DISTRIBUTED MECHANISMS FOR ENABLING VIRTUAL SENSORS IN SERVICE ORIENTED INTELLIGENT ENVIRONMENTS

Raja Bose* and Abdelsalam (Sumi) Helal*

*University of Florida, Gainesville, U.S.A, {rbose, helal}@cise.ufl.edu

Keywords: Service Oriented Sensor Networks, Sensor Networks, Virtual Sensors, Sensor Fusion, Fault Tolerance in Sensor Networks.

Abstract

Sensor fusion and virtual sensors have been extensively used in applications requiring the use of high-level abstract information derived from multiple sensory inputs, ranging from defence systems to intelligent environments. In this paper, we present distributed mechanisms for enabling virtual sensors in intelligent environments using service oriented sensor networks (SOSNs). A SOSN brings the concept of service-oriented architecture (SOA) into the sensor network domain, where each sensor is represented as a software service. A SOSN enhances the programmability of sensor networks by decoupling sensor deployment from application development and allows applications to dynamically discover, access and compose sensor services. In our previous work, we had proposed a framework model running inside the service layer of a SOSN for querying virtual data and on-demand creation of virtual sensors. However, the service layer of a SOSN is centralized and this leads to scalability and latency issues in large scale deployments. In this paper, we address the issues of scalability and latency and propose distributed algorithms for in-network creation and execution of virtual sensors. We also present mechanisms for monitoring data quality and provide fault tolerance techniques utilizing sensor-based approximation for coping with the loss of multiple physical sensors. These algorithms not only allow us to reap the benefits of SOSNs but also prevent potential scalability and latency issues. We also show through simulation experiments that our mechanisms require minimal involvement of the service layer and result in significantly lower latency and energy consumption as compared to the centralized stream-based approach.

1 Introduction

Virtual sensor data can be defined as data whose values are a function of a variable number of multiple sensory inputs. This can be due to inherent sophistication of the data type itself or the spatial scope within which it is being queried. An example of virtual data would be weather. There is no single physical sensor which can detect weather since it is an amalgamation of different types of environmental inputs such as temperature and relative humidity. Another example of virtual data can be observed in a situation where a user queries the sensor network for the temperature of a room which has multiple sensors deployed in it. Even though the actual data type is still primitive and can technically be answered by a single sensor, the spatial scope of the query spans across multiple sensors. In both cases, it is more desirable for the user to have the query processing system automatically abstract away the presence of multiple sensors and pretend as if there is a single sensor outputting virtual data. We felt that most of the research in the area of sensor fusion was focused on fusion algorithms for specific application domains and did not adequately address the issue of developing generic mechanisms for enabling sensor fusion, such as dynamic creation and/or discovery of virtual sensor services based on user queries. Hence, in [2] and [3] we proposed a virtual sensor framework for on-demand composition of multiple physical sensors into virtual sensors. We classified virtual sensors into three categories namely, *singleton*, *basic* and *derived* virtual sensors. A *singleton virtual sensor* is the service object representing a single physical sensor. A *basic virtual sensor* is composed of multiple singleton virtual sensors of the same type. Going back to our previous example, if a user queries for the temperature of a room then all the temperature sensors in that room can be composed into a single basic virtual sensor. A *derived virtual sensor* on the other hand, is composed of multiple basic and/or derived virtual sensors of heterogeneous types. The type of data output by a derived virtual sensor is obtained by fusion of multiple primitive or derived sensory inputs and can be considerably more abstract than any of the inputs. The weather sensor is an example of a derived virtual sensor.

Based on queries issued by users, the virtual sensor framework utilizes the service-oriented architecture of SOSNs for dynamic composition of physical sensor services into virtual sensors. However, the centralized service layer where this composition takes place also becomes a source of bottlenecks when the number of sensors involved in the fusion process increases. In order to rectify this situation, we propose in this paper, distributed in-network algorithms for enabling virtual

sensors and monitoring their data quality. Furthermore, we also propose a distributed fault tolerance mechanism for virtual sensors which compensates for failure of multiple physical sensors.

The key contributions of this paper are as follows:

- Scalable and energy-efficient distributed mechanisms for creation and operation of virtual sensors and monitoring their data quality.
- High performance distributed fault tolerance mechanism utilizing sensor-based approximation for compensating for the loss of multiple physical sensors and ensuring fault-resiliency of virtual sensors without requiring intervention from the service layer.

The main goal of the virtual sensor algorithms presented in this paper is to maximize utilization of distributed in-network processing and minimize involvement of the centralized service layer, in order to reduce bottlenecks, latency and average energy consumption of the sensor network.

The rest of this paper is organized as follows. Section 2 goes over some basic details about SOSNs and gives a brief description of mesh networks. Section 3 describes mechanisms for creating basic virtual sensors and associated quality monitoring and fault tolerance algorithms. Section 4 describes mechanisms for creating derived virtual sensors and associated quality monitoring and fault tolerance algorithms. Section 5 presents experimental analysis and comparison of the performance of the distributed techniques presented in this paper, versus centralized stream-based methods. Section 6 covers related work in the field of sensor fusion relevant to our focus area. Section 7 concludes this paper and outlines future work.

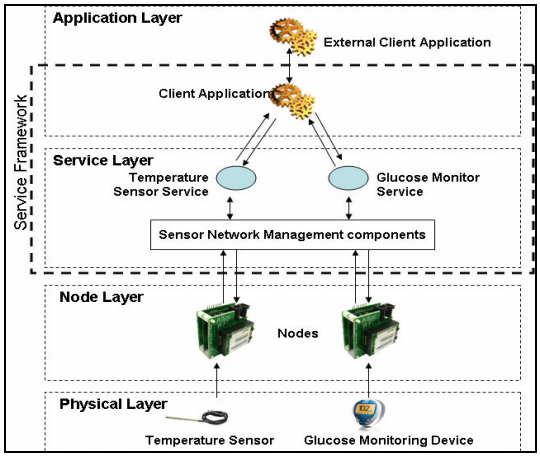

Fig. 1 Architecture of a SOSN

2 Service Oriented Sensor Networks

A service oriented sensor network (SOSN) imports the concept of Service Oriented Architecture (SOA) into the sensor network domain. It represents each of its sensors as a service object in a service framework that allows their dynamic discovery and composition into applications. With the commercial availability of sensor platforms such as Atlas [8], SOSNs have not only been deployed in research facilities such as the Gator Tech Smart House [5] but also by industry [14]. SOSNs are also at the core of efforts such as Service Oriented Device Architecture (SODA) [4] and are rapidly establishing themselves as a practical solution for developing and deploying sophisticated sensor network applications.

A SOSN consists of four basic layers namely, the physical layer, node layer, service layer and application layer, as shown in Fig. 1. The bottom-most layer is the physical layer, which contains a variety of sensors which monitor different aspects of the physical space. The node layer contains a distributed set of hardware nodes which physically interface with sensors deployed in the space. These nodes integrate sensors from the physical layer and export their service representations to the layers above. The service layer, which resides above the node layer, is built on top of a SOA-based framework such as OSGi [15], residing on a central host computer. It holds the registry of software service representations of all sensors and actuators connected to the hardware nodes. It also provides service discovery, composition, and invocation mechanisms for applications to locate and make use of specific sensors. Applications access sensors via their respective service objects. This ensures that the low level operational details of the sensors are abstracted away and application developers are able to use high level method calls to access and control sensors.

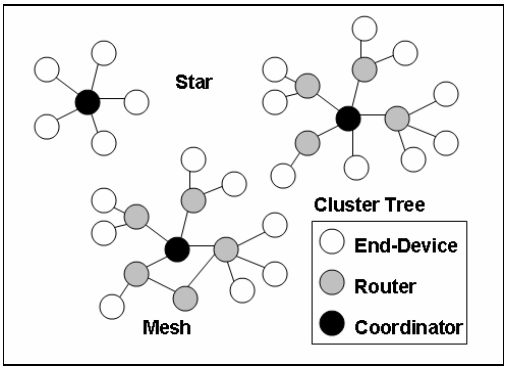

Fig. 2 ZigBee Mesh Network Topologies

2.1 The Atlas Platform

We chose to develop our virtual sensor framework on top of the Atlas Platform. Atlas is a service oriented plug-and-play sensor and actuator platform developed at the University of Florida. It provides the basic building blocks for creating a SOSN and in its most basic incarnation consists of a set of hardware nodes and a collection of management components running inside an OSGi service framework. The Atlas platform hardware consists of a set of nodes which physically interface with the sensors. The Atlas management components run inside an OSGi service framework hosted on a central computer. To cater for dynamically changing deployments, Atlas supports plug-and-play capability at the sensor level. It is able to automatically detect whenever a new sensor (and the node physically connected to it) comes online or an existing

sensor goes offline. This information is also relayed to affected applications via standard OSGi mechanisms.

2.2 Mesh Networking

Algorithms proposed for sensor networks heavily depend on the choice of networking protocol being used. We chose to use nodes with mesh networking for implementing the mechanisms proposed in this paper. In particular, we use Zlink ZigBee nodes from Atmel Corporation running firmware components of the Atlas Platform. ZigBee provides an industry standard mesh networking protocol running on top of the IEEE 802.15.4 stack. As compared to proprietary ad-hoc networking protocols widely used by the wireless sensor networking (WSN) community where all nodes function as routers, ZigBee networks have three classes of devices in them, namely, Coordinator, Router and End-Device. Each ZigBee network has one coordinator which in our case also serves as a gateway to the service layer. The routers are responsible for forming the mesh of inter-node connections and the vast majority of devices are end-devices which do not participate in the network routing process. Only end-devices are allowed to sleep in a ZigBee network and are therefore mainly battery powered whereas the coordinator and routers have to stay on all the time and are typically mains powered devices. Each node is aware of its parent and routers periodically sync with their children which enables them to detect if an existing child node has left the network or a new child node has joined the network. ZigBee also provides reliable delivery of packets using MAC layer ACKs which tells the sender if its message has reached its recipients successfully or not. Fig. 2 shows the topologies possible in mesh networks namely, star, mesh and cluster tree. For this paper, we assume that the network is a cluster tree, that is, it is a hybrid of star and mesh topologies. We also use the term node interchangeably with router and end-device depending on the ZigBee role of that node.

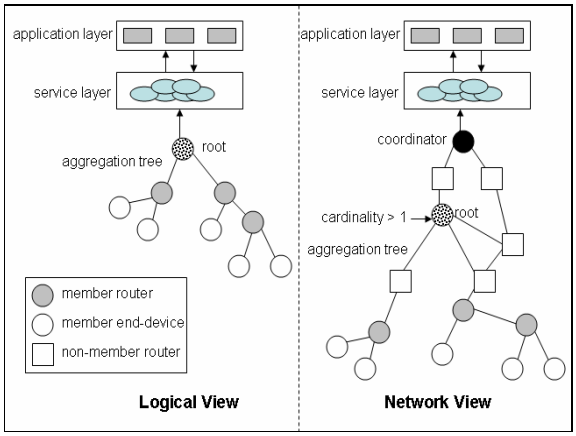

Fig. 3 Logical vs. Network View of an Aggregation Tree

3 Basic Virtual Sensor

A basic virtual sensor's primary role is to aggregate primitive data from multiple physical sensors within a specific spatial scope. This enables it to provide a single output representative of the region it covers while coping with failures of multiple physical sensors. In this section, we present algorithms for creation and in-network execution of basic virtual sensors. We also present mechanisms for data quality monitoring and fault tolerance to help cope with multiple sensor failures.

3.1 Creating a Basic Virtual Sensor

When the user queries for a basic virtual data type, the service layer looks up a knowledge base in the virtual sensor framework to find out which virtual sensor to create. This is done using sensor definitions stored in the knowledge base and using standard SOA mechanisms to compose the required singleton/physical sensor services. We do not cover this process in detail here, but readers interested to learn more about the knowledge base and other components of the virtual sensor framework should refer to [2] and [3]. The service composition takes place in the centralized service layer and the naïve way would be to stream up all the data from each of the physical sensors for centralized aggregation. However this is clearly not scalable when the number of sensors involved is large. Furthermore, streaming up all the data through a multi-hop mesh network will lead to excessive latency and waste of network bandwidth and node energy.

To address the issue of scalability, we push down the process of basic virtual sensor aggregation from the service layer on to the sensor nodes so that it runs in a distributed in-network fashion without requiring centralized processing. The aggregation is done using an aggregation tree overlaid over the topology of the ZigBee network. Such aggregation techniques have been previously proposed in the context of ad-hoc networks [10] and our mechanism follows a similar philosophy. The basic virtual sensor created by the service layer sends a tree-construction packet to the coordinator containing the id of the basic virtual sensor, its sampling rate in epochs and a list of its member sensor nodes represented by their network addresses. Before transmitting the packet, the co-ordinator applies its routing algorithm on the list of network addresses to determine the non-duplicate set of child nodes that it needs to propagate the message to. A hash function which uses network addresses as keys can be used for this purpose. This ensures that each recipient node does not receive duplicate copies of the tree-construction packet as a consequence of multiple member nodes sharing the same routing path. Each router node in the network which receives the packet continues this process. Each router is also able to determine if it is the root of the virtual sensor aggregation tree or not by looking at the set of child nodes that it needs to propagate the packet to. The first router whose set of recipient child nodes has cardinality greater than 1 (Fig. 3, Network View), becomes the root of the tree. This follows from the fact that the message only travelled along a single routing path before reaching this node and now for the first time the route gets split into multiple sub-paths. In order to indicate to other nodes that a root has been selected, this node sets a special bit in the message packet before forwarding it further down the network. In case the

coordinator is the root node it sets this special bit before forwarding the initial packet. Whenever a router or end-device finds its address listed in a tree-construction packet, it removes its address from the list and also extracts the basic virtual sensor id and along with its associated sampling rate. In this manner the aggregation tree is constructed on top of the ZigBee network. However, one must note that the logical view of the tree may be quite different from its network view as we can see in the example shown in Fig. 3. The network view of the tree might contain router nodes whose sensors are not members of the basic virtual sensor and whose only function is to act as a link with different segments of the aggregation tree. This is a consequence of self-organizing networks where nodes connect to their nearest routers based on proximity and signal strength. In case of proprietary ad-hoc routing protocols it might be possible to tweak the algorithms to force nodes to form a network in a certain way. Semantic Routing Trees (SRTs) [13] used in conjunction with link-based parent selection [11] by TinyDB, provide such a functionality. However, this implies that the network has to be re-organized whenever the index attributes change. This is not only expensive but also of limited utility in the context of SOSNs which are typically required to service multiple applications simultaneously.

3.2 Aggregation Mechanism

The aggregation mechanism consists of propagation of partial records starting from the leaf nodes all the way to the root of the tree. Every time a participating end-device samples its sensor, it creates and sends a partial aggregation record to its parent. Each router on receiving a partial aggregation record from its child nodes merges them, updates it if necessary and forwards it to its parent. For example, if the aggregation formula being used was the arithmetic mean then a node would send a partial aggregation record <Sum, Count>, where Sum denotes the partial sum of readings collected and Count denotes the number of sensors sampled for the sub-tree rooted at this node. The root node on receiving the aggregation record calculates the final output, for example, for arithmetic mean it calculates the output as Sum/Count.

3.3 Fault Tolerance

Failure is an integral part of any sensor network especially for large scale deployments. Whenever a member node (and the sensor physically connected to it) fails, the basic virtual sensor loses a data source and this affects its data quality. We propose an in-network mechanism which attempts to compensate for the loss of one or more physical sensors by approximating their readings using other live sensors. Each router in the aggregation tree runs the fault tolerance algorithm and maintains a history of similar behaviour amongst its children who are members of the basic virtual sensor. For example, if a router has 3 member sensors i, j and k as its children then it maintains three variables s(i, j), s(i,k) and s(j,k). If we use Euclidean distance as a measure of similar behaviour then, during any sampling epoch, whenever $|reading_i - reading_j| < \varepsilon$ (where ε is the sensitivity of a sensor), s(i, j) is incremented by 1. Similarly s(i, k) and s(j, k) are also updated as required. The larger the value of these similarity statistics, the better is the quality of sensor-based approximation. If a child sensor node fails, its parent router automatically approximates its readings by choosing a live child node which was observed to be behaving most similar to the failed sensor. This choice is made based on the statistics collected till that point of time. Suppose in our example sensor i fails then another live sensor (call it x) is used to approximate its readings, where x is equal to $\max_y s(i, y)$ with y = {j, k}. The variables used for maintaining similarity statistics are periodically flushed to ensure that they do not get skewed by outdated information. This sensor based approximation mechanism not only has low processing overhead as compared to model-based approximation techniques but also compensates for the loss of sensors without requiring assistance from the service layer. Approximating readings of failed sensors may result in an improvement in the quality of virtual data as compared to having no compensation at all; however, it will still result in a drop in the overall data quality. We cover data quality monitoring in section 3.4.

Failure amongst member nodes of a basic virtual sensor can be of three types:

- End-device failure: The router which is the parent of the end-device detects this situation as a consequence of the ZigBee sync mechanism and executes the approximation algorithm given above.
- Router failure: The parent of the router executes the approximation algorithm while its children on detecting the disconnection, try to re-connect to an alternate router available in the network utilizing the self-healing features of ZigBee. On re-establishing connection, a node sends a message to the new router announcing its basic virtual sensor id and aggregation formula for merging partial records. This router then adds them as child nodes and takes on the responsibility of collecting similarity statistics and executing the fault tolerance algorithm if any of them fails.
- Root of aggregation tree fails: If the root node of the aggregation tree fails, then a new root needs to be discovered. We propose a root discovery algorithm which guarantees that at most one root gets selected even if several nodes are in contention for that role. Each of the failed root's children initiates the discovery process by first transmitting a packet to its other siblings to determine if they are alive or not. After waiting for a certain period of time, it transmits a root-search packet containing its virtual sensor id and number of live siblings. If required, it also contains the sensor ids and network addresses of root nodes of the derived virtual sensors (covered in section 4) which get inputs from this basic virtual sensor. This packet has its destination address is set equal to 0 which is the network address assigned to a coordinator by default. Each router forwards this message to its parent and caches it. However, if any router receives messages

from all the siblings it considers itself as the new root of the aggregation tree. It suppresses further transmission of the root-search packet and sends a new-root message to each sibling declaring itself as the new root node. Since it is possible that due to time delays and synchronization errors, multiple nodes can declare themselves as the root of the same tree, each router which considers itself the root node also suppresses forwarding of any new-root messages. This ensures that only the node with the shortest network path to all the siblings becomes the new root. In case the basic virtual sensor is a member of a derived virtual sensor, the new root node is also responsible for linking up with the root of the derived virtual sensor.

3.4 Monitoring Data Quality

The fault tolerance mechanism discussed in section 3.3 approximates readings of failed sensors by using readings from other live sensors. This will naturally affect the quality of data being output by the virtual sensor. In order to monitor data quality, each reading originating from a basic virtual sensor is associated with a virtual sensor quality indicator ($VSQI_{BVS}$) value. This value provides a measure of the virtual sensor's data quality. $VSQI_{BVS}$ is a function of the number of live sensors contributing readings to the basic virtual sensor and the number of failed sensors whose readings are being approximated using live sensors by the fault tolerance mechanisms described in section 3.3. The formula for computing $VSQI_{BVS}$ is given by:

$$VSQI_{BVS} = \frac{(N_C + \sum_{s \in S_F} g(\Delta t_s) W_s)}{N} \; ; \; VSQI_{BVS} \in [0, 1]$$

$VSQI_{BVS}$ has a maximum value of 1, if all member sensors are alive and 0 if all the sensors are dead. N is number of live physical sensors which were originally members of the basic virtual sensor. N_C denotes the number of physical sensors which are currently alive. S_F is the set of failed sensors. Δt_s is the time elapsed between the start up and failure of a failed sensor 's'. W_s is a weight associated with sensor 's' where, $W_s \in [0, 1)$. W_s is computed by the fault tolerance mechanism, as the probability that the live sensor being used for approximation behaved similar to the failed sensor. The term $g(x) = 1 - e^{-x/a}$ is a time dependent weight function associated with W_s, where '*a*' is a constant greater than 0. The value of '*a*' determines how fast g(x) converges to 1 and depends on the sensors and their deployment environment. If W_s is calculated based on observations taken over a long time interval, then that value of W_s will be given more weight than say, a value of W_s which has been calculated based on observations taken over a shorter period of time.

The value of $VSQI_{BVS}$ is also computed using the partial aggregation method described in the section 3.2. A partial aggregation record for $VSQI_{BVS}$ is given by $<Q_P>$. Q_P denotes VSQI information, for the sub-tree (call it S_T) rooted at this node. The Q_P term is given by $N_L + \sum_{s \in S_F \cap S_T} g(\Delta t_s) W_s$, where N_L denotes the number of live sensors in S_T and g(t), t_s and W_s are as defined above. The final $VSQI_{BVS}$ value is calculated at the root as Q_P/N, where N is as defined above. For practical purposes, the partial aggregation records for sensor readings and $VSQI_{BVS}$ are not transmitted separately but instead are merged into a single record: <Sum, Count, Q_P>.

4 Derived Virtual Sensor

The role of a derived virtual sensor is to apply a fusion function on inputs from multiple heterogeneous virtual sensors (whether basic or derived) to produce more sophisticated and abstract types of data. This enables it to answer queries on data types which cannot be processed by the other virtual and physical sensors such as weather and complex events like activities of daily living.

4.1 Creating a Derived Virtual Sensor

Intuitively a derived virtual sensor should be residing solely inside the service layer since it is not directly bound to inputs originating from any physical sensor. However, this will lead to scalability and latency issues due to SOSNs having a centralized service layer. In order to address these issues, we propose a mechanism which pushes down the operations of a derived virtual sensor on to the distributed mesh network of nodes. When a user queries for a derived data type, the service layer looks up the definition of the associated virtual sensor from the knowledge base. It then composes together the different member virtual sensor services into a derived sensor service and initiates the process of creating an in-network aggregation tree. First the derived virtual sensor service obtains the network addresses of the root nodes of each of its member sensors. This is possible due to the fact that the packets containing the output of any virtual sensor will have its source address equal to that of the root node of that sensor's aggregation tree. Then a sensor-construction packet containing the list of these network addresses is dispatched to the ZigBee network via the coordinator. In a manner similar to the one described in section 3.1, each router applies its routing algorithm on the list of network addresses to determine the non-duplicate set of child nodes that it needs to propagate the message to. The first router whose set of recipient child nodes has cardinality greater than 1, becomes the root of the aggregation tree for the derived virtual sensor. This node becomes responsible for applying sensor fusion to the data originating from sensors which make up the derived virtual sensor.

4.2 Aggregation

The aggregation of readings in a derived virtual sensor is pretty straightforward, with each member sensor sending its output to the root node which computes the final output and transmits it to the service layer via the coordinator. Unlike a basic virtual sensor which aggregates data from homogeneous sources, a derived virtual sensor has to operate on multiple heterogeneous inputs. Hence, the

partial aggregation techniques used for basic virtual sensors are no longer valid in the context of derived virtual sensors.

4.3 Fault Tolerance

By virtue of its definition, a derived virtual sensor cannot be too resilient when it comes to tolerating failures. We take the position that if a member sensor fails completely then the derived virtual sensor should cease to operate due to lack of one of its inputs. Since we are not focusing on a particular application scenario for derived virtual sensors, we are forced to adopt such a pragmatic stance. It is entirely possible that in some application domains derived virtual sensors may be able to operate without the presence of one or more inputs. However, in the interests of providing a generic framework for virtual sensors, we decided to adopt an 'all-or-none' approach in the context of derived virtual sensors. We hope to rectify this as part of future work. The following are the possible failures that can occur in a derived virtual sensor which have not been discussed in the previous sections:

- Root of derived virtual sensor fails: The root of each member virtual sensor sends a root-search message addressed to the coordinator. Any router receiving messages from all the root nodes of the members considers itself the new root of the derived virtual sensor and sends a new-root message to the sender nodes. It also suppresses further transmission or root-search and new-root messages as described in section 3.2.

4.4 Monitoring Data Quality

A derived virtual sensor may be composed of multiple heterogeneous basic or derived virtual sensors; hence, monitoring the performance of a derived virtual sensor requires a slightly different approach than that of a basic virtual sensor. The $VSQI_{BVS}$ formula gives the probability that the approximated output of a basic virtual sensor matches its expected output, if it was fully functional. Following the same trend of thought, we defined VSQI associated with a derived virtual sensor as the product of VSQIs of its member basic and derived virtual sensors, given as follows:

$$VSQI_{DVS} = \prod_{1 \le i \le n} VSQI_{BVS_i} \times \prod_{1 \le j \le m} VSQI_{DVS_j};\ VSQI_{DVS} \in [0, 1].$$

The above formula reflects the fact that in case of a derived virtual sensor there is no concept of compensating for the loss of one of its member sensors. If one of the member basic or derived virtual sensors fails then its parent derived virtual sensors also fail.

5 Experimental Analysis

In this section we analyse the performance of the virtual sensor mechanisms presented in this paper, using both simulated and real-world data. We conducted two sets of experiments, one for evaluating fault tolerance and data quality monitoring and the other for analysing performance of our methods as compared to centralized techniques, in terms of latency and energy consumption of the sensor nodes.

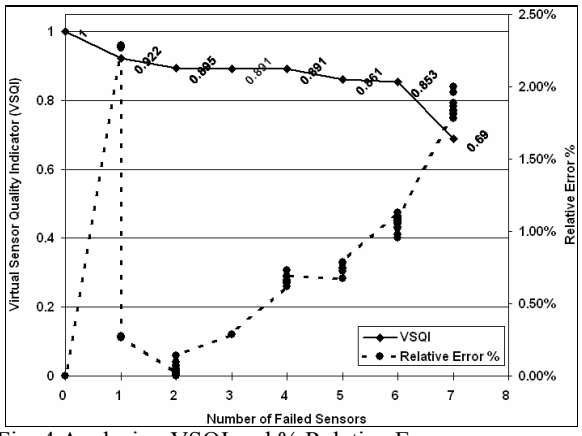

Fig. 4 Analysing VSQI and % Relative Error

5.1 Data Quality Monitoring and Fault Tolerance

The goal of the first set of experiments was to measure the effectiveness of the fault tolerance algorithm and data quality monitoring covered in sections 3.3 and 3.4. We were interested in evaluating whether the sensor-based approximation method was an effective way of compensating for failed sensors and if the virtual sensor quality indicator (VSQI) accurately reflected the effect of the approximation scheme on overall data quality. In order to achieve this, we simulated the operation of a basic virtual sensor and monitored its VSQI value and % relative error. The relative error value was computed as the deviation of the actual output of the virtual sensor (based on aggregation of live and approximated sensor readings) from the expected output (based on aggregation of readings from all live sensors). For this experiment, we decided to use a real-world data set from Intel Research Lab, Berkeley [16]. This data set consists of temperature, humidity and light level readings collected from 54 nodes over a period of one month. Out of the 54 sets of data, only 40 were short-listed as the remaining ones had incomplete sets of readings. Each sensor log was also sorted to ensure that all the readings were ascending order of their time stamps. Based on sensor location information provided in the documentation, we created basic virtual sensors composed of 8 physical temperature sensors which used arithmetic mean for aggregation. During simulation, sensor failures were injected at periodic intervals and the VSQI value and % relative error logged for each epoch of sampling. From Fig.4, we can see the sensor based approximation mechanism works reasonably well even when the number of failed sensors is large. The % relative error increases as the number of sensors fails but it never exceeds 2.5%. Relative error can be heavily dependent on the specific data sets and the deployment configuration of physical sensors. However, sensors deployed in realistic settings in close proximity, will usually tend to exhibit spatial correlation, leading to smaller and bounded relative errors. We also observe that the VSQI value is inversely proportional to the % relative error and decreases as the percentage of approximated sensors increases. We also note that the VSQI value is

closely correlated with the actual relative error and thus, is able to provide an accurate estimate of data quality to users.

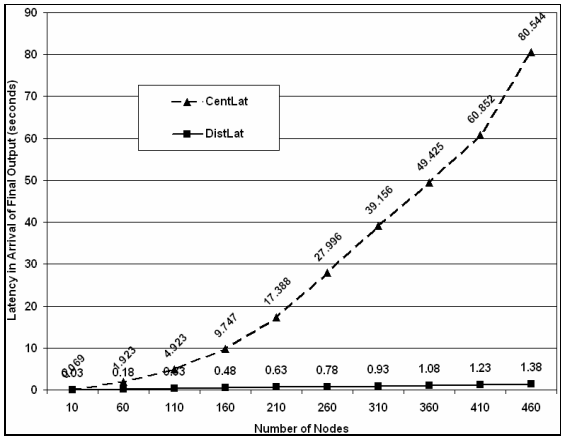

Fig.5 Comparing Latency in Arrival of Final Output

5.2 Latency of Data Arrival and Power Consumption

The second set of experiments was conducted to evaluate system performance in terms of latency in arrival of virtual sensor output and energy consumption of nodes per epoch of sampling. We compared the performance of distributed mechanisms proposed in this paper versus the mechanism where virtual sensors are executed exclusively inside the centralized service layer of the SOSN and perform centralized aggregation on data streams. The Global Sensor Network (GSN) [1] is example of a virtual sensor framework which utilizes this form of centralized streaming approach for implementing virtual sensors. For distributed tree-based aggregation, we used a simulator to randomly generate aggregation trees overlaid on top of a mesh network. The simulator takes as input the number of sensor nodes which are members of a virtual sensor and for each iteration, randomly generates an aggregation tree of nodes. To simulate the overlaying of the tree on top of a mesh network, it randomly assigns to each node one of the following roles: (1) end-device which is a member of the virtual sensor; (2) router which is a member of the virtual sensor; and (3) router which is not a member of the virtual sensor (that is, it only links different segments of the aggregation tree). To calculate latency of virtual sensor output, we assume a node takes 0.003 seconds to receive or transmit a packet (based on a packet size of 100 bytes and ZigBee transmission bit rate of 250 Kbps). For computing energy consumption of nodes we obtained power consumption data from Atmel for their Zlink RCB nodes. We use a power consumption cost model where each member end-device and router's cost is calculated as the sum of processing, sensor sampling and transmission costs. Additionally each member router also incurs the cost of receiving messages. On the other hand, each non-member router's cost is simply the cost of receiving and transmitting messages.

From Fig. 5, we observe that in case of centralized aggregation (denoted by 'CentLat'), the latency increases significantly as the number of sensors increases. This is due to the fact the each sensor reading has to traverse multiple hops all the way up the tree in order to reach the service layer and the aggregation operation cannot take place till readings from all the member sensors have streamed up. The distributed aggregation tree mechanism (denoted by 'DistLat') on the other hand exhibits extremely low latencies of approximately 1 second or less, even when the number of sensors is large. This can be explained by the fact that a reading from each sensor node only travels over 1 hop to its parent aggregation node where it is merged with other partial aggregation records. Hence, readings from each sensor do not need to be transmitted over multiple hops all the way to the service layer thereby, significantly reducing latency of virtual sensor output. Moreover, due to its lower utilization of network resources, it reduces the processing and networking costs on intermediate routers, which translates into significant reduction in overall energy consumption. From Fig. 6, we observe that for the centralized method (denoted by 'CentEnergy'), the total energy consumed by nodes for each epoch of sampling dramatically increases as the number of member sensors increases whereas for the distributed mechanism (denoted by 'DistEnergy') there is no significant increase in energy consumption. In fact the total energy requirements per epoch for the distributed mechanism is anywhere from 90-98% less than the centralized method. This implies that the distributed mechanism is not only more scalable but is also more energy efficient as compared to the centralized method.

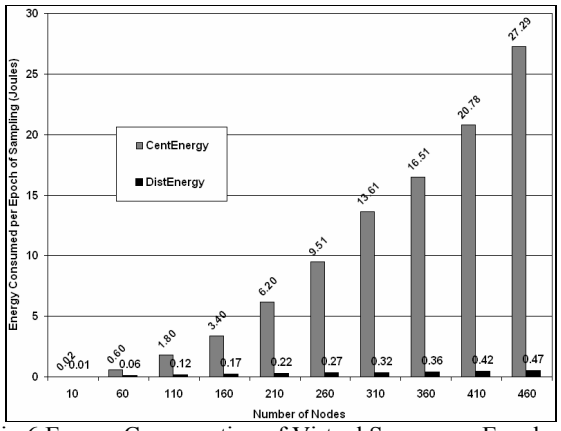

Fig.6 Energy Consumption of Virtual Sensor per Epoch

6 Related Work

Sensor fusion has always been an actively researched topic in the sensor network community. The majority of publications in this area have tended to focus on developing algorithms for sensor fusion [6] [12] and application specific virtual sensors. However, there are some groups of researchers who have looked at virtual sensors from a system perspective and developed mechanisms for creation and deployment of virtual sensors. Kabadayi et al. [7] describe a middleware for virtual sensors where heterogeneous sensor readings are aggregated to generate higher abstractions of data. However, they only look at aggregation aspects of virtual sensors without addressing reliability and availability issues. Furthermore, their proposed middleware seems to

be too rudimentary and can easily be superseded by native mechanisms available in service oriented architecture (SOA) frameworks such as OSGi which form the basis of SOSNs. Lewis [9] defines virtual sensor as a combination of a physical transducer along with data signal processing (DSP) elements for reliable data estimation. Both Kabadayi and Lewis have considered different aspects of virtual sensors but none of them have proposed practical distributed mechanisms for execution, data quality monitoring and fault tolerance. The Global Sensor Network (GSN) [1] is a project at EPFL, Switzerland which aims at providing a unified framework to users for accessing real and virtual sensors. Their work comes closest to our concept of a virtual sensor framework which enables on-demand creation and execution of software sensors. However, GSN follows a stream-based centralized approach for implementing virtual sensors where physical sensors are required to stream up all their readings to the central host where virtual sensors are executed. This not only leads to increased overhead at the central host but also leads to high latency and power consumption. In contrast, even though we also have a centralized service layer in our architecture, we take a distributed acquisition-based view of the internal operations of a virtual sensor and propose mechanisms which reduce latency and improve fault tolerance of virtual sensors and lower power consumption of the sensor nodes, without sacrificing the advantages of providing a user with a unified framework for accessing virtual sensors.

7 Conclusions and Future Work

In this paper, we proposed distributed mechanisms for in-network execution of virtual sensors and provided techniques for monitoring their data quality and improving their fault tolerance. We also validated the usefulness of our approach through experimental analysis using simulations and real-world sensor data and showed that our methods provide superior performance in terms of fault tolerance, low latency and energy consumption.

Finally, we conclude this paper with a list of our on-going and future research work involving virtual sensors:

- Developing fault tolerance mechanisms for derived virtual sensors.
- Using virtual sensors in conjunction with service selection and replacement mechanisms for ubiquitous service composition in pervasive spaces. Whenever a virtual sensor fails, the service selection and replacement mechanisms will kick in to either locate another suitable virtual sensor or instantiate a new one.
- Applying virtual sensors in a smart home environment for unencumbered monitoring of conditions such as obesity and diabetes of its residents.

Acknowledgements

The authors would like to thank Atmel Corporation for providing technical assistance for their Zlink line of ZigBee products. We would like to thank Ivan Zellner for providing technical data on power consumption of Atmel ZigBee nodes. We would also like to thank Hen-I Yang for contributing both ideas and code for experimental evaluation of the fault tolerance mechanisms described in this paper.

References

[1] K. Aberer, et al., A middleware for fast and flexible sensor network deployment, *Proceedings of 32nd International Conference on Very Large Data Bases (VLDB)*, 2006.

[2] R. Bose, et al., Virtual Sensors for Service Oriented Intelligent Environments, *Proceedings of the 3rd IASTED International Conference on Advances in Computer Science and Technology*, 2007.

[3] R. Bose, et al., Fault-resilient Pervasive Service Composition, Book Chapter in *Advanced Intelligent Environments*, H. Hagras, Editor, Springer Verlag. To appear 2008.

[4] S. de Deugd, et al., SODA: Service-Oriented Device Architecture, *IEEE Pervasive Computing*, 5(3), 2006.

[5] A. Helal, et al., Gator Tech Smart House: A Programmable Pervasive Space, *IEEE Computer*, March, 2005.

[6] R. Jiang and B. Chen, Fusion of Censored Decisions in Wireless Sensor Networks, *IEEE Transactions on Wireless Communications*, 4(6), 2005.

[7] S. Kabadayi, et al., Virtual Sensors: Abstracting Data from Physical Sensors, *Proceedings of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2006.

[8] J. King, et al., Atlas: A Service-Oriented Sensor Platform, *Proceedings of 1st IEEE International Workshop on Practical Issues in Building Sensor Network Applications*, 2006.

[9] F. Lewis, Wireless Sensor Networks, *Smart Environments: Technologies, Protocols and Applications*, John Wiley, 2004.

[10] S. Madden, et al., TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks, *Proceedings of 5th Annual Symposium on Operating Systems Design and Implementation*, 2002.

[11] S. Madden, et al., TinyDB: An acquisitional query processing system for sensor networks, *ACM Transactions on Database Systems*, 30(1), 2005.

[12] R. Olfati-Saber and J. Shamma, Consensus Filters for Sensor Networks and Distributed Sensor Fusion, *Proceedings of IEEE Conference on Decision and Control and the European Control Conference*, 2005.

[13] A. Woo and D. Culler, A transmission control scheme for media access in sensor networks, *Proceedings of 7th Annual International Conference on Mobile Computing and Networking*, 2001.

[14] Stepstone Project, IBM press release, http://www-03.ibm.com/press/us/en/pressrelease/21926.wss.

[15] The OSGi Alliance, http://www.osgi.org.

[16] Sensor Data Set from Intel Research Lab Berkeley, http://db.csail.mit.edu/labdata/labdata.html.