RESTORE: A Real-Time Event Correlation and Storage Service for Sensor Networks

Sudha Krishnamurthy
Deutsche Telekom Laboratories
Berlin, Germany
sudha.krishnamurthy@telekom.de

Tian He
Department of Computer Science
University of Minnesota
Minneapolis, MN
tianhe@cs.umn.edu

Gang Zhou, John A. Stankovic, Sang H. Son
Department of Computer Science
University of Virginia
Charlottesville, VA
{gzhou,stankovic,son}@cs.virginia.edu

***Abstract*— This paper describes the design of RESTORE, which is a framework for providing in-network event correlation and storage service for sensor environments. RESTORE uses a data-centric approach in which it partitions a sensor network into zones and maps every event to a zone. The sensor nodes in a zone make use of their cooperative storage resources and redundancy to improve information availability and energy efficiency. RESTORE is useful for temporarily buffering information in sensor environments that have intermittent connectivity to a base station. RESTORE may also be used as the underlying framework for an event notification service that publishes events directly from a sensor network to the subscribers. The contributions of this paper include a) an event taxonomy for correlating sensor events within the network, and b) mechanisms to organize information using the collaborative resources, in order to enable real-time event correlation within a sensor network. We also present preliminary simulation results that examine how the collaboration within a zone in RESTORE impacts the energy consumption, information availability, and message overhead.**

I. Introduction

Wireless sensor networks are primarily large-scale, decentralized information systems. An information system for sensors needs to have the ability to store the stream of observations generated locally by the sensors and more importantly, correlate the raw observations from different parts of the network to generate events that are meaningful to the end users. There have been several efforts to correlate the information gathered by the deployed sensors and generate higher-level inferences in the context of different applications, such as surveillance [1], environmental monitoring [2], [3], and structural health monitoring [4]. However, in a majority of these efforts, the storage and correlation of sensor information is performed at the edge of the network in more powerful devices that serve as base stations. This approach is certainly the most viable option when the devices in the network are primitive and the infrastructure allows easy and uninterrupted access to the powerful edge devices. However, recent technological advances are making it possible to integrate multi-modal sensing capabilities, resulting in network devices that are smarter and capable of fine-grained classification. Should this technological trend continue, performing event correlation and storage within the sensor network should be an increasingly viable option and benefit several applications. In this paper, we motivate the need for an in-network service that buffers the observations from the sensors for a limited period of time and uses that as a basis for correlating events within the network. We also present an architecture that makes use of the inherent redundancy and collaborative nature of sensor networks to realize such a service.

A. The Case for In-Network Correlation and Storage

The use of external base stations for storage and correlation has some drawbacks. Unlike the tiny sensor devices that can be left unattended in remote environments, the base stations are often more power-consuming and less unobtrusive. Leaving these base stations unattended in remote sites may not be a feasible option, especially in harsh and unfriendly environments where stealthiness is important, such as in battlefield surveillance applications. This problem can be addressed by using long-range relays to transmit the sensor observations to remote base stations. However, this long-range communication may not always be reliable. End-to-end connectivity may be intermittent or periodic and may be disrupted due to several factors, such as the weather. Moreover, such long-range communication may be more vulnerable to interception.

A better alternative in such environments, where communication may be disrupted, is to buffer the observations of the sensors within the sensor network for limited periods of time and correlate the observations within the network. Event notifications generated as a result of the correlation can then be directly delivered to the end subscribers. The data stored in the network may also be uploaded in batches, whenever the base stations are connected. Such an approach reduces the dependence on external base stations and minimizes the use of long-range communication. Recent efforts, such as the Zigbee gateway working group [5], delay-tolerant architectures [6], and TinyREST [7] are addressing the interoperability issues involved in integrating the sensor networks more closely with the Internet and traditional networks. We think that an in-network event correlation and storage service would vastly improve the utility of sensor networks and provide impetus and motivation for such efforts that attempt to bridge the gap between sensor networks and traditional networks.

Sensor networks are typically perceived as challenged environments [6]. So a valid question is whether the deployed sensor devices are capable of realizing an in-network storage

and correlation service. Recent technological trends have been encouraging in this regard. If the sensor devices are low-end devices, like motes, each of which has a storage capacity of only 512 KB [8], building an in-network storage scheme will require cooperative caching schemes. However, if the motes are deployed along with devices that have larger storage capacity, such as PDAs and Stargate processors [9], then such a heterogeneous collection of devices would make it increasingly feasible to provide long-term, in-network storage.

In order to implement an in-network event correlation service, the nodes in the network should at the minimum be able to order events in space and time. Mote-like devices possess this ability and this allows them to perform simple correlations pertaining to a single object, such as tracking the motion of a single car with the help of magnetic sensors. However, if multiple cars of identical size appear in the network concurrently and their trajectories intersect, this basic ability is insufficient to distinguish between the different objects. To perform event correlation within the network in such scenarios, we would need nodes with the ability to perform finer-grained classification. Recent advances in nodes with multi-modal sensing is promising in this regard. For example, the integrated RFID-sensor systems from SkyeTek [10] combine a MicaDot node [11] with an RFID reader. This combination of sensors and RFID provides access to a richer source of data and allows different instances of the same kind of event to be distinguished, making it possible to perform more fine-grained event correlation within the network.

B. Overview of RESTORE

In this paper, we describe an in-network event-correlation and storage service for sensor networks, called RESTORE. The design principle behind RESTORE is that no single node in a zone has enough resources to store and correlate information about all the events in the network. So RESTORE makes use of the inherent redundancy and cooperative resources of a sensor network to store and correlate events within the network. RESTORE organizes the nodes in the sensor network into zones. Each zone is responsible for storing information about one or more events. A zone functions like a cooperative cache and is the smallest unit of collaborative storage in the sensor network. Each zone has a storage manager that is responsible for coordinating the storage and event correlation within its zone. Zone members collaborate to achieve energy efficiency and fault tolerance. The two main goals of RESTORE are the following:

Real-time event correlation: RESTORE enables a sensor network to publish event notifications directly to the subscribers, without any intermediate processing at base stations. In many applications, the subscribers may be interested in being notified about composite events instead of every individual event. The cooperative resources of the members within a zone can be used to record observations from sensors across the network in a decentralized manner to form a more composite view of the events in the network. As new observations arrive, they can be correlated with relevant portions of the stored data in real-time. While the correlation happens as new data arrives, the event notifications are generated according to the frequency and specifications of the subscribers.

Short-term, reliable storage: RESTORE not only minimizes the need for long-range communication, it also provides reliability. In environments where connectivity to an external persistent storage is intermittent, if the observations are directly transmitted to the base station and are lost in the process due to unreliable channels, the base station may not be able to generate some event notifications. In such an environment, RESTORE enables a sensor network to store information and correlate events within the network. If the event notifications to the subscribers are lost, they can be retransmitted, because the observations used to generate the notifications are stored in the network for a period of time. Whenever a base station is available, the individual zone managers can upload the stored data from their zones and reclaim the storage resources. RESTORE also allows the stored data to be uploaded to mobile base stations, which is useful for applications, such as, wildlife tracking[12].

The RESTORE architecture described in this paper assumes a network consisting of homogeneous low-end devices, like the motes. However, if nodes with more powerful storage capabilities, such as PDAs, are also present, RESTORE would take advantage of such nodes by favoring them as storage managers. RESTORE also takes advantage of RFID-enabled objects, if they are present, and uses the tags to distinguish between different instances of an object and perform fine-grained event correlation.

The rest of the paper is organized as follows. In Section II, we describe the use of an in-network correlation and storage service for a few application scenarios. In Section III, we discuss the architecture of RESTORE and describe how it partitions the network into zones, in order to provide energy-efficient storage. In Section IV, we present an event taxonomy and describe how RESTORE uses the zones to store and correlate information for events characterized by this taxonomy. In Section V, we present preliminary performance results. In Section VI, we compare RESTORE with other sensor network storage schemes. In Section VII, we present our conclusions.

II. Applications

We now illustrate the use of an in-network event correlation and storage service, like RESTORE, in some typical sensor applications.

- Consider an application that makes use of integrated RFID-sensor nodes in parking meters to monitor parking violations. Most vehicles these days have RFID-based toll tags or license plates. The sensors keep track of the parking duration and the RFID reader identifies the vehicle. The sensor nodes are grouped into regions and collectively store the violations in that region. The stored data can be logged to an external persistent storage either periodically or when the number of violations in a region exceeds a certain threshold. Alternatively, a cop can drive around with a PDA and retrieve the stored data from

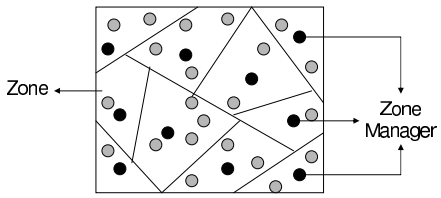

Fig. 1. Zone partitioning in RESTORE

the wireless sensors. Reliable storage is important here, because loss of data results in improper law enforcement and loss of revenue. The in-network event correlation triggers an event whenever the number of violations in a region exceeds a certain threshold.

- Consider an application that makes use of sensors to monitor the behavior of autistic patients. These patients usually follow a very regular pattern and caregivers need to be notified only when the behavior on a particular day does not match the pattern. An in-network store can be used to record the behavior of the patients daily. As new sensor observations arrive each day, they can be correlated with the stored data and if there is a significant deviation in the behavior of a patient on any day, the in-network correlation service triggers an event notification.
- Whenever new sensor network protocols are developed, they often need to be debugged and evaluated experimentally through an actual deployment. This usually involves collecting statistics from the sensor network over a period of time and performing an offline analysis of the observations. During this phase, the sensors are typically deployed in isolated environments where it may not be possible to provide continuous access to base stations for the entire experimental period. In such a scenario, it would be convenient to eliminate the use of base stations by using an in-network store to collect the statistics at runtime. The data collected can then be uploaded to a base station at the conclusion of the experimental period.

III. Description of RESTORE Architecture

The RESTORE architecture needs to support in-network storage and correlation of events in a timely manner. One way to achieve this is to allow the nodes that have observed an event to store the information locally and then retrieve the information from all the nodes that have observed the same event during the correlation process. However, such a dispersed storage of information makes the retrieval process more complex and increases the time for correlation. Instead, our goal in designing RESTORE is to enable local decision making based on a global view of an event. In order to achieve this, RESTORE logically partitions a sensor network into storage zones as shown in Figure 1. RESTORE follows a data-centric approach in which it maps every network event to a zone. The zone to which an event is mapped becomes the primary storage zone (PSZ) for that event. Members of that zone cooperatively use their resources to store and correlate information gathered by sensors across the network about that event. One of the nodes in each zone serves as the *zone manager*. The zone managers serve as the primary storage managers (PSM) in their zone. They are responsible for deciding how to store and correlate information within their zone by making use of the cooperative resources of their zone members to achieve energy efficiency and increased availability.

Thus, the design of RESTORE must address two main issues: a) a mechanism for partitioning a sensor network into zones, and b) mechanisms for utilizing the collective resources of a zone for in-network storage and event correlation. In order to address these issues, RESTORE makes the following assumptions about the nodes in the network. RESTORE assumes that all of the sensor nodes know their location relative to each other with some degree of accuracy. In this paper, we assume that all sensing nodes have the same sensing range, although this can be relaxed when multiple types of sensors are involved. RESTORE assumes that the communication range of a node is at least twice its sensing range. This ensures that if managers are chosen in such a way that there is at least one manager in every sensing radius, then adjacent managers will be within communication range of each other. RESTORE also assumes that the sensor network is reasonably dense and has a fairly uniform distribution of nodes. We now describe how RESTORE partitions a sensor network into storage zones. In the next section, we describe how these zones are used for in-network event correlation and storage.

The goals of the partitioning algorithm in RESTORE are as follows. The basic goal is to split the sensor network into disjoint zones in which each zone member is within one-hop communication radius from its manager. There is a zone manager within every sensing radius. Since zone managers are always awake, this helps to provide sensing coverage for the entire network. Every zone must be reachable from every other zone in the network, so that reports about an event detected by a zone can be routed to the PSZ for that event. This is done by allowing all of the zone managers in RESTORE to form a communication backbone for the entire network. A zone in RESTORE is identified by its manager and whenever the members of a zone detect an event, the zone manager uses the identities (locations) of the managers to find the mapping between an event and its PSZ. Hence, every zone manager needs to know the identities of all the other zone managers in the network.

Several clustering and partitioning algorithms have been proposed for sensor networks and some of them follow the minimum dominating set approach to minimize the number of partitions by maximizing the size of each partition (for e.g. [13]). Such an approach primarily attempts to minimize energy consumption and maximize coverage. However, RESTORE uses the partitions to store information about a certain number of distinct events (N) and provide a minimum degree of information availability (A) for each event. The zone size, which we define as the number of nodes in a zone, is a tradeoff between the parameters N and A. Maximizing the

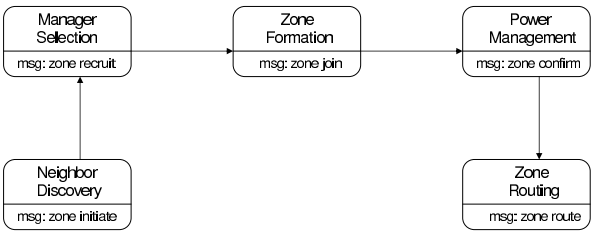

Fig. 2. Phase transition during zone partitioning in RESTORE

size of each zone helps to store more information about an event and improves the information availability for the event. However, it reduces the number of distinct events that can be stored in the network. RESTORE regards both N and A as application-specific inputs. The parameter, A determines the minimum number of nodes that need to be present in each zone. We now describe a way to partition the network into storage zones, in order to meet the goals stated above. Figure 2 shows the phases involved in the partitioning process and the messages transmitted in each phase.

A. Neighbor Discovery

The sensor nodes in RESTORE organize themselves into zones based on their locality. In order to do this, the sensor nodes first discover their neighbors by propagating discovery beacons that contain the sender's location, residual energy, storage capacity, and local clock value. In addition to neighbor discovery, these beacons help the nodes to synchronize their local clocks. Each node broadcasts its discovery beacon locally within its sensing range. We omit the details of how the discovery beacons are propagated through the network, because there are well-known mechanisms to do this (for e.g., [1]).

At the end of the discovery process, every sensor node conceptually has its own zone. Its potential zone members are the nodes within its sensing range that are recorded in its local neighborhood table. In a high density network, some of the nodes may be included in multiple neighboring zones. Hence, the next step is to select a subset of the nodes as zone managers and ensure with a high probability that every node in the network joins the zone of exactly one of the selected managers [1].

B. Manager Selection

The decision to become a manager is taken locally by each node by first computing the eligibility value (EV) for each of the nodes in its neighborhood table. In RESTORE, we use the information that a node receives from its neighbors during the neighbor discovery phase to compute the eligibility value.

[1]On account of the unreliability of the underlying communication channels, we can only provide probabilistic guarantees for the zone membership.

The EV of a node m in a temporary zone z defined by its neighborhood table is denoted as $EV_m(z)$ and is defined by Equation 1. $P_m(t)$ is the probability that a node m that remains awake during the entire interval t will not fail at the end of the interval, where t is the time interval before the managers are rotated again. Here, we primarily consider failure due to power drain. So $P_m(t)$ is determined by the residual energy of the node m. S_m is the storage capacity of node m and d_{mi} is the Euclidean distance between nodes m and i, both of which are members of the temporary zone z.

$$EV_m(z) = \frac{P_m(t) * S_m}{\sum_{i \in z} d_{mi}} \quad (1)$$

A node nominates itself as a zone manager if a) its temporary zone z has at least the minimum number of nodes required to meet the availability A, defined earlier, and b) it has the highest eligibility value among its potential zone members. The first criterion favors the creation of zones that have enough members for storing information. The second criterion favors nodes that have higher residual energy, storage capacity, and that are closer to their zone members as zone managers. Since a zone manager remains awake all the time and consumes energy, choosing a manager node that has higher residual energy is preferable. A zone manager needs to communicate with its zone members in order to store information and the zone members, in turn, need to communicate the information they have stored to their zone manager during event correlation. In order to reduce the energy spent in this two-way communication, it is preferable to choose zone managers that are closer to most of their zone members. Finally, in the case of heterogeneous networks, choosing nodes that have higher storage capacity as zone managers helps reduce the frequency of communication with the zone members.

A node that nominates itself as a zone manager advertises its nomination by locally broadcasting a zone recruit message. The recruit message lists the nodes in the manager's neighborhood table as potential zone members. A node listens to these recruitments for a certain period of time and records a) all the nodes that have nominated themselves as managers in its neighborhood, and b) all those managers that have recruited it as their zone member. The recruitment message serves two purposes. First, it allows the non-manager nodes to decide which zone they should join. Second, a manager node can snoop on the recruit messages broadcast in its neighborhood and build its list of neighboring managers. The next step in zone partitioning is to ensure that the zones are disjoint by allowing every non-managerial node to join only one of the zones in which it has been recruited.

C. Zone Formation

After listening to the recruitment messages for a certain period of time, a node is in one of four states: it may have been recruited as a zone member in a single zone, multiple zones, none of the zones, or it may have nominated itself as a manager.

If a non-manager has been recruited by only a single manager, then it simply acknowledges by sending that manager a zone join message. A non-manager node that has heard recruitment messages from multiple managers, joins the zone of the manager that has recruited the least number of nodes and whose recruitment message had good signal strength. The former criterion tries to distribute the availability across the zones, while the latter tries to ensure good connectivity between a manager and its zone members. The manager nodes snoop on the zone join messages broadcast in their neighborhood and eliminate their potential zone members that have joined other zones. Nodes that have been recruited in multiple zones can serve as bridge nodes between two zones. If a manager does not receive enough join responses to meet the minimum availability requirement mentioned earlier in Section III, then the zone members can regroup and join an appropriate adjacent zone. Alternatively, the manager can meet the required degree of availability by soliciting nodes from nearby zones that have more zone members than required.

At the end of the above two-phase message exchange in which the recruitment messages are acknowledged by join messages, every node in the network is either a zone manager or is a non-managerial member of a single zone. This two-phase exchange also ensures that the communication between a zone manager and its members does not suffer from asymmetric effects [14]. A manager uses the knowledge of its zone members and their residual energies to provide energy-efficient storage, as we now describe.

D. Power Management

Every manager assigns a rank to each of its zone members based on their residual energy. The rank of a member in a zone determines its sleep schedule. Nodes with lower residual energy are assigned higher ranks and sleep for a longer duration of time. Thus the zone members form a hierarchy. A manager broadcasts the ranks to its zone members in a zone confirmation message. This message confirms the membership of a node in a zone and allows the zone members to begin their power management schedule according to their ranks. In addition, since a manager broadcasts this message in its communication range, it allows all the managers to finalize their list of neighboring managers.

Finally, in order to route messages from zones that detect an event (source) to the primary storage zone corresponding to that event (sink), every zone needs to be reachable from every other zone. This can be done by creating a common communication backbone that connects all the zone managers (for e.g., as done in [1]).

In order to balance the energy load, the zone managers need to be rotated. This can be done whenever the data stored in the zones is flushed or when the data is uploaded to a base station that becomes accessible to the sensor network. During each rotation, the phase transition process depicted in Figure 2 is repeated. This process also helps the nodes to resynchronize their local clocks. If the duration between successive rotations is too long, the zone members can synchronize their local clocks with their zone manager's clock periodically, in order to ensure that the clock skew is bounded.

IV. In-Network Event Correlation and Storage

In this section, we describe how the resources within a zone can be collectively used to store and correlate events that occur across the network. When the nodes in a zone detect an event, they send the information they collect to their zone manager. The manager uses a hash function to map the event to its primary storage zone (PSZ) and routes the information it has collected to the PSZ for that event. If the event is mobile, this process is repeated by each zone that detects the event, as the event propagates through the network. The primary storage manager (PSM) receives reports about the event from different zones and uses the resources of its zone members to store and correlate information pertaining to the event in real-time. The information stored in the zones can be retrieved by connecting a base station to some point in the network. The zone managers form a spanning tree to connect to the base station on demand and upload the information from their respective zones. We now describe the mapping process, in-network event correlation, and organization of information within a storage zone.

A. Mapping Events to Zones

RESTORE maps every event to a storage zone using a hash function. Since a storage zone is represented by its zone manager, the hash function effectively maps an event to one of the zone managers. RESTORE can be used with any suitable hash function. For example, if nodes are identified by their geographic coordinates, then a geographic hashing scheme [15] may be used. In order to map an event or object to a zone, RESTORE makes use of a unique identity for each object. This identification may be obtained from RFID tags, wherever tagging is possible, or from application-specific signatures gathered from other types of sensors. While RESTORE does not focus on how the identification is obtained, the type of identification influences the granularity of the mapping. For example, if all the vehicles have RFID tags that allow different brands of the same type of vehicle to be distinguished, then it is possible for RESTORE to assign a different zone to keep track of the traffic statistics of each brand of car along a highway. However, if such fine-grained distinction is not possible, then RESTORE has to resort to a coarser level of mapping, such as that induced by the vehicle size. In such a case, all car-related statistics would be stored in one zone, truck-related statistics would be in another zone, and SUV-related statistics would be in a third zone.

B. Real-time Event Correlation in a Zone

Many of the sensor-based information can be correlated and notifications generated using simple operators, such as max, min, average, and sum. Such simple operations are well within the capability of low-end sensor nodes, such as motes, which makes it possible to carry out event correlation within the network. Table I presents a taxonomy of some of the events

TABLE I
EVENT TAXONOMY

	Static	Mobile
Single scope	temperature monitoring, WSN debugging, parking violations	tracking small vehicles, patient behavior
Adjacent scope	energy monitoring in a building floor	tracking large trucks
Diffuse scope		fire, pipeline cracks, chemical spills

that we can correlate using RESTORE. Each row in the table classifies events based on their scope during their lifetime with respect to a zone in RESTORE, while the columns classify events based on their mobility with respect to the zones. In RESTORE, static events are observed by the same zone or set of zones for their entire lifetime, while mobile events are observed by different zones at different points in time. We now explain this taxonomy with examples.

Single Scope: Each instance of a single-scoped event occurs only in a single zone at any given time. As a result, a PSZ for a single-scoped event receives reports about each individual instance of the event from at most one zone at any given time.

A single-scoped event may be static, in which case the scope of the event is confined to the same zone for the lifetime of the event. An example of a static, single zonal event is a group of sensors monitoring the temperature in their vicinity. Another example is the sensor network debugging application listed in Section II, in which groups of sensors collect statistics locally for later introspection. Monitoring the parking violations in different zones, which was also described in Section II, is also a static, single-scoped event. Each occurrence of a parking violation is reported by only one zone at any instant of time. Different instances of parking violations may be detected at the same time or at different times in multiple zones. However, the violations that are detected within a zone neither propagate to other zones, nor are they typically related to the violations reported by other zones. Hence, the events in this case are static and have single scope. In the case of static, single-scoped events, the PSM can sequentially order the information it receives from the zones based on time.

Single-scoped events may also be mobile, in which case we assume that they typically follow continuous trajectories. So the event is tracked by adjacent zones over a continuous period of time. Tracking the movement of a normal-sized car or tracking human motion are examples of mobile, single-scoped events. As the event moves across the zones, only one of the zones detects and reports the event to the PSZ at any given instant of time. In this case, the PSM can sequentially order the information it receives from the zones based on space and time, because the reports from adjacent zones have different timestamps.

Adjacent Scope: Events with adjacent scope always span multiple adjacent zones at the same time. As a result, the PSZ for an adjacent-scoped event receives concurrent reports about each individual instance of the event from different, but adjacent zones. In the case of adjacent, multi-zonal events, the PSM has to correlate the reports from adjacent zones to infer that they relate to the same instance of the event and not to distinct, single-scoped events. This can be enabled by ordering the reports from the zones based on space, time, or a fine-grained identity of the object, if that is available.

Adjacent, multi-zonal events may be static, in which case the same set of adjacent zones report the event throughout the duration of the event. Monitoring the energy consumption in a building floor with multiple rooms, wherein every room provides the zonal perspective is an example of a static, adjacent-scoped event.

Adjacent, multi-zonal events may be mobile, in which case the PSM receives reports from different sets of adjacent zones at different intervals of time. Tracking the movement of a large truck that spans multiple zones is an example of a mobile, adjacent-scoped event.

Diffuse Scope: Diffuse events differ from single and adjacent-scoped events in that their scope varies with time. An instance of a diffuse event may initially be reported by a single zone, thereby making it a single-scoped event. Alternatively, different instances of the event may be reported by different single zones. However, the event may propagate in time to other adjacent zones and extend its scope from being a single-scoped event to becoming an adjacent, multi-zonal event. Similarly, an event that begins as an adjacent-scoped multi-zonal event may fragment over time to multiple, single-scoped events. Chemical spills, fire, cracks along a pipeline, and seismic activity along the fault lines in an earthquake prone area are examples of diffuse events. In this case, the PSM receives reports from different zones that may or may not be adjacent as the event progresses. Moreover, these reports may be concurrent or distributed in time. Due to the time-varying pattern of information, the PSM needs to store and correlate the information to determine if the reports are causally related.

C. Information Organization in a Zone

When a primary storage zone for an event receives the observations from the zones across the network or from its own zone members (in the case of static, single zonal events), the primary storage manager has to decide how to use the cooperative resources within its zone to store the information so as to enable real-time correlation. We now present some ways in which information can be organized within a zone for the event taxonomy presented in Table I. Each primary storage manager in the network chooses the organization depending on the event that is mapped to its zone.

Temporal Ordering: A PSM can use the resources of its zone members to store information related to an event in temporal order of the occurrence of the event. This scheme assumes that the clock skew in the sensor observations is bounded within an acceptable threshold. In this scheme, every zone member stores information related to an event over a specific time interval. A zone member needs to be awakened only when its corresponding time interval is active. RESTORE uses this organization for most static events. For instance, in the temperature monitoring application, the PSM uses a

different zone member to store the temperature observations reported by the sensors during each 24-hour period. Similarly, images of driving violations captured at different intervals by camera sensors at a traffic intersection are temporally partitioned across the nodes in a zone. In each case, temporal ordering allows new observations to be easily correlated, as and when they arrive, with past events that have been stored. This can then be used to publish notifications about time-based events, as illustrated by the following examples:

- the average temperature in the boiler room for the past 1 hour has been greater than 90 degrees. (single scope, static)
- the building occupancy reported by different offices on the 5th floor of an office building has been less than 10% for the past 1 hour. (adjacent scope, static)

Spatial Ordering: Information in a PSZ can be ordered spatially, based on the zone that reported the event. RESTORE uses this approach for some mobile events. For example, when a PSM receives reports about the presence of a vehicle either from a single zone or concurrently from multiple zones, it uses the resources of different zone members to store the reports received from different zones. The information within the same zone member is ordered temporally. The PSM needs to awaken a zone member only when the incoming report is from a spatial region associated with that zone member. This organization may be used to correlate events within the same zone temporally or to causally relate events across adjacent zones and publish events of interest, as illustrated by the following examples:

- in a surveillance scenario, notify an abnormal activity when more than 10 enemy tanks are detected to be advancing towards a target. (adjacent scope, mobile)
- publish an alert when at least 5 different snipers have been detected in a region in the last hour and they are at most 10 meters from each other. (single scope, mobile)

Identity-based Ordering: Whenever it is possible to distinguish between different instances of the same event, for example using RFID or other means, the storage managers in RESTORE store information related to different instances in different zone members. Information stored in the same zone member may be ordered spatially or temporally. When new reports arrive, only the zone member that is associated with that particular instance of the event needs to be awakened. For example, in the application for monitoring the behavior of autistic patients in a facility, the storage manager stores the sensor observations related to each patient in a different zone member. Within the same zone member, the information is temporally ordered. When new observations about a patient arrive, the storage manager awakens the appropriate zone member and correlates the new readings with the stored data. This correlation can be used to generate a notification, if there is a deviation in the behavioral pattern or if there is an absence of an event for a patient at the expected interval of time.

Multi-resolution Storage: The storage managers can use the hierarchical organization of their zone members to store information at different resolutions. For example, in the WSN debugging application, some parameters are measured more frequently than others. The more frequently reported observations may be stored in zone members that are lower ranked and hence, wake up more frequently, while higher-ranked nodes that wake up less often may be used to store the parameters that are measured less frequently.

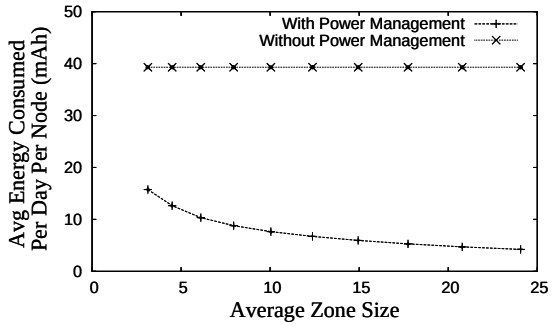

Fig. 3. Variation of energy consumption with zone size

Replication: When information needs to be stored reliably, the PSM can replicate information across the nodes in its zone. Replication can be combined with any of the above schemes to provide reliability. For example, replication is useful in the case of the parking violation application, where loss of information results in loss of revenue. It is also useful in some surveillance applications, where it is hard to reconstruct the events when some critical piece of information is lost. The PSM can either fully replicate the information across all of its zone members or choose the degree of replication depending on the required reliability. The PSM can replicate information in an energy-efficient manner by choosing the zone members having the same ranks as replicas. Since these members have the same sleep schedules, the PSM can download information to all of them simultaneously when they wake up. Thus, nodes that have the same ranks store consistent information.

While replication within a zone provides fault tolerance for independent failures occurring within a zone, it does not handle the case of spatially correlated failures in which a subset of the zone members may fail simultaneously (for e.g. when a truck runs over a section of motes). One way to handle such correlated failures is to use a buddy zone approach in which each event is mapped to a pair of spatially distributed zones. Both of the buddies store the same information. The buddy zone approach trades energy for increased reliability. We have currently not implemented this scheme in RESTORE, because we do not consider spatially correlated failures.

V. Performance Evaluation

The collaborative schemes in RESTORE impact the energy consumption, information availability, and message overhead. We have evaluated these parameters for the replication scheme described in Section IV-C, using a simulator program that we have written. In our experiments, we randomly distributed 10,000 nodes within a 100,000 m^2 rectangular area. We repeated each experiment 30 times with different random

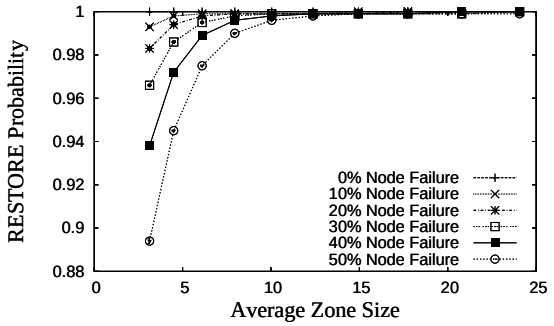

Fig. 4. Variation of information availability with zone size

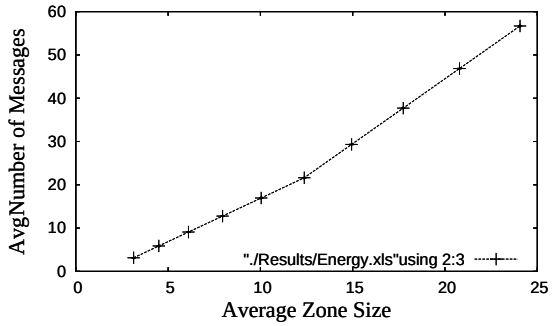

Fig. 5. Variation of message overhead with zone size

number seeds. In this section, we present the results of our experiments.

A. Energy Consumption

In this section, we study the impact of the zone size on the average energy consumption within the network. The zone size is defined as the number of nodes within a zone. To provide sufficient coverage, a zone manager is always active until the next rotation. To ensure reliability, we use the power management scheme in RESTORE to ensure that at least one of the member nodes, in addition to the zone manager, is active at any instant of time. Figure 3 shows that with a larger zone, more energy can be saved by turning off more zone members. For example, when the zone size increases from 5 to 25, we see that the energy consumption in RESTORE reduces by nearly 60%. In addition, Figure 3 shows that the power management scheme in RESTORE achieves significant energy efficiency and reduces the energy consumption by as much as 90% when the zone size is 25, compared to a scheme without power management.

B. Information Availability

In addition to providing energy efficiency, one of the design goals of RESTORE is to improve information availability within the sensor network. Figure 4 shows how the availability varies for different zone sizes and node failure percentages, when information is stored among the zone members using the full replication scheme. We assume that failure of a sensor node occurs due to power loss or an independent hardware fault. The y-axis plots the RESTORE-probability, which is the probability to restore information within a zone in the presence of failures, and is therefore a measure of information availability. It is trivial to conclude that if the percentage of node failures is zero, then RESTORE guarantees that information is always available, regardless of the zone size. However, when the percentage of node failures increases, a relatively large zone size is needed to restore the events with a very high probability. A larger zone size provides greater redundancy and thereby, increases the chances of complete recovery of the stored information. For example, Figure 4 shows that to achieve over 99.9% availability for a failure percentage of 10% and 50%, RESTORE would need a zone size of 7 and 15, respectively.

C. Message Overhead

In our final experiment, we study the impact of the zone size on the message overhead within the network. We specifically consider the message overhead involved in updating the zone member replicas about an event. Figure 5 shows that the overhead increases approximately linearly with the zone size. The reason for the increase is that as the size of a zone increases, there is more variation in the ranks of the zone members. This in turn results in more differentiation among their sleep schedules, as described in Section III-D. Such a differentiated schedule provides higher reliability by increasing the likelihood that at least one zone member, in addition to the manager will be awake at any given time. However, since different replicas awake at different times, the cost of updating the replicas also increases with the zone size. Thus, Figure 4 and Figure 5 reveal a tradeoff between information availability and message overhead. A higher availability is achieved at the cost of incurring higher message overhead.

VI. Related Work

We now compare and contrast RESTORE with some of the related efforts that have addressed the issue of data storage within sensor networks. The cluster-based collaborative storage mechanism presented in [16] organizes the network into clusters in order to store data. It primarily targets applications that do not need to access the in-network data in real-time. However, there is no description of how data is stored within each cluster. In contrast, RESTORE uses different mechanisms to organize data within each zone and in addition, uses this data to correlate events within the network in real-time. The data-centric storage (DCS) scheme [15] stores data by mapping the sensor data to a node in the network using geographic hashing. It focuses more on optimizing the routing of queries to the appropriate storage node in the network, by taking advantage of the routing features of GPSR. RESTORE is complimentary to this scheme in that it focuses more on using the cooperative resources within a sensor network to provide an effective in-network storage mechanism. Geographic hashing is one of many ways in which an event can be mapped to a storage zone in RESTORE. However, if nodes are identified by means other than their locations, then other hashing techniques may be employed

in RESTORE. Moreover, RESTORE does not depend on a specific routing scheme, such as GPSR. In the multi-resolution storage mechanism [17], nodes at different hierarchical levels store information at different levels of resolution. Every node has detailed information about the local events, but has only a compressed view of the events witnessed by the nodes that are below it in the hierarchy. This hierarchical organization can be used to optimize query routing. In addition to multi-resolution storage scheme, RESTORE uses other schemes, such as temporal ordering, spatial ordering, and replication to organize information using the collaborative resources of its zone members.

VII. Conclusions

RESTORE is an overall framework that takes advantage of the collaboration among the sensor nodes to provide in-network event correlation and storage service for sensor networks. RESTORE uses a divide and conquer approach in which it partitions the network into zones, in order to store and correlate information related to multiple events. RESTORE can be used to buffer data over a limited period of time in sensor environments with intermittent connectivity to persistent storage. It can also be used as a basis for a publish-subscribe system involving sensor networks and subscribers on traditional networks. The zone partitioning and collaboration mechanism in RESTORE is a tradeoff between different parameters, such as energy consumption, information availability, message overhead, and the number of distinct types of events that can be stored in the network. We have presented some initial results that show how the zone partitioning scheme influences those parameters when information is replicated across a zone. As part of future work, we plan to implement RESTORE using sensor nodes and conduct a more detailed study of how the above parameters influence the performance of RESTORE for the different storage schemes presented in Section IV-C.

References

[1] T. He *et al.*, "An Energy-Efficient Surveillance System for Wireless Sensor Networks," in *Proc. of MobiSys*, June 2004.

[2] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, "Habitat Monitoring: Application Driver for Wireless Communications Technology," in *Proc. of the 2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, April 2001.

[3] A. Mainwaring, J. Polastre, R. Szewczyk, D. E. Culler, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," in *Proc. of the ACM Workshop on Sensor Networks and Application (WSNA)*, September 2002.

[4] V. Kottapalli, A. Kiremidjian, J. Lynch, E. Carryer, T. Kenny, K. Law, and Y. Lei, "Two-tiered Wireless Sensor Network Architecture for Structural Health Monitoring," in *Proc. of the Intl. Symp. on Smart Structures and Materials*, March 2003.

[5] Zigbee Alliance, available at http://www.zigbee.org/en/index.asp.

[6] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets," in *Proc. of SIGCOMM*, 2003.

[7] T. Luckenbach, P. Gober, S. Arbanowski, A. Kotsopoulos, and K. Kim, "TinyREST: A Protocol for Integrating Sensor Networks into the Internet," in *Proc. of REALWSN*, 2005.

[8] *MICA2 Wireless Measurement System*, Crossbow Technologies, http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0042-04_A_MICA2.pdf.

[9] *STARGATE: X-Scale Processor Platform*, Crossbow Technologies, http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0049-01_A_STARGATE.pdf.

[10] *SkyeRead Mini*, SkyeTek Inc., available at skyetek.com/readers_Mini.html.

[11] *Mica2Dot Series*, Crossbow Inc., available at www.xbow.com/Products/Productsdetails.aspx?sid=73.

[12] 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," in *Proc. of ASPLOS-X*, October 2002.

[13] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," *ACM Wireless Networks*, vol. 8, no. 5, September 2002.

[14] G. Zhou, T. He, S. Krishnamurthy, and J. Stankovic, "Impact of Radio Irregularity on Wireless Sensor Networks," in *Proc. of MOBISYS*, 2004.

[15] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, "Data-Centric Storage in Sensornets with GHT, A Geographic Hash Table," in *Proc. of Mobile Networks and Applications (MONET)*, March 2003.

[16] S. Tilak, N. Abu-Ghazaleh, and W. Heinzelman, "Collaborative Storage Management for Sensor Networks," *Intl. Journal of Ad-Hoc and Ubiquitous Computing*, vol. 1, pp. 47–58, 2005.

[17] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, R. Govindan, and J. Heidemann, "An Evaluation of Multi-Resolution Storage for Sensor Networks," in *Proc. of SenSys*, November 2003.