3)	5.193	0.588	1.81	15.99
(18, 2)	4.36	0.589	2.156	15.972
XOR_1	—	0.74	—	12.76
XOR_1	—	0.75	—	12.72

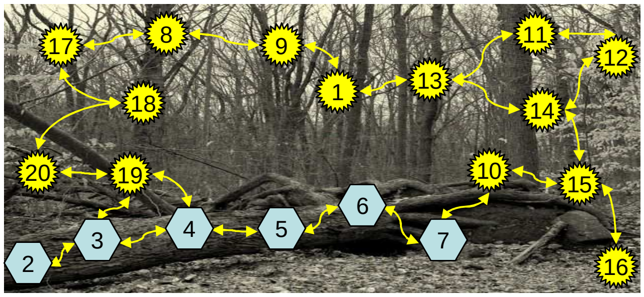

Figure 2. Node 2 replicates its data on nodes 3–7 but is more likely to suffer data loss even from a small destructive event.

2.2 Node Choice

The impact of correlated failures caused by localized damage can be mitigated by spreading redundant data over a large physical area. There is a cost in energy to send the data further away. For example, observations from node 2 in Figure 2 will be lost, despite replicating them on nodes 3–7, if a tree rooted near node 2 falls. Data from node 2 is more likely to survive if node 2 sends its data to nodes $\langle 3,8,12,16,20\rangle$ for redundant storage, as shown in Figure 3. Nodes $\langle 8,12,16,20\rangle$ are well-spread out; and so are less likely to fail simultaneously. Even though the number of nodes in node 2's redundancy group is the same in both examples, the latter scheme is more reliable but also more expensive both for node 2 and its neighboring nodes because multi-hop transmissions consume more energy.

Mirroring alone is energy-consuming for making sensor network storage reliable. In order to reduce energy expenditure, it may be better to mirror data only to nearby nodes and to use erasure codes for nodes that are further away. This approach can quickly replicate data nearby, guarding against individual node failure, and can use widespread replication to protect against correlated node failures. For example, in Figure 3, node 2 can mirror data to node 3 to safeguard against its own failure, but use erasure codes with nodes $\langle 8,12,16,20\rangle$ to safeguard against correlated failures. Systems such as OceanStore [16] use erasure codes to tolerate relatively large numbers of failed nodes; we plan to do the same for making sensor network storage reliable. Our file system has the

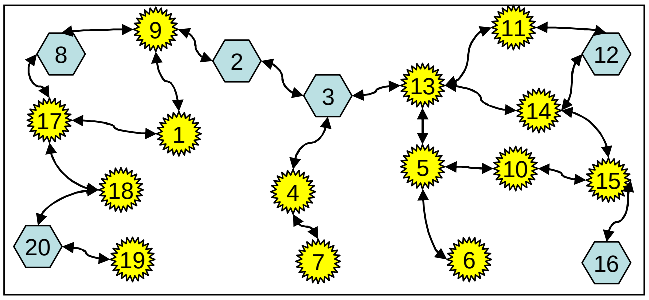

Figure 3. Node 2 replicates its data on nodes $\langle 3,8,12,16,20\rangle$ to increase its likelihood of surviving large destructive events.

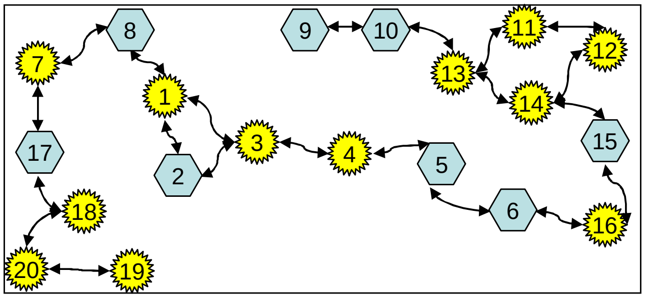

Figure 4. Nodes $\langle 2,5,6,8,9,10,15,17\rangle$ form an 8-node redundancy group such that the network can tolerate the failure of 3-node combinations such as $\langle 5,10,17\rangle$ and $\langle 2,6,10\rangle$.

advantage of using less-expensive XOR-based codes in place of RS codes by carefully placing redundant data on particular nodes. When using a $(5,3)$ XOR-based code, by arranging data so that the "fatal" three-node sets cover a large physical area, the sensor network can gain nearly all benefits of RS codes with the computational cost of XOR-based codes. For example, node 2 in Figure 4 might choose nodes $\langle 2,5,6,8,9,10,15,17\rangle$ in its eight-node redundancy group. If only three-node combinations, such as $\langle 5,10,17\rangle$ and $\langle 2,6,9\rangle$, caused data loss, then the system would be relatively safe since these node-sets cover a widespread area, and therefore, are less likely to suffer correlated failures simultaneously. The system could provide additional reliability by choosing some very distant nodes as part of its redundancy group, perhaps replacing node 5 with node 19 and node 6 with node 12.

We use a simple Markov model to analyze the availability of the Mirror$_4$, XOR$_1$, and XOR$_2$ schemes. Figure 5, depicts 4-way mirroring, but can easily be generalized to an *n*-node redundancy group. The transitions are exponentially distributed with mean failure rate λ, and mean repair rate μ. For simplicity, we let $\rho = \lambda/\mu$. State $(0,0)$ represents the failed state.

In the XOR$_1$ scheme, each node stores its own data and the XOR of data from two other nodes. For example, node A stores its own data and $B \oplus C$; node B stores its own data and $C \oplus D$; node C stores its own data and $D \oplus E$; node D stores its own data and $A \oplus E$; and node E stores its own data and $A \oplus B$. In the XOR$_2$ scheme, each node stores its own data and data from four other nodes as two-node XORs. For example, node A stores its own data and $B \oplus C$ and $D \oplus E$; node B stores its own data and $D \oplus E$ and $A \oplus C$; node C stores its own data and $A \oplus E$ and $B \oplus D$; node D stores its own data and $A \oplus C$ and $B \oplus E$; and node E stores its own data and $A \oplus B$ and $C \oplus D$. The storage overhead of Mirror$_4$ is four times that of the original data set. The storage overhead of XOR$_1$ and XOR$_2$ schemes is, respectively, two and three times the original data set. Figure 6 shows that XOR$_2$ delivers availability similar to Mirror$_4$, but at a lower overhead. Mirror$_4$ can tolerate at most four node failures, while XOR$_1$ and XOR$_2$ schemes can, respectively, tolerate at most two- and three-node failures. Markov models provide good approximate analysis, but do not work well for "irregular" XOR codes or for systems that experience correlated failures; these are better suited to simulation.

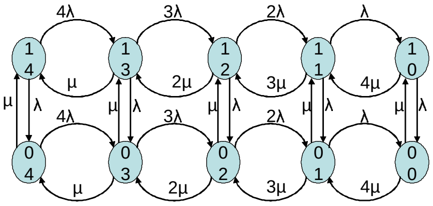

Figure 5. Markov model of a 5-node redundancy group depicting Mirror$_4$.

Table 2. MTTDL, in hours, for Mirror$_4$, XOR$_1$ and XOR$_2$ schemes with and without repair.

	Mirror$_4$	**XOR$_1$**	**XOR$_2$**
MTTDL with repair	4.87×10^{11}	2.42×10^{6}	6.50×10^{8}
MTTDL w/o repair	4932	1692	2772

The availability of a node's data when Mirror$_4$, XOR$_1$, and XOR$_2$ schemes are used to create redundancy in the sensor network are given by:

$$\text{Mirror}_4 = 1 - \frac{\rho^5}{(1+\rho)^5},$$

$$\text{XOR}_1 = \frac{10\rho^2 + 5\rho + 1}{(\rho+1)^5}, \quad \text{and}$$

$$\text{XOR}_2 = 1 - \frac{5\rho + 1}{(\rho+1)^5}.$$

These availability models are simple and assume that the nodes may be repaired. In the case of no repair, steady-state does not exist and so the system must be modeled using differential equations. These equations quickly become unmanagable, and so a better solution is to use simulation, which has the additional advantage of being able to model correlated failures.

Modeling mean-time-to-data-loss (MTTDL) is easier, and uses the same transition matrix that would be used for modeling with differential equations. We assume that both failures and repairs are exponentially distributed. We solve all these models by building a transition matrix M, as discussed by Schwarz [17], and computing

$$MTTDL = -[1,1,1,\ldots,1] \cdot M^{-1} \cdot [1,0,0,\ldots,0].$$

Table 2 presents the MTTDL for Mirror$_4$, XOR$_1$, and XOR$_2$ schemes, with and without repairs. For this example we assume that nodes are organized into five-node redundancy groups and choose $\rho = 5.56 \times 10^{-3}$, which assumes that failures occur on average every 3 months and nodes are repaired, on average, in 12 hours.

2.3 Frequency of Integrity Checks

Regardless of the technique used to generate redundancy, each sensor node must periodically check to ensure that its back-up data is still being stored correctly. If a node replicates its data to distant nodes, then its integrity checks and their responses must also travel further, thereby expending more energy. Moreover, the more frequently a node checks the correctness of its back-ups, the more energy it expends. Furthermore, additional energy is expended at the responding node which must generate a signature and transmit

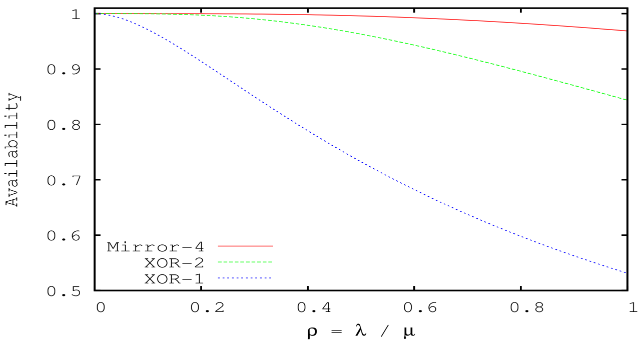

Figure 6. Data availability.

it back over multiple hops. However, in a system where node failure is frequent, it is necessary to detect small problems before they grow bigger and cause data loss. It may be energy-wise to allow small problems to become a little bigger, but not fatal, because the energy cost to restore redundancy is sub-linear. We are currently exploring the energy tradeoffs between more frequent integrity checks with that of the overall reliability of the system.

We plan to use algebraic signatures [18] to verify the correctness of remotely-stored redundant data. Although algebraic signatures are not cryptographically secure, they change in response to small changes in the data from which they are generated. Moreover, they can be used in conjunction with XOR or RS codes to ensure that a set of returned signatures is consistent. An algebraic signature operation requires a node to calculate a function on its own piece of stored redundant data, thereby, generating a small (4–8 byte) signature. When combined, these signatures obey the same relationship as the data from which they were generated; if the signatures form a valid code word in the XOR or RS scheme, the underlying data is highly likely to be consistent as well—the chance of agreement with an underlying error is approximately 2^{-b} for a b-bit signature.

3 Experimental Approach

We have developed a cost model to compute the total energy expenditure of making sensor network storage reliable. This total energy expenditure is comprised of I/O, processing, and radio costs, and includes the energy expended at the originating node as well as at each node that stores redundancy data. We also evaluate the storage overhead of mirroring, erasure coding, and of corresponding metadata. Our evaluation assumes that the energy expended per-byte to read/write data from SRAM and flash is the same. The cost of radio transmission is calculated by multiplying the number of bytes transmitted with the per-byte per-hop energy expenditure. Radio reception cost is calculated by multiplying the number of bytes received by the per-byte energy expenditure for reception. Most sensor nodes follow a “write-once, read-never, modify-never” access policy, therefore, nodes do not need to perform incremental back-ups: a file once written will not be modified during the nodes’ deployment. Each node maintains metadata such as the originating node’s ID, the chunk ID, and the receiving node’s ID. Although it may be sufficient to store this metadata on either the originating node or on the receiving node, storing it on the originating node as well as each corresponding back-up node will help back-up the metadata, and prevent it from being a single point of failure.

An originating node, ready to back-up its data, sends a “hello” message, as shown in Figure 7, to n nodes to check if they have space to store its data. Back-up nodes that have sufficient storage respond with an “ack” message. The originating node spends energy in sending n “hello” messages and receiving m “ack” messages, where $m \leq n$. Each back-up node spends energy receiving a “hello” message and sending an “ack” message. Each originating node uses time stamps and message sequence numbers to keep track of what data has already been received and backed-up correctly by the back-up nodes. By doing this, if the connection between two communicating nodes is lost, the transmitting node can avoid unnecessary retransmissions. An originating node spends energy in transmitting data to each back-up node, while each back-up node spends energy in receiving and storing back-up data in its flash memory.

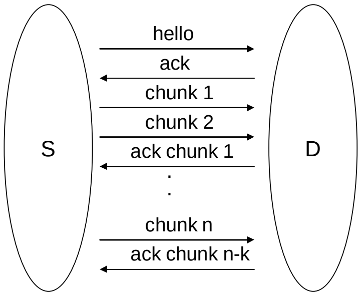

Figure 7. An originating node, *S*, uses this protocol to replicate its data to another node, *D*.

4 Optimizations

We are currently researching several optimizations that can help reduce energy requirements for making sensor network storage reliable. For example, it may be possible to piggy-back integrity check messages and responses on other network traffic such as “hello” or “ack” messages or on other traffic related to updating routing and neighborhood tables. Such piggy-backing has the potential of reducing transmission cost because integrity check messages are relatively small and the marginal cost of including additional information in another message is minimal. In order to reduce energy expenditure of reliability, some redundancy can be generated at remote nodes to reduce the total volume of data that must be transmitted over large distances. Sending all data to a remote node and letting it distribute it to its nearby neighbors may also be more energy efficient than the originating node distributing its data to all nodes. For example, in Figure 3, node 2 can transmit its data to node 10, which can distribute the data to nodes 5, 6, and 15. Energy expended in transmission can be further reduced by using some of the “routing” nodes or the intermediate nodes in the path between a source node and its destination back-up node.

5 Related Work

Koushanfar, *et al.* [8] identify computing, storage, communication, sensing, and actuating as resources and propose backing-up a resource running low with one that is abundantly available. However, the application software that computes resource availability may itself consume lots of energy. The solutions presented by Kamra, *et al.* [7] and Lin, *et al.* [10] are designed for sink-based network architectures. Although our solution is applicable to both distributed and centrally-controlled networks, we assume a distributed network architecture without a sink. Lin, *et al.* [11] use decentralized fountain codes to introduce redundancy into the network. Ghose, *et al.* [4] present a Resilient Data Centric Storage (R-DCS) scheme to reduce energy consumption while increasing resilience to node failures. Schemes presented by authors [4, 11] require a complete picture of the network. This may not always be possible with *ad hoc* networks [10]. In contrast, we assume nearly homogeneous nodes with no single point of failure. This assumption may not hold well in *ad hoc* networks deployed by dropping nodes from an airplane or artillery shell. Dimakis, *et al.* [3] use decentralized erasure codes to reduce latency and unreliability between query times and the time at which data reaches the data collector. The authors assume a fixed ratio between the number of storage nodes and the number of nodes that contain original data.

6 Conclusion

"Sense and store" sensor networks are gaining popularity due to the recent availability of gigabyte-scale local storage on sensor nodes, and because storage operations are more energy efficient than radio operations. It is important to make the data stored locally on sensor nodes reliable because sensor nodes suffer from unusually high failure rates (both individual and correlated). We discussed three factors that influence energy-reliability tradeoffs—redundancy techniques, node choice, and frequency of integrity checks. We presented a simple analytical model for modeling the availability of a node's data, and are currently exploring these issues in more detail using simulation-based models. Our research on energy-reliability tradeoffs will enable long-term reliable storage in sensor nodes and enable their deployment in environments where frequent data collection is infeasible.

7 Acknowledgments

This work was supported in part by the Department of Education GANN program, the Department of Energy under award DE-FC02-06ER25768, the Institute for Scalable Scientific Data Management in cooperation with Los Alamos National Laboratory and industrial sponsors of the Storage Systems Research Center including Agámi Systems, Data Domain, Hewlett Packard, LSI Logic, NetApp, Seagate, and Symantec.

8 References

[1] www.arm.com/products/CPUs/ARM926EJ-S.html.

[2] CHONG, C.-Y., AND KUMAR, S. P. Sensor Networks: Evolution, Opportunities, and Challenges. In *Proc. of the IEEE* (2003), vol. 91, pp. 1247– 1256.

[3] DIMAKIS, A. G., PRABHAKARAN, V., AND RAMCHANDRAN, K. Ubiquitous access to distributed data in large-scale sensor networks through decentralized erasure codes. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks* (2005), IEEE Press, p. 15.

[4] GHOSE, A., GROSSKLAGS, J., AND CHUANG, J. Resilient data-centric storage in wireless ad-hoc sensor networks. In *Proceedings of the 4th International Conference on Mobile Data Management* (2003), Springer-Verlag, pp. 45–62.

[5] GREENAN, K., MILLER, E. L., AND SCHWARZ, T. Analysis and construction of Galois fields for efficient storage reliability. Tech. Rep. Technical Report UCSC-SSRC-07-09, University of California, Santa Cruz, 2007.

[6] HAFNER, J. L. WEAVER codes: Highly fault tolerant erasure codes for storage systems. In *Proceedings of the Second USENIX Conf. on File and Storage Technologies (FAST)* (2005).

[7] KAMRA, A., FELDMAN, J., MISRA, V., AND RUBENSTEIN, D. Data persistence for zero-configuration sensor networks. In *ACM Special Interest Group on Data Communications (SIGCOMM)* (2006).

[8] KOUSHANFAR, F., POTKONJAK, M., AND SANGIOVANNI-VINCENTELLI, A. Fault tolerance techniques in wireless adhoc sensor networks. In *Proc. of IEEE Sensors* (2002), vol. 2, pp. 1491– 1496.

[9] LAI, S. Current status of the phase change memory and its future. *IEDM Technical Digest* (2003), 10.1.1– 10.1.4.

[10] LIN, S., ARAI, B., AND GUNOPULOS, D. Reliable hierarchical data storage in sensor networks. In *19th Int'l Conf. on Scientific and Statistical Database Mgmt, SSDBM* (2007), pp. 26–35.

[11] LIN, Y., LIANG, B., AND LI, B. Data persistence in large-scale sensor networks with decentralized fountain codes. In *INFOCOM 2007. 26th IEEE Int'l Conf. on Computer Communications* (2007), pp. 1658–1666.

[12] MATHUR, G., DESNOYERS, P., GANESAN, D., AND SHENOY, P. Ultra-low power data storage for sensor networks. In *IPSN '06: Proceedings of the fifth international conference on Information processing in sensor networks* (2006), ACM, pp. 374–381.

[13] MITRA, A., BANERJEE, A., NAJJAR, W., ZEINALIPOUR-YAZTI, D., KALOGERAKI, V., AND GUNOPULOS, D. High-Performance Low Power Sensor Platforms Featuring Gigabyte Scale Storage. In *IEEE/ACM 3rd Int'l Workshop on Measurement, Modelling, and Perf. Anal. of WSNs* (2005).

[14] NATH, S., YU, H., GIBBONS, P. B., AND SESHAN, S. Subtleties in tolerating correlated failures in wide-area storage systems. In *Networked Systems Design and Implementation (NSDI)* (2006).

[15] PLANK, J. S. A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems. *Software, Practice and Experience 27*, 9 (1997), 995–1012.

[16] RHEA, S., EATON, P., GEELS, D., WEATHERSPOON, H., ZHAO, B., AND KUBIATOWICZ, J. Pond: the OceanStore prototype. In *Proceedings of the Second USENIX Conf. on File and Storage Technologies (FAST)* (2003), pp. 1–14.

[17] SCHWARZ, T. *Reliability and Performance of RAID Systems*. PhD thesis, Univ. of California at San Diego, 1994.

[18] SCHWARZ, T. S. J., AND MILLER, E. L. Store, forget, and check: Using algebraic signatures to check remotely administered storage. In *ICDCS '06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems* (2006), IEEE Computer Society, p. 12.

[19] SHENG, B., LI, Q., AND MAO, W. Data storage placement in sensor networks. In *ACM International Symposium on Mobile Ad Hoc Networking and Computing* (2006), pp. 344–355.

[20] WERNER-ALLEN, G., LORINCZ, K., JOHNSON, J., LEES, J., AND WELSH, M. Fidelity and yield in a volcano monitoring sensor network. In *OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation* (2006), USENIX Association, pp. 381–396.

[21] WYLIE, J. J., AND SWAMINATHAN, R. Determining fault tolerance of XOR-based erasure codes efficiently. In *Dependable Systems and Networks (DSN)* (2007), pp. 206–215.