Collaborative storage management in sensor networks

Sameer Tilak* and Nael B. Abu-Ghazaleh

Computer System Research Laboratory,
Department of CS, Binghamton University, 13902 6000 Binghamton, NY
E-mail: sameer@cs.binghamton.edu E-mail: nael@cs.binghamton.edu
*Corresponding author

Wendi Heinzelman

Department of Electrical and Computer Engineering,
University of Rochester, 14627 0126 Rochester, NY
E-mail: wheinzel@ece.rochester.edu

Abstract: In this paper, we consider a class of sensor networks where the data is not required in real-time by an observer; for example: a sensor network monitoring a scientific phenomenon for later play back and analysis. In such networks, the data must be stored in the network. Thus, in addition to battery power, storage is a primary resource; the useful lifetime of the network is constrained by its ability to store the generated data samples. We explore the use of collaborative storage techniques to efficiently manage data in storage constrained sensor networks. The proposed collaborative storage technique takes advantage of spatial correlation among the data collected by nearby sensors to significantly reduce the size of the data near the data sources. In addition, local coordination can be used to adjust the sampling rate to match the required application fidelity. We show that the proposed approach provides significant savings in the size of the stored data vs. local buffering. These savings allow the network to operate for a longer time without exhausting storage space. Furthermore, the savings reduce the amount of data that will eventually be relayed in response to queries or upon eventual collection of the data. In addition, collaborative storage performs load balancing of the available storage space if data generation rates are not uniform across sensors (as would be the case in an event driven sensor network), or if the available storage varies across the network.

Keywords: sensor networks; collaborative storage management.

Reference to this paper should be made as follows: Tilak, S., Abu-Ghazaleh, N.B. and Heinzelman, W. (2005) 'Collaborative storage management in sensor networks', *Int. J. Ad Hoc and Ubiquitous Computing*, Vol. 1, Nos. 1/2, pp.47–58.

Biographical notes: Sameer Tilak is a PhD student in the Computer Science department at Binghamton University. He received his Bachelor of Engineering degree in Computer Engineering from Pune Institute of Computer Technology, Pune University in 1999. He received his MS in Computer Science from the University of Rochester in 2003. His research interests include Wireless Networks (specifically ad-hoc and sensor networks), Grid Computing, Storage/File Systems, and Parallel Discrete-event Simulation.

Nael Abu-Ghazaleh is an Associate Professor in the Computer Science Department at Binghamton University. He received his BSc in Electrical Engineering from the University of Jordan in 1990, and his MS and PhD degrees in Computer Engineering from the University of Cincinnati in 1994 and 1997 respectively. His research interests are in Mobile and Ad hoc Networks, Sensor Networks and High Performance Computing.

Wendi B. Heinzelman is an Assistant Professor in the Department of Electrical and Computer Engineering at the University of Rochester. She received a BS degree in Electrical Engineering from Cornell University in 1995 and MS and PhD degrees in Electrical Engineering and Computer Science from MIT in 1997 and 2000, respectively. Her current research interests lie in the areas of Wireless Communications and Networking, Mobile Computing, and Multimedia Communication. Dr. Heinzelman received the NSF CAREER award in 2005 for her research on Cross-layer Architectures for Wireless Sensor Networks. She is a Member of Sigma Xi, the IEEE, and the ACM.

1 Introduction

In this paper, we consider a class of sensor networks where the information collected by the sensors is not collected in real-time. In such applications, the data must be stored, at least temporarily, within the network and used in response to dynamic queries until it is later collected by an observer, or until it ceases to be useful. For example, data may be collected for a scientific application in a sensor field. Scientists appear occasionally (e.g., every week) to check on the experiment and collect the data. Furthermore, some applications have sensors which collect data that may be needed by users of the networks that generate queries dynamically. In such applications, the data must be stored in the network: storage is a primary resource, which in addition to energy, determines the useful lifetime of the network. This is the problem considered in this paper: how to use limited persistent storage of a sensor to store sampled data effectively.

One basic storage management approach is to buffer the data locally at the sensors that collect them. However, such an approach does not allow the neighbours to collaborate to reduce the size of their overall data. Two approaches for reducing the data size are possible if neighbours collaborate:

- *Aggregation.* They can capitalise on the spatial correlation of data among neighbouring sensors to reduce the overall size of the stored data. Aggregation is well known in the context of traditional real-time monitoring sensors (Boulis et al., 2003; Intanagonwiwat et al., 2001). However, there are important differences in this case: aggregation relies on the presence of data from multiple sensors at intermediate nodes as it is being sent in real-time towards a sink. With local buffering this is not possible without forcing data exchange among sensors. There are other important differences from traditional aggregation such as the long term availability of the data and the ability to revisit aggregation decisions since storage is reassignable.
- *Coordination for redundancy control.* More specifically, with local buffering, the sensors may be collecting and storing redundant data. By coordinating infrequently, redundancy can be estimated, allowing the sensors to sample less aggressively and save storage and energy.

Collaborative storage management can provide the following advantages over a simple buffering technique:

- More efficient storage allows the network to continue storing data for a longer time without exhausting storage space.
- *Load balancing is possible.* If the rate of data generation is not uniform at the sensors (e.g., in the case where a localised event causes neighbouring sensors to collect data more aggressively), some sensors may run out of storage space while space remains available at others. In such a case, it is important for the sensors to collaborate to achieve load balancing for storage, and avoid or delay data loss due to insufficient local storage.
- Dynamic, localised reconfiguration of the network (such as adjusting sampling frequencies of sensors based on estimated data redundancy and current resources).

We describe a cluster based collaborative storage approach and compare it through simulations to a local buffering technique. Our experiments show that collaborative storage makes more efficient use of sensor storage and provides load balancing, especially if a high level of spatial correlation/redundancy among the data of neighbouring sensors is present. The tradeoff is that, using collaborative storage, data need to be communicated among neighbouring nodes, and thus collaborative storage expends more energy than local buffering in the data collection phase. However, since data is aggregated using collaborative storage, a smaller amount of data is stored and a smaller amount of data is eventually relayed to the observer, thereby reducing energy dissipation in this phase of operation.

The remainder of this paper is organised as follows. Section 2 overviews the partitioned sensor network problem and motivates collaborative storage in more detail in the context of this problem. Section 3 discusses the design goals of a storage management scheme with an overview of important factors. Section 4 provides an overview of related work in this area. Section 5 presents the proposed storage management protocols and discusses the important design tradeoffs. In Section 6 we evaluate the storage alternatives under different scenarios. Finally Section 7 presents conclusion and our future research.

2 Motivation

In this section, we motivate the storage problem by describing applications that require ‘in network’ storage. We identify the following two classes of applications:

- *Offline monitoring.* The sensors are deployed to collect detailed information about a phenomenon for later playback and analysis. Eventual data collection (reachback) can be accomplished by an observer who moves around the sensor field or relayed back through multihop communication among the sensors themselves. In either case, the *data is stored continuously, but read only once.*
- ZebraNet (Juang et al., 2002) is an example of such a network: it is a sensor network for wildlife tracking, whose goal is to provide more insight into complex issues such as migration patterns, social structures and mobility models of various animal species. In this application, sensors are attached to animals. Scientists (aka observers) collect the data by driving around the monitored habitat, receiving information from Zebras as they come in range with them. Data collection is not pre planned: it might be unpredictable and infrequent.

The sensors do not have an estimate regarding the observer's schedule. The observer would like the network to maintain all the new data samples available since the last time the data was collected. Further, we would like the collection time to be small since the observer may not be in range with the zebra for a long time.

- *Dynamic augmented reality.* In this type of application, the sensors store data about ongoing events. The data is dynamically queried by users within the network. The queried data can be data about current and recent events or even historical data. For example, such a network deployed in a battlefield may be queried by soldiers to learn about nearby enemy units (current and recent data) or by commanders to learn about long term enemy movement. In such a network, collected data may be accessed multiple times (or not at all). Moreover, the importance of a piece of data may change with time.

Storage may also be used to tolerate temporary network partitioning, where the observer is not reachable from the partitioned sensors, without losing potentially valuable data. For example, the Remote Ecological Micro-Sensor Network focuses on remote visual surveillance of federally listed rare and endangered plants (http://www.botany.hawaii.edu/pods/overview.htm). This project aims to provide near real-time monitoring of important events such as visitation by pollinators and consumption by herbivores along with monitoring a number of weather conditions and events. Sensors are placed in different habitats, ranging from scattered low shrubs to dense tropical forests. Environmental conditions can be severe; e.g., some locations frequently freeze. In this application, network partitioning (relay nodes becoming unavailable) may occur due to the extreme physical conditions (e.g., deep freeze). Moreover, voluntary partitioning may occur if relay nodes operate in low duty cycles to conserve energy, to reduce interference with the observed phenomena. Important events that occur during disconnection periods should be recorded and reported once the connection is re-established. Effective storage management is needed to maximise the partitioning time that can be tolerated.

In the next section, we discuss various factors that affect the design of a storage management scheme and then describe the design goals of such a system.

3 Design goals

A major area of sensor network research focuses on making sensor network design and operation, energy efficient. However, for storage bound applications, there is an additional finite resource: the available storage at the sensors. Once the available storage space is exhausted, a sensor can no longer collect and store data locally; unless they delete older data, the sensors that have run out of storage space cease to be useful. Thus, the sensor network utility is bound by two resources: its available energy and its available storage space. Effective storage management protocols must balance these two resources to prolong the network's useful lifetime.

Energy and storage are fundamentally different resources. Specifically, storage is a reassignable resource, while energy is not. For example, a node may free up some storage space by deleting or compressing data. This is not possible in the case of energy, as the battery power spent in either transmitting or receiving data can not be reassigned to new data. Additionally, storage at other nodes may be utilised, at the cost of transmitting the data to them. The alternative to storing the data locally is transmitting the data towards an observer or a collection point.

In existing technology, storage devices consume significantly less energy than RF communication devices. Accordingly, the tradeoff between storage and energy is complex. Sensors may exchange their data with nearby sensors to take advantage of the spatial correlation in their data to reduce the overall data size. Another positive side effect is that the storage load can be balanced even if the data generation rates or the storage resources are not. However, the exchange of data among the sensors consumes more energy in the data collection phase; the energy cost of communicating the data far outweighs the energy savings in storage (due to the smaller data size). On the surface, it may appear that locally storing data is the most energy efficient solution. However, the extra energy spent in exchanging data may be counterbalanced by the energy saved by storing smaller amounts of data and more importantly, by the smaller energy expenditure when replying to queries or relaying the data back to observers.

We now present the design goals of a storage management protocol.

- *Storage efficiency*. Since the amount of storage available to a sensor is very limited, it is important to minimise the data that needs to be stored. Efficient use of storage space leads to better coverage, since a given sensor can continue to store data for a longer time period.
- *Storage load balancing.* If the available storage resources or the data generation rates are nonuniform, it is desirable that the storage be load balanced to avoid exhausting the storage at important sensors and losing their data.
- *Data coverage.* This is a measure of the fraction of the data samples that the network was able to collect and retain.
- *Energy efficiency.* Sensors are constrained by the limited battery power available to them. Any storage management scheme should be designed with the goal of energy efficiency in mind. Energy is spent in two phases:

- *During data collection*. This is the energy spent to store the data, in addition to any communication that occurs, to take advantage of collaborative storage
- *During data access.* This is the energy spent in relaying the data to one or more data sinks. In the offline monitoring model, data access occurs once. In the augmented reality model, it may be accessed any number of times.

4 Related work

Because of the wireless nature of sensors, the primary resource constraint is the limited battery energy. Energy awareness permeates all aspects of sensor design and operation, from the physical design of the sensor (Asada et al., 1998; Chandrakasan et al., 1999) to the design of its operating system (Hill et al., 2000), communication protocols and applications (Yao and Gehrke, 2002). In this section, we briefly overview some of the issues involved in storage management; for a more detailed review of these issues, please refer to our survey on this topic (Tilak et al., 2005).

Ratnasamy et al. (2002) proposed using Data Centric Storage (DCS) to store data by name within a sensor network such that all related data is stored at the same (or nearby) sensor nodes using geographic hashing. GHT is a structured approach to sensor network storage that makes it possible to index data based on content, without requiring query flooding. GHT also provides load balancing of storage usage (assuming fairly uniform sensor deployment). GHT implements a Distributed Hash Table by hashing a key k into geographic coordinates. Thus, queries for data of a certain type are likely to be satisfied by a small number of nodes, significantly improving the performance of queries. However, this enhanced query performance requires moving related data from its point of generation to its appropriate keeper, as determined by geographic hashing. We view this work as a higher level management of data focusing on optimising queries rather than storage: our approach could compliment DCS by providing more effective storage of the data as it is collected.

GHT targets retrieval of high level, precisely defined events. The original GHT implementation is limited to reporting whether a specific high level event occurred. However, it is not able to efficiently locate data in response to more complex queries. The Distributed Index for Features in Sensor Networks (DIFS) attempts to efficiently support range queries. Range queries (Greenstein et al., 2003) are the queries where only events within a certain range are desired. In DIFS, the authors propose a distributed index that provides low average search and storage communication requirements while balancing the load across the participating nodes

Concurrently with us (Tilak et al., 2003), Ganesan et al. (2003) have explored protocols for storage constrained sensor networks. They consider a problem similar to ours and explore some of the solution space we are considering, with some important differences. More specifically, our work differs in the following ways:

- We explore additional approaches to storage management, including those using coordination for redundancy control.
- We explore issues that arise due to uneven data generation (e.g., due to event driven or adaptive sampling applications) and nonuniform storage distribution (e.g., due to nonuniform deployment of the sensors). In such applications, effective load balancing is required.
- We study some additional characteristics of the storage protocols including coverage and collection time and energy.

Conversely, Ganesan et al. (2003) consider some aspects of the problem that we do not examine in detail. For example, in storage constrained networks, one of the issues is how the algorithm behaves when storage is limiting. The proposed approach is to use multiresolution storage – adaptively reducing the resolution of the stored data based on its importance. They explore a policy for multiresolution storage based on the *age* of the data. They proposed and evaluated several novel ageing strategies (reduction of resolution based on age).

5 Storage management protocols

A primary objective of storage management protocols is to efficiently utilise the available storage space to continue collecting data for the longest possible time without losing samples in an energy efficient way. Storage management approaches can be classified as:

- *Local storage*. This is the simplest solution where every sensor stores its data locally. This protocol is energy efficient during the storage phase since it requires no data communication. Even though the storage energy is high (due to all the data being stored), the current state of technology is such that storage costs less than communication. However, this protocol is storage inefficient since the data is not aggregated and redundant data is stored among neighbouring nodes. Local storage is unable to load balance if data generation or the available storage varies across sensors.
- *Collaborative storage*. Collaborative storage refers to any approach where nodes collaborate. This includes cooperation to estimate local redundancy as well exchange of data for aggregation as well as load balancing. Collaboration leads to two benefits:

- *Less data is stored.* Measurements obtained from nearby sensors are typically correlated. This allows data samples from neighbouring sensors to be aggregated.
- *Load balancing*. Collaboration among sensors allows them to load balance the storage.

It is important to consider the energy implications of collaborative storage relative to local storage. Collaborative storage requires sensors to exchange data, causing them to expend energy during the storage phase. However, because they are able to aggregate data, the energy expended in storing this data to a storage device is reduced. In addition, once connectivity with the observer is established, less energy is needed during the collection stage to relay the stored data to the observer. We note that this holds true even if 'in network' aggregation is carried out for locally buffered data during the reachback stage, due to the following two reasons:

- Initial communication (first hop) of the locally buffered data will not be aggregated.
- Less efficient aggregation. A smaller amount of time and resources are available when near real-time data aggregation is applied during reachback as compared to aggregation during the storage phase. Aggregating data during reachback is limited because all the data collected during the storage phase is compressed in a short time.

In the remainder of this section, we first discuss the use of collaboration for data aggregation and then for redundancy control.

5.1 Collaborative storage protocols for data aggregation

One use of collaboration is to take advantage of data aggregation. Aggregation in the application domain we are considering, differs from that in traditional sensor networks because of the non real-time nature. More specifically, in traditional applications, aggregation is carried out using a snapshot of the available data. Data cannot be delayed because it is assumed that an observer is interested in continuously monitoring the data. In the applications we are considering, the data is held locally for extended periods, allowing more effective 'wide angle' aggregation to be carried out.

Aggregation is highly application dependent. Consider a tracking application where nearby sensors exchange a local estimate of distance to a phenomenon. Once this information is available at a single node, it may be triangulated into a single location estimate. In another application, multiple samples from nearby sensors are 'beam formed' to produce a single high quality sample. In order to develop the general storage tradeoff, we abstract the details of the aggregation model and consider only the resulting data size reduction. Primarily, we model aggregation as compression of the collected data samples and vary the compression ratio. This model is not representative of all applications (e.g., beam forming); we discuss the effect of alternative aggregation models later.

A number of organisations are possible for collaborative storage. Data is most correlated and redundant among nearby sensors. Moreover, to minimise data exchange cost, we restrict data exchange within and among neighbouring sensors (recall that the cost of communication is extremely high compared to storage). Finally, data must be collected at a single location for aggregation. As a result of these three factors, a cluster based model suggests itself. Clustering has been widely studied in sensor and ad hoc networks; the specific clustering algorithm used is not important – virtually any existing clustering algorithm can be used. In the remainder, we briefly describe the features of the Cluster Based Collaborative Storage (CBCS) protocol used in our evaluation study. CBCS uses collaboration to take advantage of data aggregation.

In CBCS, clusters are formed in a distributed connectivity based or geography based fashion. Each sensor sends its observations to the elected Cluster Head (CH) periodically. The CH then aggregates the observations and stores the aggregated data. Only the CH needs to store aggregated data, thereby resulting in low storage. The clusters are rotated periodically to balance the storage load and energy usage. Note that only the CH needs to keep its radio on during its tenure, while a cluster member can turn off its radio except when it has data to send. This results in high energy efficiency: idle power consumes significant energy in the long run if radios are kept on. The reception of unneeded packets while the radio is on also consumes energy.

Operation during CBCS can be viewed as a continuous sequence of rounds until an observer/base station is present and the reachback stage can begin. Each round consists of two phases:

- *CH selection phase.* In this phase, each sensor advertises its resources to its one hop neighbours. Based on this resource information, a cluster head (CH) is selected. The remaining nodes then attach themselves to that CH during the data transfer phase
- *Data exchange phase.* If a node is connected to a CH, it sends its observations to the CH; otherwise, it stores its observations locally.

The CH selection approach used in CBCS is based on the characteristics of the sensor nodes such as available storage, available energy or proximity to the 'expected' observer location. The criteria for CH selection can be arbitrarily complex; in our experiments we used available storage as the criteria.

We borrow the idea of cluster head rotation for load balancing from the LEACH protocol (Heinzelman, 2000). CH rotation is done by repeating the cluster selection phases with every round. The frequency of cluster rotation influences the performance of the protocol. Depending on

the cluster formation criteria, there is an overhead for cluster formation due to the exchange of messages.

The cluster selection approach above may result in a situation where a node A selects a neighbour B to be its CH, when B itself selects C (which is out of range with A) to be its own CH. This may result in chains of cluster heads leading to ineffective/multi-hop clustering. To eliminate the above problem and restrict clusters to one hop, geographical zoning is used: an idea that is similar to the approach of constructing virtual grids (Xu et al., 2001). More specifically, the sensor field is divided into zones, such that all nodes within a zone are in range with each other. Cluster selection is then localised to a zone such that, a node only considers cluster advertisements occurring in its zone. Only one CH is selected per zone, eliminating CH chaining as discussed above. We note that this approach requires either preconfiguration of the sensors or the presence of a location discovery mechanism (GPS cards or a distributed localisation algorithm (Bulusu et al., 2000)). In sensor networks, localisation is of fundamental importance, as the physical context of the reporting sensors must be known, in order to interpret the data. We therefore argue that our assumption that sensors know their physical coordinates is realistic. In any case, we emphasise that cluster formation is orthogonal to collaborative storage and other cluster formation approaches can be used.

5.2 Coordination for redundancy control

One idea we explore is coordination among the sensors. Specifically, each sensor has a local view of the phenomenon, but cannot assess the importance of its information, given that other sensors may report correlated information. For example, in an application where three sensors are sufficient to triangulate a phenomenon, ten sensors may be in a position to do so and be storing this information locally or sending it to the cluster head for collaborative storage. Through coordination, the cluster head can inform the nodes of the degree of the redundancy, allowing the sensors to alternate triangulating the phenomenon. Coordination can be carried out periodically at low frequency, with a small overhead (e.g., with CH election). Similar to CH selection, the nodes exchange meta data describing their reporting behaviour and we assume that some application specific estimate of redundancy is performed to adjust the sampling rate.

Coordination can be used in conjunction with local storage or collaborative aggregated storage. In Coordinated Local Storage (CLS), the sensors coordinate periodically and adjust their sampling schedules to reduce the overall redundancy, thus reducing the amount of data that will be stored. Note that the sensors continue to store their readings locally. Relative to Local Storage (LS), CLS results in a smaller overall storage requirements and savings in energy in storing the data. This also results in a smaller and more energy efficient data collection phase. Similarly, Coordinated Collaborative Storage (CCS) uses coordination to adjust the sampling rate locally. Similar to CBCS, the data is still sent to the cluster head where aggregation is applied. However, as a result of coordination, a sensor can adapt its sampling frequency/data resolution to match the application requirements. In this case, the energy expended in sending the data to the cluster head is reduced because of the smaller size of the generated data, but the overall size of the data is not reduced. We evaluate CLS and CCS compared to the noncoordinated counterparts, LS and CBCS.

6 Experimental evaluation

We simulated the proposed collaborative storage protocols using the NS-2 simulator. We use a CSMA based MAC layer protocol. A sensor field (http://www.isi.edu/nsnam/ns/) of $350\ m^2 \times 350\ m^2$ is used with each sensor having a transmission range of 100 metres. We considered three levels of sensor density: 50 sensors, 100 sensors and 150 sensors deployed randomly. We divide the field into 25 zones (each zone is $70 \times 70\ m^2$ to ensure that any sensor in the zone is in range with any other sensor). The simulation time for each scenario was set to 500 seconds and each point represents an average of over five different topologies. Cluster rotation and coordination are performed every 100 seconds in the appropriate protocols.

We assume that the sensors have a constant sampling rate (set to one sample per second). For the coordinated redundancy control protocols, we used a scenario where the available redundancy was, on average, 30% of the data size–this is the percentage of the data that can be eliminated, using coordination. We note that this reduction in the data size represents a portion of the reduction possible, using aggregation. With aggregation, the full data is available at the cluster head and can be compressed at a higher efficiency.

Several sensor nodes that are appearing on the market, including Berkeley MICA nodes, have Flash memories. Flash memories have excellent power (http://www.xbow.com) dissipation properties and small form factor. As a representative we consider a Simple Tech flash memory USB cards it Transfer Energy/Mbyte 0.055 J. In current wireless (http://www.simpletech.com/products/consumer/datasheets/R187.pdf) communication technologies (Radio Frequency based), the cost of communication is high compared to the cost of storage. For example, representative radios following the Zigbee IEEE 802.15.4 standard, consume energy at roughly 40 times the cost of the Simple Tech USB card above per unit data. Our energy models in the simulation are based on these two devices. Further, we adjust the radio properties to match those of a Zigbee device.

Note that both the possible data aggregation /compression and the reduction due to redundancy control are application as well as topology dependent. Consider a temperature sensing application. For this application, a given sensor can collect data from all its neighbours and then simply take the average and store a single value (or maybe minimum, mean and maximum values) as representative. However, if the sensors are sending video

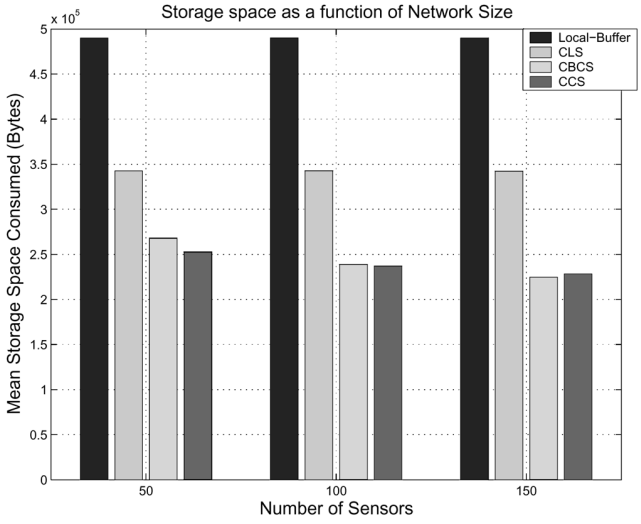
data, then such high spatial compression might not be possible. In this paper, instead of considering a specific application, we assume a data aggregation model where the cluster head is able to compress the size of the data by an aggregation ratio α. By controlling α, we can consider different applications with different levels of available spatial correlation. While this model is useful in exposing the tradeoff space for collaborative storage, it is not representative of all applications. More specifically, the size of the aggregated data grows linearly with the number of available sensors, rather than as a function of the phenomenon, as should be the case under effective monitoring. We consider the implications of this model on collaborative storage and explore other possible models later in this section.

6.1 Storage and energy tradeoffs

Figure 1 shows the average storage used, per sensor, as a function of the number of sensors (50, 100 and 150 sensors) for the four storage management techniques:

- local storage (LS)
- cluster-based collaborative storage (CBCS)
- coordinated local storage (CLS)
- coordinated collaborative storage (CCS).

Figure 1 Storage space vs. network density

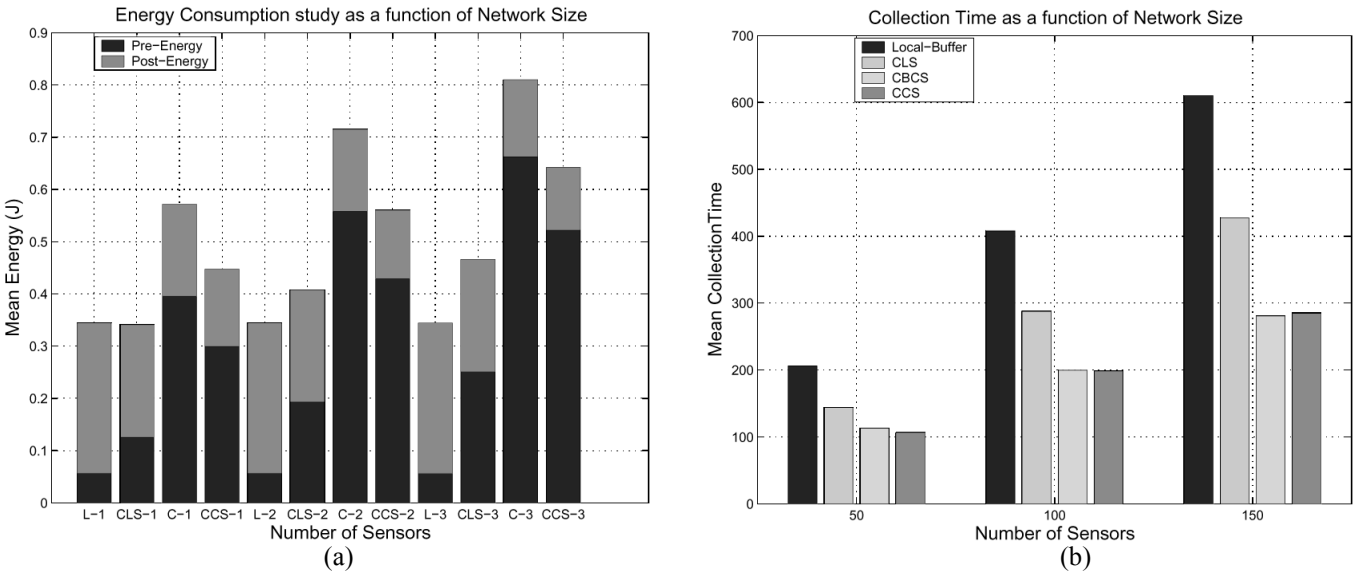


In the case of CBCS, aggregation ratio was set to 0.5. The storage space consumption is independent of the density for LS and is greater than storage space consumption in CBCS and CCS (roughly in proportion to the aggregation ratio). CLS storage requirement is in between the two approaches because it is able to reduce the storage requirement using coordination (we assumed that coordination yields improvement uniformly distributed between 20% and 40%). Note that after data exchange, the storage requirement for CBCS and CCS are roughly the same, since aggregation at the cluster head can reduce the data to a minimum size, regardless of whether coordination took place or not.

Surprisingly, in the case of collaborative storage, the storage space consumption decreases slightly as the density increases. While this is counter intuitive, it is due to higher packet loss observed during the exchange phase as the density increases; as density increases, the probability of collisions increases. These losses are due to the use of a contention based unreliable MAC layer protocol: when a node wants to transmit its data to the CH. The negligible difference in the storage space consumption between CBCS and CCS is also an artefact due to the slight difference in the number of collisions observed in the two protocols. The use of a reliable protocol such as that in IEEE 802.11 or a reservation based protocol such as the TDMA based protocol employed by LEACH (Heinzelman, 2000) can be used to reduce or eliminate losses due to collisions (at an increased communication cost). Regardless of the effect of collisions, one can clearly see that the collaborative storage achieves significant savings in storage space compared to local storage protocols (in proportion to the aggregation ratio).

Figure 2(a) shows the consumed energy for the protocols in Joules as a function of network density. The X axis represents protocols for different network densities: L and C stand for local buffering and CBCS respectively. *L*-1, *L*-2, and *L*-3 represents the results with local buffering technique for network size 50, 100 and 150 respectively. The energy bars are broken into two parts: preenergy, which is the energy consumed during the storage phase, and postenergy, which is the energy consumed during data collection (the relaying of the data to the observer). The energy consumed during storage phase is higher for collaborative storage because of the data communication among neighbouring nodes (not present in local storage) and due to the overhead for cluster rotation. CCS spends less energy than CBCS due to reduction in data size that results from coordination. However, CLS has higher expenditure than LS since it requires costly communication for coordination. This cost grows with the density of the network because our coordination implementation has each node broadcasting its update, and receiving updates from all other nodes.

Figure 2 Energy consumption and collection time study (a) Energy consumption vs. density and (b) Mean collection time vs. density



For the storage and communication technologies used, the cost of communication dominates that of storage. As a result, the cost of the additional communication during collaborative storage might not be recovered by the reduced energy needed for storage except at very high compression ratios. This tradeoff is a function of the ratio of communication cost to storage cost; if this ratio goes down in the future (for example, due to the use of infrared communication or ultra low power RF radios), collaborative storage becomes more energy efficient compared to local storage. Conversely, if the ratio goes up, collaborative storage becomes less efficient

The data collection model depends on the application and network organisation; several models are in use for deployed sensor networks. We use a simple collection model where we only account for the cost of transferring the data one hop. This model is representative of an observer that moves around and gathers data from the sensors. Also, in cases where the local buffering approach carries out aggregation at the first hop towards the observer, the size of the data becomes similar in the two approaches and the remainder of the collection cost is the same. However, this is optimistic in favour of local storage because near real-time data aggregation will not, in general, be able to achieve the same aggregation level during collection as is achieved during collaborative storage. This is due to the fact that collaborative storage can afford to wait for samples and compress them efficiently. Moreover, in collaborative storage, the aggregation is done incrementally over time, requiring fewer resources than aggregation during collection, where large amounts of data are processed during a short time period. The collaborative storage approaches outperform the local storage ones according to this metric due to their smaller storage size. CLS outperforms LS for the same reason.

Finally, we assumed a reachback model (eventual data collection) where the data is read once at the end, consistent with offline monitoring applications. In case of dynamic applications, data may be accessed multiple times by different observers. In this case, the energy saving in the post phase will be further in favour of collaborative storage because the reduced data size ends up benefiting multiple queries.

Figure 2(b) shows that, with collaborative storage, the collection time is considerably lower than that of local buffering. In addition, CLS outperforms LS. Low collection time and energy are important parameters from a practical standpoint. After exploring the effect of coordination, the remainder of the paper presents results only with the two uncoordinated protocols (LS and CBCS).

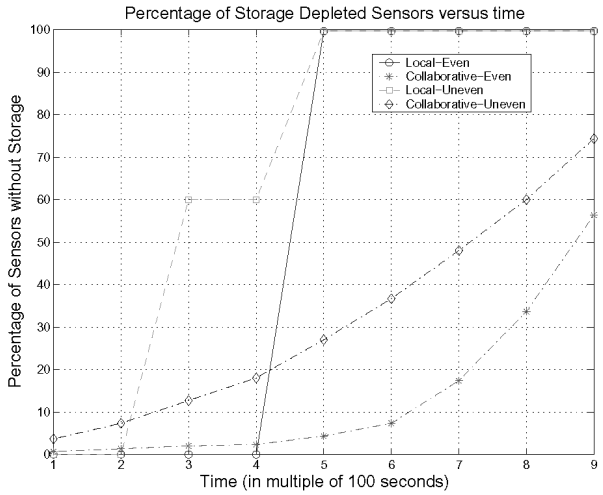
6.2 Storage balancing effect

In this study, we explore the load balancing effect of collaborative storage. More specifically, the sensors are started with a limited storage space and the time until this space is exhausted is tracked. We consider an application where a subset of the sensors generates data at twice the rate of the others, for example, in response to higher observed activity close to some of the sensors. To model the data correlation, we assume that sensors within a zone have correlated data. Therefore all the sensors within a zone will report their readings with the same frequency. We randomly select zones with high activity; sensors within those zones will report twice as often as those sensors within low activity zone.

In Figure 3, the X-axis denotes time (in multiples of 100 seconds), whereas the Y-axis denotes the percentage of sensors that have no storage space left. Using LS, in the even data generation case, all sensors run

out of storage space at the same time and all data collected after that is lost. In comparison, CBCS provides longer time without running out of storage, because of its more efficient storage.

Figure 3 Percentage of storage depleted sensors vs. time



The uneven data generation case highlights the load balancing capability of CBCS. Using LS, the sensors that generate data at a high rate exhaust their storage quickly; we observe two subsets of sensors getting their storage exhausted at two different times. In comparison, CBCS has much longer mean sensor storage depletion time due to its load balancing properties, with sensors exhausting their resources gradually, extending the network lifetime much longer than LS.

6.3 Coverage analysis

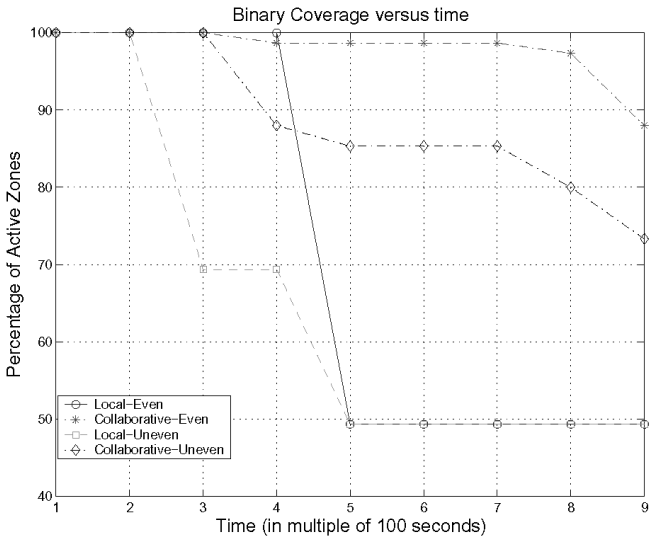
Physically colocated sensors have redundant data. For simplicity, we assume that all sensors within a zone have correlated data. In this work we consider two types of coverage, namely, binary coverage and manifold coverage, defined as follows:

- *Binary coverage*. A given zone Z_i is said to be covered at time t if any one of the sensors $S_1, \ldots, S_k$ in Z_i is reporting and storing the reading. Binary coverage can be visualised as a step function.
- *Manifold coverage*. a given zone Z_i is said to be covered at time t proportional to the number of sensors $j (j < k)$ out of its given set of sensors $S_1, \ldots, S_k$ that are reporting and storing the reading.

This coverage function can be visualised as a monotonically increasing function (which might have diminishing returns after some point). This means that the higher the number of reporting sensors, the better is the coverage.

Figure 4 shows the Binary Coverage as a function of time. One can see that CBCS has a higher percentage of active zones compared to LS for both data generation models.

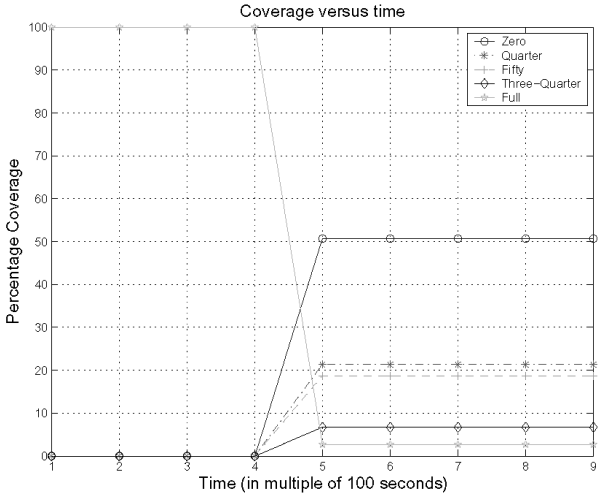
Figure 4 Binary coverage



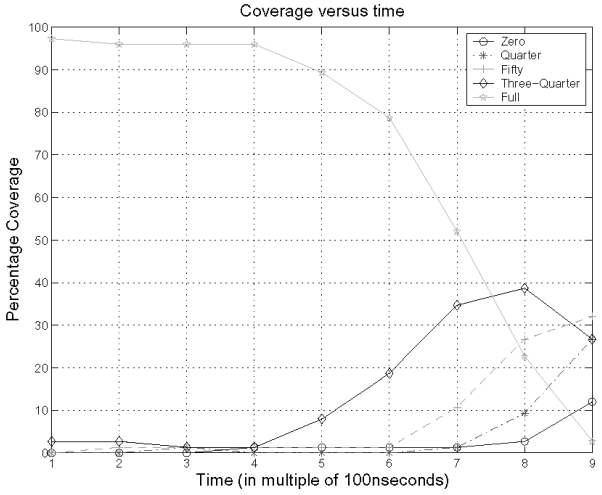
Similar trends are seen when considering Manifold Coverage (Figure 5). Each line represents the percentage of zones with some specific coverage level: for example, the line 'quarter' represents the percentage of zones where at least 25% of the sensors have storage space left. One can clearly see that in the case of LS, with even data generation (Figure 5(a)) the percentage of zones with full coverage is 100% at 300 seconds, whereas with uneven data generation it reduces to less than 50% within 300 seconds. In CBCS, at the same times, the coverage is around 96% with the even data generation model and with the uneven data model it is around 77%. Note that, a CH stored more data than an individual sensor; therefore if the round time is very long, it might happen that the given CH runs out of storage sooner than a sensor storing its data locally. In LS, the percentage of dead zones (zones with all sensors out of storage space) rises in two waves for the uneven data model, reaching up to 30% within 300 seconds and 50% in 500 seconds However, with CBCS, with the uneven data model, the percentage of dead zones rises slowly and is below 30% even at the end of the simulation.

In general, from these figures, one can see that the manifold coverage changes are abrupt for local buffering. In contrast, collaborative storage provides smooth degradation of coverage. Moreover, the average coverage is higher for collaborative storage due to the data aggregation and load balancing ability, by transferring data from high activity zones to low activity zones.

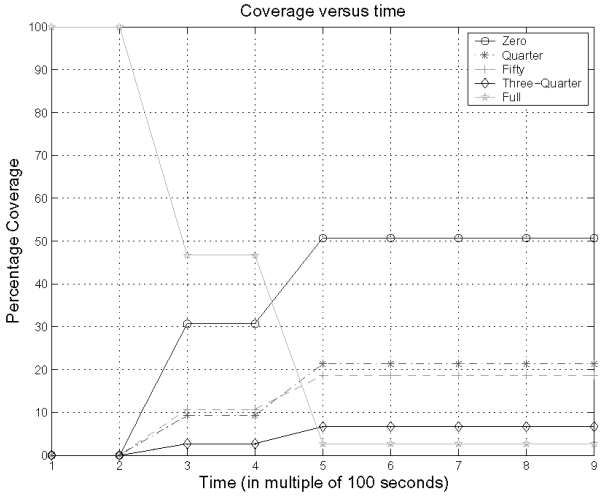
Figure 5 Manifold coverage (a) LS manifold coverage (even data generation); (b) CBCS manifold coverage (even data generation); (c) LS manifold coverage (uneven data generation) and (d) CBCS manifold coverage (uneven data generation)



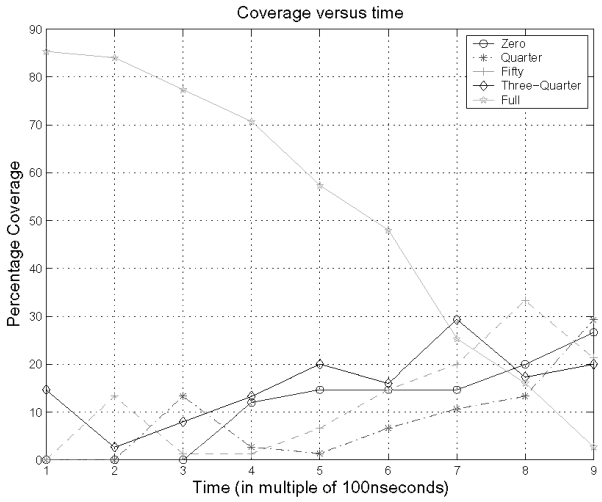
(a)



(b)



(c)



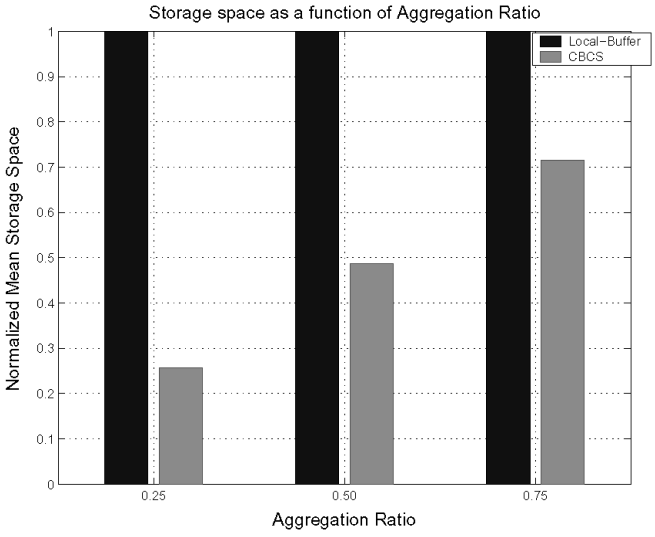
(d)

6.4 *Effect of the aggregation model*

Figure 6 shows effect of storage space consumption as a function of aggregation ratio. As expected, the amount of storage space consumed in the case of CBCS protocol is proportional to the extent of aggregation ratio. Collaborative storage management will work well for the applications with high spatio-temporal coverage.

One limitation of the aggregation model we have used so far is that the required storage size under collaboration grows in direct proportion to the number of sensors in the cluster; that is, the storage consumed in a round is $\alpha N \cdot D$, where α is the aggregation ratio, N is the number of sensors and D is the data sample size. Since the available storage ($N \cdot S$, where S is the available storage per sensor) is also a function of the number of sensors, storage is consumed at a rate (*fracαDS*) which is independent of the number of sensors present in the zone, assuming perfect load balancing. For most applications, this will not be the case: the aggregated data necessary to describe the phenomenon in the zone does not grow strictly proportionately to the number of sensors and we expect storage lifetime to be longer in dense areas than in sparse ones.

Figure 6 Storage space vs. aggregation ratio

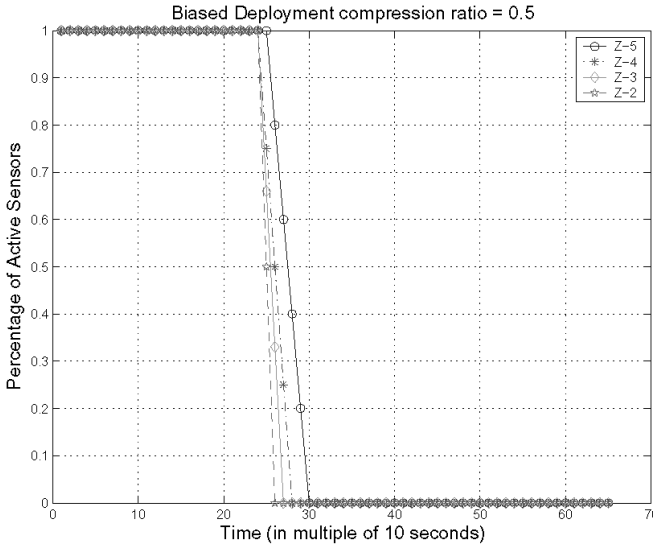


To highlight the above effect, we consider the case of a *biased deployment* where sensors are deployed randomly but with nonuniform density. In addition to the aggregation model considered so far, we consider a case where the CH, upon receiving packets from its N members, just needs to store one packet; for example, if the aggregation function is to store the average value of the N samples (e.g. average temperature reading). Clearly, in the second case, the size of the aggregated data is independent of network density. We now study how these applications with different aggregation functions perform on top of a biased deployment. To model biased deployment, we consider four zones with sensors respectively. In these simulations, the round time was set to 10 seconds (CH selection happens every 10 seconds).

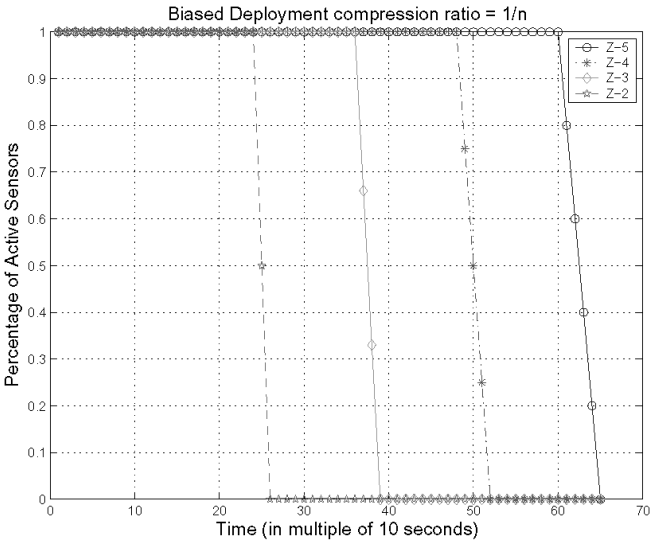
In Figure 7, the X-axis shows time (in multiple of 10 seconds), whereas the Y-axis shows the percentage of coverage sensors within a given zone. As described earlier we considered 4 zones for this study and each line in the Figure 7 represents a particular zone. For example line *Z*-5 stands for a zone with five sensors in it and *Z*-2 denotes the zone with two sensors in it and so on. As shown in Figure 7(a), when the aggregation ratio is a constant (0.5), all the zones provide coverage for almost same duration. However, in the second case, as shown in Figure 7(b), coverage is directly proportional to the network density; the higher the density, the longer is the coverage.

The sensor network coverage from a storage management perspective depends on the event generation rate, the aggregation properties as well as the available storage. If the aggregated data size is independent of the number of sensors (or grows slowly with it), the density of the zone correlates with the availability of storage resources. Thus, both the availability of storage resources as well as the consumption of them may vary within a sensor network. This argues for the need of load balancing across zones to provide long network lifetime and effective coverage. This is a topic for future research.

Figure 7 Biased deployment vs. coverage (a) Aggregation ratio = 0.5: Coverage and (b) Aggregation ratio = 1/N: Coverage



(a)



(b)

7 Conclusion and future work

In this paper, we considered the problem of storage management in sensor networks where the data is not continuously reported in realtime and must therefore be stored within the network. Collaborative storage is a promising approach for storage management because it enables the use of spatial data aggregation and redundancy control among neighbouring sensors to compress the stored data and optimise the storage use. Collaborative storage also allows load balancing of the storage space to allow the network to maximise the time before data loss due to insufficient memory. Collaborative storage results in lower time to transfer the data to the observer during the reachback stage and has better binary and manifold coverage than a simple local buffering approach.

While collaborative storage reduces the energy required for storage, it requires additional communication. Using current technologies, collaborative storage requires more energy than local buffering. Network effectiveness is bound both by storage availability (to allow continued storage of collected data) as well as energy. Thus, protocol designers must be careful to balance these constraints: if the network is energy constrained, but has abundant storage, local storage is most efficient from an energy perspective. Alternatively, if the network is storage constrained, collaborative storage is most effective from a storage perspective. When the network is constrained by both, a combination of the two approaches would probably perform best.

We did not examine the implications of collaborative storage on data indexing and retrieval strategies. Furthermore we did not consider the effect on the file system design: often sensor network files have simple sequential 'append access' interfaces that are suitable for data logging, but not necessarily collaborative storage. These issues are among the areas of future research interest.

As part of our future research, we would like to implement these protocols on real sensor hardware platforms such as the Berkeley motes. Furthermore, in this study we consider all events to be of the same importance, and thus they are stored with the same compression ratio (resolution). In our future research, we will explore the protocol space wherein different events are stored with different resolutions (important events are stored in detail whereas unimportant events are stored with a coarser granularity).

References

Asada, G., Dong, M., Lin, T., Newberg, F., Pottie, G. and Kaiser, W. (1998) 'Wireless integrated network sensors: low power systems on a chip', *European Solid State Circuits Conference*, October, Elsevier, pp.9–12.

Boulis, A., Ganeriwal, S. and Srivastava, M. (2003) 'Aggregation in sensor networks: an energy-accuracy trade-off', *Proc. of the First IEEE Internation Workshop on Sensor Network Protocols and Applications (SNPA)*, pp.128–138.

Bulusu, N., Heidemann, J. and Estrin, D. (2000) 'GPS-less low cost outdoor localization for very small devices', *IEEE Personal Communications Magazine*, October, pp.28–34.

Chandrakasan, A., Amirtharajah, A., Cho, S., Goodman, J., Konduri, G., Kulik, J., Rabiner, W. and Wang, A. (1999) 'Design considerations for distributed microsensor systems', *Proc. of the IEEE 1999 Custom Integrated Circuits Conference (CICC'99)*, May, pp.279–286.

Ganesan, D., Greenstein, B., Perelyubskiy, D., Estrin, D. and Heidemann, J. (2003) 'An evaluation of multi-resolution storage for sensor networks', *Proceedings of the ACM SenSys Conference*, *ACM*, November, Los Angeles, California, USA, pp.89–102.

Greenstein, B., Estrin, D., Govindan, R., Ratnasamy, S. and Shenker, S. (2003) 'Difs: a distributed index for features in sensor networks', *First IEEE International Workshop on Sensor Network Protocols an Applications (SNPA, 2003)*, pp.163–173.

Heinzelman, W. (2000) *Application-Specific Protocol Architectures for Wireless Networks*, PhD thesis, Massachusetts Institute of Technology.

Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D. and Pister, K. (2000) 'System architecture directions for network sensors', *Proc. of ASPLOS*, November, pp.93–104.

Intanagonwiwat, C., Estrin, D., Govindan, R. and Heidemann, J. (2001) 'Impact of network density on data aggregation in wireless sensor networks', *Tech. Rep. TR-01- 750*, University of Southern California, November.

Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh, L.S. and Rubenstein, D. (2002) 'Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet', *Prof. of ASPLOS 2002*, ACM Press, San Jose, CA, pp.96–107.

Ratnasamy, S., Estrin, D., Govindan, R., Karp, B., Shenker, S., Yin, L. and Yu, F. (2002) 'Data-centric storage in sensornets', *Proceedings of the First ACM SIGCOMM Workshop on Hot Topics in Networks*, October.

Tilak, S., Abu-Ghazaleh, N. and Heinzelman, W.B. (2005) 'Storage management in wireless sensor networks', *Mobile, Wireless and Sensor Networks*, John Wiley publishers.

Tilak, S., Abu-Ghazaleh, N. and Heinzleman, W. (2003) 'Storage management issues for sensor networks', Student Posters *(ICNP 2003)*, November.

Xu, Y., Heidemann, J. and Estrin, D. (2001) 'Geography-informed energy conservation for ad hoc routing', *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, ACM Press, pp.70–84.

Yao, Y. and Gehrke, J. (2002) 'The cougar approach to in-network query processing in sensor networks', *SIGMOD Record 31*, September, p.3.

Websites

A remote ecological micro-sensor network, http://www.botany.hawaii.edu/pods/overview.htm

Crossbow–smart sensors in silicon (crossbow website) (2003) *Commercial product based on Berkeley MICAs* http://www.xbow.com.

Network Simulator, http://isi.edu/nsnam/ns.

SimpleTech flash memory card datasheet, http://www.simpletech.com/products/consumer/ datasheets/R187.pdf.