Quality-Aware Sensor Data Management

Zhijing Qin, Qi Han, Sharad Mehrotra, Nalini Venkatasubramanian
Colorado School of Mines and University of California, Irvine

1 Motivation

Continuing advances in computational power, radio components, and reduction in the cost of high-performance processing and memory elements has led to the proliferation of portable devices (e.g., intelligent sensors, actuators, and sensory prosthetics) with substantial processing capabilities. By providing the ability to monitor phenomena in close proximity with multihop wireless communication (enabled by on-board radios) and collaborative in-network computation (enabled by on-board processing), embedded networked sensors are able to achieve accuracy, latency, and coverage in monitoring real-world cluttered environments that a small number of complex sensors cannot. Such devices are rapidly permeating a variety of applications domains such as avionics[48], environmental [75], structural sensing[20], telemedicine[67], space exploration[44], and command and control[4]. Popularly used wireless sensor devices include Mica motes from Crossbow, Tmote Sky from Moteiv, the MKII nodes from UCLA, and SunSpot from Sun.

In this chapter, we provide a data management perspective on large scale sensor environments applications posing non-functional requirements to meet the underlying timeliness, reliability and accuracy needs in addition to the functional needs of data collection. Consider the following use-case of sensor networks in the domain of public health and safety where one needs to monitor chemical and biological contaminants in soil, ground water, streams, etc. The application scenarios range from long-term monitoring for slowly evolving disasters (e.g., leakage of industrial contaminants into ground water) to detection of sudden disasters (e.g., bio-chemical terrorist attack). Since the phenomena that we seek to monitor and respond to may involve large-scale regional spread (tens to hundreds of kilometers), we need methods that will allow scaling of today's systems to large scale deployments. A fundamental challenge in such situations is the ability to handle the explosion of sensor data in such networks. This explosion in data occurs either due to scaling of the network or due to increased data generation by highly capable and 'media-rich' nodes. Since data movement costs precious network resources (e.g., energy, storage, bandwidth), data management must be part of the overall system and software architecture.

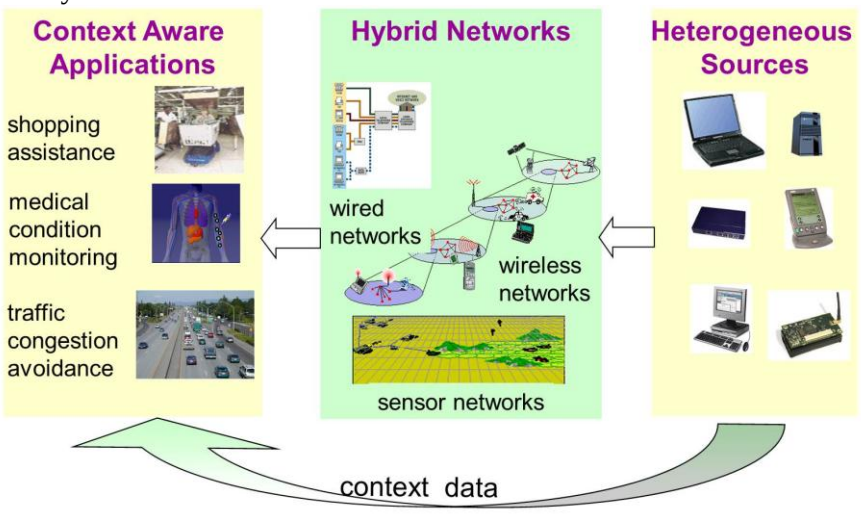

Figure 1. Emerging applications in pervasive sensing environments

In circumstances where the presence of a human in the loop is either too expensive or too slow, embedded sensing systems must also respond autonomously and flexibly to unanticipated combinations of events at run-time. These systems are networked to form long-lived "systems of systems" that must run unobtrusively and autonomously, shielding operators from unnecessary details, while simultaneously communicating and responding to application-critical information at heretofore infeasible rates. An important design challenge for such complex distributed computing systems is to satisfy performance and reliability constraints while ensuring efficient exploration through a very large space of device and network operational choices. This process, unless addressed by the software development and execution architecture, is likely going to be manually driven thus limiting the application potential due to prohibitive application development and interoperability across the sensor networks.

There are several reasons why a data management perspective on sensor networks is increasingly important. Given the proliferation of sensing capabilities, we are observing a transition from a device centric view of sensor systems to an information centric view where sensors generate large amounts of data that is stored and processed to generate higher level information for applications. The notion of what is a sensor is changing (above and beyond mote-like devices) to accommodate multimodal sensors, human sensors, smartphones, all of which are capable of capturing, storing, processing, and communicating the sensed data. Managing heterogeneous sensor data and extracting information from it is key to many real world applications – e.g., cyberphysical systems[56].

Networks composed of large numbers of wireless sensors present significant challenges to the designers of data management systems which aim to incorporate data produced by such networks. We envision future generations of sensor networks that would take input from many remote sensors, and provide geographically-dispersed operators with the ability to interact with the collected information and to control remote actuators. As shown in figure 1, possible sensor sources may be smart phones, lap top, usb sensor, etc. Sensor data generated by those sources is collected via heterogeneous network (e.g. wifi, cellular, Ethernet, etc). A bunch of context aware applications use those data for various kinds of purposes. Ideally, user applications will interact with sensor-generated data in a high-level language appropriate to their application domain. This will then be translated to data management primitives, e.g., queries in an SQL-like language which will then be evaluated by the data management system, by some appropriate processing strategy. Such strategies should take into account (a) the quality requirements of applications, e.g., the level of precision requested for an average temperature value, (b) the underlying observed physical phenomena, whose properties may suggest a processing strategy, and (c) the characteristics and current state of the sensor network, e.g., its scale, degree of heterogeneity, processing/memory/energy capabilities of sensors.

In this chapter, we will first illustrate the different kinds of applications likely to use sensor data, and in particular, real-time monitoring, e.g., pollutant tracking using chemical sensors, data archival for future use, e.g., the recording of natural phenomena for future analysis by qualified scientists, and forecasting which extrapolates into the future, as in e.g., the prediction of the likely trajectory of a hurricane. We will characterize applications' non-functional needs in terms of quality of service and quality of data, describe how to specify these application needs and how to translate high level application needs to requirements for sensor data. We will further develop a model of a sensor system and sensor data. We will evaluate alternative architectures which can achieve application goals, spanning the spectrum from traditional fully-centralized approaches whose simplicity is counterbalanced by their inability to leverage sensor capabilities to fully decentralized ones which lack global scope but

may be resilient to failures. Finally, we will examine how alternative evaluation strategies, given an architectural choice, affect optimization goals such as timeliness, energy efficiency and resilience to faults.

2 A Landscape of Distributed Sensor Applications

Often, sensor-based systems are built with narrow application goals in mind. Consider for instance a simple application using chemical sensors to track and report pollutants in water streams periodically. We anticipate that as sensor network infrastructures become more sophisticated, they will have to accommodate several concurrent applications, some of which may have conflicting requirements in terms of timeliness, reliability and data accuracy. It is important for future sensor systems to accommodate alternative application types, and ensure that their conflicting requirements mesh with each other gracefully.

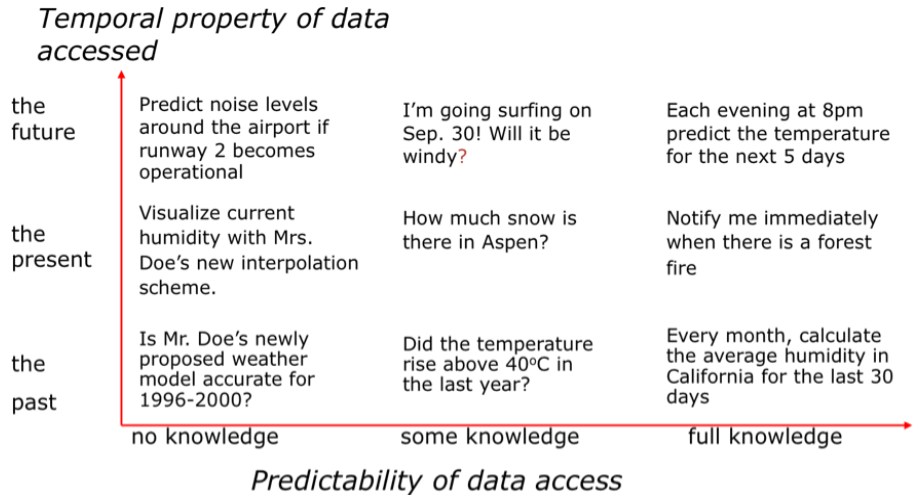

Figure 2. Sensor Application Landscape

In order to cover a wide spectrum of sensor applications, we first distinguish between different application types based on the temporal aspect of sensor data (Figure 2): archival applications focus on historical (past) data, e.g., in order to detect patterns over time and build time-varying models. Real-time data collection is not critical here, but high quality and reliable archival of sensor-generated data is; monitoring applications are interested in current sensor values such as intrusion detection systems; and forecasting applications are interested in predicting future sensor values, where human operators involved in decision making processes can avail of information about trends in sensor values. An example of this is route selection in intelligent transportation applications, where techniques to predict traffic conditions (estimated from traffic monitoring devices) among various route candidates can help in reducing travel times and latencies.

We can also classify applications based on the pattern of access to sensor data (Figure 2). In some cases, e.g., pollutant tracking, the target application is known *before* sensor data is collected. Thus, one can set up the data collection framework in a manner that is optimal for the particular application. In particular, if multiple such applications co-exist, as in e.g., multiple continuous queries then one could exploit the overlap in applications' data needs to optimize the amount of data communication. In other cases, the application type (e.g., requests for aggregate pollutant levels over spatial grids) is known, but not the query instances, which

are unknown and arrive in an *ad-hoc* manner. Finally, there is the case where the specific application type may not be known beforehand. For example, one might instrument a network of traffic monitoring sensors for the purpose of monitoring speed levels in different segments of the road. However, sensors in such an infrastructure may potentially be used in the future for very different applications: e.g., to control traffic signals, or disseminate optimal routes in real-time to drivers.

The basic functionality of the sensor network is to sense, capture, and communicate data to answer application queries. The ability to use sensors in a wide range of applications has expanded the way sensing (and sensors) are used today. Traditionally, sensor data is viewed as being small – it was designed to encompass parameters such as temperature and smoke density; more recently, it has expanded to incorporate diverse and richer context data from cyberphysical systems, e.g. smartspace video surveillance data and power grid voltmeter. Further, sensor data can also be data captured by smart phones either from user input or from built in sensors such as accelerometers. The sensor application landscape discussed above is valid regardless of data sources. Increasingly, it has become more important to enable the functionality of the sensor network while addressing the tradeoffs and non-functional needs of various sensor applications. By non-functional needs, we mean applications' timeliness, reliability, and data accuracy needs. This can be achieved by taking advantage of a few key intuitions and building upon the wealth of techniques developed over the years in real-time systems, fault tolerant distributed computing and dynamic data management.

Consider the following example of sensing for emergency response at a crisis site. (1) Medical sensing: medical sensors can be deployed on a patient's body to monitor health related parameters. These data is collected via wireless personal area network for the doctors to monitor the patient's health status in real time. (2) Deployed environmental sensors at crisis site: smoke sensor can detect fire in a building, and it can also work with camera sensors to help determine a rescue route in a timely manner. The above example illustrates the utility of sensing in emergency response applications. The example also illustrates the need for timeliness, prioritization and fault tolerance in sensing. It illustrates that the networked sensor environment in practice is highly heterogeneous. It illustrates the need for careful and judicious utilization of scarce communication resources. In fact, it also illustrates that in the immediate aftermath of a disaster, energy efficiency of the sensors monitoring patients and environmental phenomena is of less importance than the timely and accurate communication of sensed information. It illustrates that these needs shift over time where enhancing sensor lifetimes through careful energy management at the sensor node is critical to monitor remnant or new phenomena in the days and weeks after the disaster. It illustrates that the sensing data has changing needs which will bring about different desired tradeoffs. For example, before a fire starts, smoke sensors only need to report when the smoke density exceeds a specified threshold. While during the fire, smoke sensors need to periodically report the smoke density with a high frequency. The need of the sensing changes from reliability to timeliness.

The first step towards the goal of supporting sensor applications with different quality needs is to fully understand the diverse needs of monitoring, archiving or forecasting sensor applications. This requires a careful analysis of a wide range of performance requirements, which define the extent to which performance specifications such as timeliness, reliability, and accuracy may be violated. We explore applications' quality requirements from several dimensions.

- *QoS-timeliness* may be specified in the format of periodicity, deadline, or a certain relative order of different tasks. For instance, in the subsurface contaminant tracking scenar-

io, conductivity readings will only be collected after the temperature readings suggest the existence of a potential plume. Current sensor systems support a basic form of time constraints - data collection frequency as in TAG [60] and Cougar [26]. It is worthwhile to investigate other timing notions used in temporal, real-time databases [68, 76, 83] and active databases [17, 81], and then develop a variety of timing semantics appropriate for distributed sensor environments.

- *QoS-reliability* is most commonly defined as the percentage of nodes participating in the collection among all the nodes in the sensor network [36, 79], or as a set of nodes that cover the entire sensor network [72]. In addition to supporting these reliability specifications, we have developed more informative reliability metrics with well-defined semantics. For instance, the recall metric used in information retrieval may be used to indicate the desired completeness of the answer set, if the application is gathering all the readings that meet certain conditions [51]; reliability can also be specified as tolerable thresholds on "false-alarm" or "missed-event" probabilities (i.e., bounds on detection or estimation accuracy) [47].
- *QoD* (Quality of Data) desired from the sensing substrate may be imposed on individual sensor values, or on an answer computed over readings from a set of sensor reports. QoD requirements may be specified as desired data freshness, absolute or relative accuracy bounds. For instance, an application may be satisfied with a report that is off the true value by ±5 [37] (i.e., absolute accuracy) or by 10% [65] (i.e., relative accuracy).
- *Cost* is simply defined as energy consumption since energy is the most stringent resource constraint in sensor networks.

A sensor application may have requirements for one of the multidimensional parameters (QoS, QoD, Cost), or a combination of them with different preferences towards different constraints. These preferences will be used to guide future monitoring, sensing and collection plans. In practice, it may be very difficult (if not impossible) to satisfy all the specified requirements simultaneously, to reach the multi-constraint optimal point in reality given the dynamic network conditions and severe resource constraints in the sensor network. Therefore, we allow applications to specify their preference towards different constraints.

One of the important sensor application programming methodologies that have emerged in recent research is a separation between application logic and the sensor data acquisition that drives the application. Examples of such efforts include the Cougar [26] and Berkeley TinyDB projects [61] that have promoted the view of sensors as data producers and support declarative database languages such as SQL, suitably extended and modified, as a way for applications to specify their sensor data needs (e.g., ACQL [63] developed as part of TinyDB supports event-based queries, lifetime queries, etc.) which are of specific interest in sensor environments. An important research direction is extensions of both syntax and semantics of such languages to enable specification of not just their functional but also non-functional needs of applications. A step in that direction is TiNA [80] that extended ACQL to support applications with data accuracy requirements. However, constructs to specify other aspects including timeliness and reliability constraints are missing. Developing such a language requires a careful analysis of diverse needs of sensor applications ranging from monitoring, archival, prediction, actuation, etc.

To ensure end-to-end support for applications with multiple non-functional needs, it is essential to provide a channel for applications to specify what they need and in what manner. First, we will need a high level declarative language, using which applications can specify both functional and non-functional needs. Second, we will also need a compiler that takes in an

application program and generates executable code. The specification of high level application needs will be either translated to data needs or tradeoffs to guide sensing and collection planning. We next discuss why existing work in the literature cannot be directly used and suggest possible ideas to address this issue.

Specification of application needs: Sensor applications are often interested in data of certain type (e.g., temperature, or humidity data), and they are not interested in where and how the data is obtained. TinyDB uses a modified version of traditional SQL, focusing on issues related to when and how often data are acquired [63]. It supports several new features that are unique to sensor applications, such as event-based queries, lifetime-based queries and actuation queries, etc. Worthy of mention is the `LIFETIME' clause, which provides an intuitive way for users to reason about power consumption. This language has been furthered extended for continuous aggregate queries by TAG [60] and Cougar [26] where sampling period is specified in a clause beginning 'EPOCH DURATION' or 'EVERY'. Building on top of these, TiNA [80] introduces the 'TOLERANCE' clause in the query specification, using which users can specify the temporal coherency tolerance for the query. e.g., if the user specifies the tolerance to be 10%, the sensor network will only report sensor readings that differ from previous reported readings by more than 10%.

It is necessary to further extend the query language and provide other clauses for users to specify other non-functional needs in addition to accuracy and energy efficiency: timeliness, reliability.

- Specifying accuracy: The concept of relative accuracy introduced by TiNA allows a uniform and easy to understand definition of user tolerance on heterogeneous data sources, where the domain of sensed values is different from one sensor to another. However, there do exist other applications that prefer to specify their absolute tolerance (the maximum deviation of sensor reports from actual measurements). Therefore, we should add the support for absolute accuracy tolerance.
- Specifying timeliness: Specification of time constraints has been used extensively in real-time databases and active databases, where time constraints of transactions take the form of periodic, non-periodic, deadline based or non-deadline based; they can also be specified as relative relationships among different transactions. e.g., transaction A must be finished 10 seconds before transaction B. We should leverage those well-studied methodologies and build time constraints into sensor query language. In addition, we should provide an approximate means for specifying time constraints. When there is uncertainty in the exact timing of event occurrences [55], each event occurrence can have a timestamp given by a time interval. We believe this will provide more flexibility for applications.
- Specifying reliability: Faults in sensor networks create ambiguity in answers to queries. For example, it is impossible for applications to know whether the answer is based on partial reports from sensors, whether the unreported readings are missing or just because those sensors do not satisfy the predicates. Reliability requirements can take the form of event detection probability, or percentage of readings from a certain region, etc. The `recall' metric used in information retrieval can be used to measure the completeness of answer to selection queries. i.e., applications can specify in the language their desired recall requirements.

Our discussion so far about language support assumes deterministic guarantees to these non-functional needs. In fact, these assumptions can be relaxed. Some applications would be satisfied with probabilistic guarantees [21]; in addition, applications may have various preferences towards these non-functional needs. Some type of probabilistic reasoning will be necessitated

by the fact that sensors record measurements at points in space and time, whereas applications are interested in the underlying phenomena which are continuous in nature, and can be approximated e.g., by interpolation [34]. Therefore, the language should include clauses to applications to specify their preferences, which can later be used to guide sensing and collection.

Translation of Application Needs: In fact, an application's non-functional needs manifest themselves in the several layers of the system; by adapting and translating non-functional requirements between different layers in the system, we are able to satisfy the application requirements in a manner that is sensitive to the underlying resource availability. We next use the accuracy requirement as an example to explain the implication of accuracy needs at application layer, query service layer and data collection layer. We then illustrate how an application's accuracy needs can be mapped to data accuracy needs.

Application accuracy need is related to the real-life goals of applications. For example, a particular application might be interested in detecting a certain event in a region. Accuracy here can be defined as event detection accuracy. Data collection system can adjust accuracy settings at the various service layers with the ultimate goal of satisfying the application accuracy requirement while minimizing system resource assumption. Conversions between accuracy representations at query service and data collection layers depend on the particular query types.

Query accuracy requirements are specified by each query and each answer has certain accuracy guarantees. The querying service uses the data produced by the data collection service to produce answers to queries posed by users. There are several types of queries considered in our system: (a) Queries on an individual sensor may ask for its value at a given time at some accuracy level. When the query can be satisfied using the collected data, these queries can be processed and answered by utilizing only data obtained from the data collection service. Otherwise, the data collection service may begin sampling at a higher rate in order to provide the desired level of quality. (b) A set-based query asks for the set of sensors that possess certain property. The accuracy on the answer set can be usually captured by using 'precision and recall' [10]. If E is the set of sensors which possess the property, e.g., "with its sensing value greater than 100" and A is the set of sensors returned as an answer, then 'precision' measures the fraction of A which should have been returned, i.e., $\frac{A \cap E}{A}$, which measures the purity of the answer, whereas recall measures the fraction of E that was returned, i.e., $\frac{A \cap E}{E}$, which quantifies the completeness of the answer. (c) Aggregate queries are also quite common, which computer count, sum, average, min or max over a set of data. All the queries discussed above can be either one-time queries or continuous queries. Dealing with continuous queries is more challenging due to the fact that storing and updating data demand more system resource. Adapting data accuracy to meet query accuracy requirements is one of the main focuses in this research thrust.

Data collection accuracy is upper bounded by sensing technology, as it is impossible to obtain a better estimate of observed phenomena than that which is technologically feasible using the best sensing technology. However, invoking the sensing, processing the sensed data, and transmitting it, all require significant amount of resources, including energy, CPU cycles and disk space. This can become an excessive burden for resource-limited sensor devices or damaged/overloaded sensing infrastructures. Thus, the data collection should be flexible by: (i)

switching between different sensors if more than one are available; (ii) approximating data time series in order to minimize transmission; finally,(iii) adapting the rate at which samples are obtained, saving on the cost of using the sensors, but producing a coarser approximation of sensed phenomena. The sensing accuracy is governed mostly by the deployed sensors. The accuracy of sensing corresponds to the "measurement error" in a traditional scientific experiment.

The complexity of mapping an application's accuracy requirement to accuracy requirements over sensor measurements depends on both the application as well as the nature of the underlying sensor network. For example, when we use acoustic sensors to track a moving object, the measurement at the sensor cannot directly translate to the displacement of the object. We should allow the application writer to adapt the application logic to deal with imprecise data without worrying about the underlying translation of data to measurement accuracy. We have demonstrated how such a translation can be achieved using a mathematical framework for a target tracking application willing to tolerate a bounded inaccuracy (e.g., tolerance to within 10 meters from the target trajectory) [91]. Similar with accuracy need mapping, we can translate timeliness and reliability requirements as well. For queries with time constraints, we can avoid probing those nodes whose responses are slow [39]; for queries with reliability requirements, we can decide whether those missing reports need to be re-transmitted.

3. Sensor Data Models and Representation

A data model is used to store and represent sensor data at different levels of the architecture. Data representation plays an important role in answering user queries - a good model not only reduces the amount of efforts needed to answer queries, but also facilitates the satisfaction of non-functional needs, such as accuracy and reliability of the sensor data. There are several questions to be answered in developing a data representation model. (a) Shall we choose a simple or a complex model? A complex model might be more accurate, but typically requires more parameters that will need to be exchanged between sensors and servers, incurring extra overhead. It is preferred to use a simple model if it can suffice. (b) How is a model generated? A data model can be static and chosen from a set of fixed models; a model can also be dynamically learned from the changing sensor readings. (c) When should a model or the model related parameters be changed? If it is changed immediately upon a model violation, that would be too aggressive since the violation may be temporary a phenomena; if it is never changed, then it would be too conservative and sensor does not always comply with a single model. (d) Who updates the model? A server may have global knowledge and can maintain history, so a long-haul model is possible; however it does not have the most recent changes in sensor readings. Keeping the data at the servers and sensors consistent requires additional communication overhead. If a sensor updates the model, the advantage is that the sensor has better knowledge of recent history. However, a long-haul model may not be feasible if the sensor does not have enough memory to store the history.

Traditional sensor applications such as temperature and humidity monitoring require the sensor to report the exact data to the server, and the update may be periodically or event-triggered. Such kind of data enables the server to have an accurate view of the sensor data. However, given the limited computational, communication, and storage resources at the sensors, it is expensive to collect and communicate the exact data to the server, especially when the sensor data becomes voluminous. Moreover, there are inherent errors existing in sensors and a single exact data may be incorrect. For example, in an application such as target tracking in a sensor network, error in sensor intensity readings may result in error in localizing the object. Similarly, the result of a query for average temperature in a given region may be im-

precise due to data error. Fortunately, one key observation is that a large number of sensor applications can tolerate a certain degree of error in data. The communication overhead between the data sensors and the server can be alleviated by exploiting the applications' error tolerance. This leads the possibility of using a range to represent the collected data. A larger range will cause less messaging overhead and energy consumption, but will lower the data quality. In contrast, small range will provide higher data quality while incurring bigger overhead and energy consumption.

Related research efforts [64, 37] explores the natural tradeoff between application quality and energy consumption at the sensors. With the dynamicity on both the application requirement and the data source, an adaptive range representation model is proposed [64, 37].The range of the collected data depends on both the application requirement and the cost. An optimized range can be chosen by satisfying the application requirement while minimize the cost.

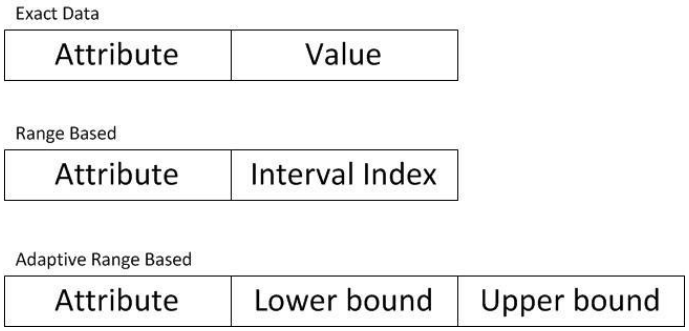

Figure 3: Sensor Data Representations

SensorML: More recently, increasingly ubiquitous sensors brought us sensor data with various content, quality and formats from heterogeneous data source[14], e.g. video surveillance data from pervasive computing space and speech data from firefighters' interphones. To integrate and utilize these various kinds of data, one single sensor data model is not sufficient. More generic and standardized sensor data models should be proposed. Senor Modeling Language (SensorML)[9] is an XML-based modeling language designed for such a purpose. SensorML provides the mechanism for describing the whole range of sensing from simple sensors to arbitrary complex sensor systems and its corresponding platforms. Each sensor is modeled as an operator that is an integral part of a system. Operators consist of input and output behavior, i.e. describe the stimulus received by the sensor and its subsequent action, additional parameters and the input-output transformation function. SensorML meta-data can be used to provide answers to the questions, 1) what is measured (phenomenon); 2) how is it measured (calibration, quality); 3) where is it measured (geometry, spatial response, and sampling); 4) when is it measured (temporal sampling, impulse response); and 5) why is it measured (target application, future processing).

However, the limitation of this standard is that it assumes that users and application writers are able to specify their needs by describing the sensors and operators required for the task at hand. It does not address middleware level challenges such as the need to represent high level concepts, such as entities and activities, which are much more natural for application designers to reason about. Furthermore, the specification does not address adaptivity challenges, hence assuming that there are no resource constraints.

The Virtual Sensor Abstraction: To capture sensing needs at a higher levels of abstraction, SATware[2] abstracts specific sensor formats, data types and the representation of sensor readings so that developers can build applications at the logical level without specific details about the acquisition process or the data formats of the underlying sensor(s). SATware suggests a conceptual level abstracting the raw sensor streams from application writers. More specifically, SATware encapsulates a query into a single operator***: a virtual sensor.*** This is the similar idea to encapsulation in object-oriented programming and aims to hide complexity of operators and simplify the design of applications. It also speeds up application development in SATware and reduces probability of having faulty applications.
Applications can also be modularized and tested independently. Also, reusability is increased since (1) operator topologies (and not only operators) can be now reused, and (2) virtual sensors can be replaced without changing the rest of the application. Virtual sensors provide a controlled avenue toward producing more semantic views closer to the problem domain of the users of sensor data steam processing application.
An example of a virtual sensor is:

$$\text{WhoLeftCoffeeBurning} = O10(\text{CoffeeBurning}, \text{Person\_Id})$$

Where $O10$ is an operator that detects if a person has let the coffee burn more than three times, given a person ID and the burning event. Additionally, the notion of a virtual sensor enables optimizations at other system levels to deal with non-functional constraints. For example the middleware services can be utilized to adapt the data collection process, perform sensor actuations and enable re-calibration of sensors to perturbations and errors.

Semantics-Based Data Models: While virtual sensors provides a higher level abstraction of the sensor data, there is still a significant semantic gap between the information of interest for users (e.g. "where is the evacuation warden?") and the data produced by sensors ("RFID reader 57 and read tag 0815"). A key reason for this gap is that traditional database systems, particularly those focused on capturing and managing data from the real world, are not good at dealing with the noise, loss, and uncertainty in data inherent in information that is obtained from sensors. We argue that a semantics-based data model should be used to represent the uncertainty that is inherent in information obtained from sensors, as a way of bridging this gap. Recently, there has been renewed interest in modeling the uncertainty of a sensor data value using a Bayesian framework. For example, Graphical models [23] have been used to managing and querying large-scale uncertain databases; these models capture not only tuple-level and attribute-level uncertainties, but can also represent arbitrary correlations that may be present among the data. Efficient strategies for query evaluation over such probabilistic databases are also been studied. For example, Deshpande[24] et al have proposed a suite of techniques based on probabilistic models that are designed to allow database to tolerate noise and loss, exploit correlations to predict missing values and identify outliers [22]. Such correlations also provide a way to give approximate answers to users at a significantly lower cost and enable a range of new types of queries over the correlation structure. SATWare extends the semantic ideas in SensorML to capture concepts relevant to sensor based system. It employs several ontologies and concepts to represent the physical world, the sensor, the data and their correlations.

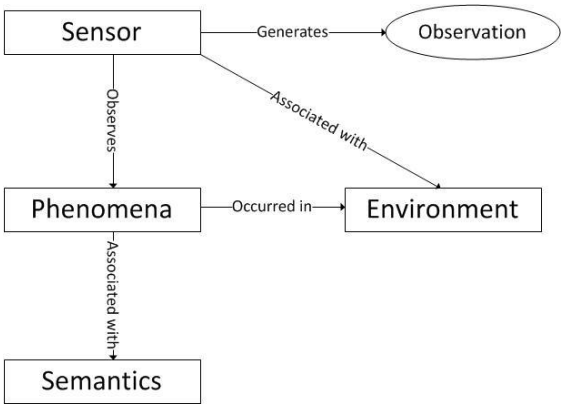

Figure 4 A Generalized Semantics-based sensor data model.

Figure 4 illustrates the key elements of a semantics-based sensor data model that include (a) the sensor; (b)the environment model in which the sensor is embedded; (c)the phenomena (observed by a sensor) and the associated phenomena semantics; (d) the observability of a phenomena, an observation generated by a sensor and the corresponding observation extraction process. We describe these concepts in detail below.

***Phenomena and phenomena semantics*:** Phenomena represent any kind of feature property whose value is amenable to observation or estimation, including physical properties, existence and occurrence assessments, etc. Phenomenon has a type, for example, a person walked in certain space, meeting has started in a certain room, number of people in a region. The phenomena is a measureable value instance that happens in the pervasive space and changes over time. Almost any phenomena can be captured and represented by discretizing the value range. For example, the level of coffee in a coffee pot can be categorized to one of the following values: "full", "half-full" and "empty". The coffee level changes due to brewing of fresh coffee and consumption of coffee and an accurate representation models these state transitions as a function of time. *Phenomena semantics* represent the way that the phenomena evolves as a function of time and space. The phenomena semantics capture the state transition nature of the phenomena as a function of time and space. Consider the coffee level detection task described above, a full coffee level can transition to half-full, empty and overflowing states. The likelihood of each of the possible future states based on the current and past states gives us a meaningful context and understanding of the monitored system, in this case the coffee machine. The semantics of a phenomena give the application a ``window" to the future states of the phenomena in the expected sense. This, in turn, is used to guide the middleware in tasks such as data collection - if it knows that an entity is more likely to appear in a certain camera we can allocate resources accordingly.

Environment Model: creates an abstract representation of the monitored environment. The abstract environment representation allows the modeling of the phenomena state independent of the sensing infrastructure. The environment model creates unique space identifiers for different regions in the monitored space - r1,..,rk. For example, in a building setting the environment model is used to refer to different regions such as ``r1 = south west hallway, second floor" and "r2 = second floor kitchen".

Sensors: Sensors represent physical sources (e.g. devices) that provide input to observe a phenomena. The sensor is associated with metadata information that is used by the different processes that extract observations. Metadata information includes the

- Sensor type. Sensor types include video, motion, temperature, etc. Different sensor types are capable of observing different types of phenomena, for instance, measuring the temperature can be done using a thermometer and not a video camera.

- Environment Cover - the regions of the space that are covered by this sensor. For example, in the case of a camera sensor the regions covered include the parts of the environment in the field of view of the camera.

- Sensor parameters - the state of the sensor (e.g., in the case of camera sensor the state can be zoomed in or zoomed out, titled at 65%, etc.).

Virtual Sensors and the Observation Extraction Process: Virtual sensors abstract the physical source of data and capture the extraction process that is used to observe a phenomena. Observations are generated following an extraction process - D that is controlled by extraction parameters. The extraction process is responsible of generating a digital representation of the high level event that was captured by the sensor. Generating the observation might include multiple operators and processes, however, for our problem we abstract that process and consider that the observation process contains an extraction process as well. For example, detecting if there is a face in the field of view requires processing by a face detection operator. We assume that the observation extraction process also executes all the relevant operators to generate the observation of interest to the application.

Observability of a phenomena*:* A sensor has a finite number of possible states that control the way the sensor observes the monitored phenomena. Given a phenomena at a given location and time, a sensor would be able to observe it and hence will be in the observability set of the phenomena if the following four conditions hold:

1) The sensor type can observe the phenomena of interest.
2) The coverage of the sensor includes the location of the phenomena as specified by the environment model.
3) The state of the sensor is such that the phenomena type is observable in that state by that sensor type.
4) The observation parameters are calibrated to extract the observation correctly.

For example, collecting a high resolution frontal face image of an entity in MeerkatU, can only be done if (a) the sensor is of video camera type, and (b) the face of the entity appears in a region that is part of the field of view of the camera, as specified by our environment model. 3) the camera is zoomed into the region in which the face is present. 4) The face detection software is calibrated to detect that there is face in the field of view of the camera. The observability of a phenomena is related to observability in control systems[1] in which the internals of a system need to be observed using external measurements.

To exploit the above data model concepts effectively in gathering and processing data from large multisensory systems. one must understand the architecture of the distributed sensor system in more detail – this is the topic of the following section.

4. Architectures for Executing Sensor Applications

Given heterogeneous sensor platform distributed over a space and applications data needs expressed as queries, multiple execution architectures are possible. A traditional client-server approach would collect data at a (logically) centralized repository and evaluate queries over such a repository. In such a centralized sensing architecture, sensors are passive units that

send data to the central server. Applications subsequently execute over this server, operating over the data stored there. This approach simply uses the minimum functionality expected of sensors, namely their ability to communicate their captured data to the world at large. Such an approach does not exploit the computation and storage capabilities available within the sensor network at sensor nodes.

In recent years, the availability of cheap wireless sensors has led many researchers to re-evaluate this architecture. Several observations led to this: first, current sensor designs include processing and memory components which are not leveraged if the sensor is considered as a passive beacon of data; second, bandwidth and energy limitations of battery-powered sensors may make the centralized approach very inefficient, as it entails the transmission of every value from its source, via multiple hops, to the server; third, servers themselves would not be able to scale gracefully to the large number of data sources anticipated for future sensor networks due to the low cost of these devices; fourth, time delays for transmitting data to a server before processing it may be prohibitive, and this may be especially critical in applications where real-time response, e.g., actuation, is driven by the sensors themselves.

As a result, many researchers have adopted approaches in which processing is pushed to the sensors themselves, either by eliminating the need for a server altogether, or by introducing some intelligence in the data flow structure, e.g., routing tree, used to transmit data from the sensors to the central server. In the traditional centralized approach, sensors sample periodically and transmit them to the server who then does all the processing. In a completely decentralized approach, data is transmitted between sensors and processed by the sensors themselves to determine whether there is an event of interest [91]. The third and more popular approach is to introduce some intelligence in the routing tree leading to the server; for example, a sensor routes values of other sensors towards the server, and it might avoid doing so if it determines that a sensor's value places it well outside the application's interest. This splits the processing between the server and the sensors, and would result at lower data transmission costs, as fewer values need to be transmitted.

Such “in-network” computation can effectively exploit resources at the sensors to trade computation for reduced communication. By computing directly in the sensor network, the data routing and computing can be co-optimized, resulting in higher scalability. There are, however, limitations of such an approach including limitations on types of data access that can be supported in-network, complexity of optimally splitting computation between sensors and servers, and the lack of a direct way of exploiting applications tolerance to errors and faults. Yet another approach is a hierarchical view in which the actual placement of the distributed server functionality is a function of the node capabilities in a hierarchical setting, the architecture allowing for dynamic migration of functionality based on the well-developed client server model. The exploration of such architectures has, in the literature, been motivated from a narrow perspective of suitability for one (or more) sensor application scenarios (e.g., monitoring, archiving), a comprehensive understanding of the suitability and feasibility of diverse architectures under different situations and blend of application loads is missing. Additionally, the implications of supporting non-functional requirements over these architectures is not well understood in many cases. Therefore, there is a need for a thorough comprehensive analysis of possible architectures with the focus on supporting non-functional application requirements.

There is an intuitive simplicity in the centralized approach, since any application can be conceivably built on top of it. By contrast, approaches which push processing to the sensors entail the need to create distributed versions of every task posed to the network, each of which must

(i) work correctly, i.e., produce a result equivalent to that produced by a centralized approach, and (ii) work efficiently and robustly, especially in the sense of minimizing energy drain and being resilient to failures. It is not immediately clear which architecture is the most appropriate for what type of applications. A methodology for systematic evaluation of different architectural approaches must be based on the following factors:

- ***Application needs:*** Consider an application monitoring certain phenomenon within a geographical region for real-time response. Imagine that a routing tree is used to achieve this goal, with each node in the tree aggregating data received from its children. There are many ways to build such a tree with different node placement and tree construction strategies. To minimize makespan, i.e., the completion time of collecting all data, it is essential to enable and exploit parallel transmissions as much as possible. It seems intuitive that a routing tree with large node fanouts may help increase parallel transmissions and reduce the depth of the tree, thus reduce the notification delay for changes in the observed phenomenon. However, when interference in wireless communication is considered, it is not obvious that large fanouts can always decrease makespan. Given a monitoring application, it is challenging to design a sensor placement and tree construction algorithm under constraints such as network coverage and wireless interference, to achieve the ultimate goal of timeliness. Furthermore, large fanouts may impose a great burden on individual parent sensors which must listen for and aggregate over several values. Therefore, an archival application would prefer a different architecture which prolongs the longevity of the sensor network by optimizing for energy drain alone.
- ***Node density of sensor networks:*** Imagine two sensor networks that differ in the number of nodes which they contain. This point is salient in view of the fact that many current sensor deployments and many research efforts are evaluated over networks of small size, of up to at most a few hundred nodes. Much of the initial excitement about wireless sensors centered around the prospect that networks involving thousands and millions of sensors would be built. What is the best architecture for networks of different size? A centralized approach might impose tremendous processing costs on the central server over a large network. On the other hand, a fully distributed solution may be completely impractical for networks of this size, e.g., because of the long paths of lateral (sensor-to-sensor) communication. It appears that network clustering, which groups sensor nodes into clusters, is the most feasible approach. However, identifying the criteria that determine the sizes and even hierarchies of clusters is a challenging issue. Dynamically maintaining and changing the clusters is also an immediate challenge to accommodate node failures and balance energy drain across sensors.
- ***Heterogeneity of sensor networks:*** Current approaches often assume the existence of a network monitoring a single environmental attribute, e.g., temperature. If multiple sensors exist, e.g., for temperature and atmospheric pressure, then integration of their data might need to happen de facto in a central server, depending on the ability of sensor nodes for temperature/pressure to communicate with each other. Some efforts [45,50,87] address issues emerging from the co-existence of multiple sensing modalities within a single sensor node. Yet, there exist many different possible gradations between a situation where temperature and pressure can only be correlated at a central server and one in which they are measured on the same node. Consider the example of an application for detecting anomalous behavior, e.g., intrusions, in sensitive installations. Light intensity, sound, smoke, video, motion detection and numerous other sensor types may co-exist in such a setting, and data from them ought to be integrated either in-network or at a central site for the purpose of e.g., raising an alarm or tracking an intruder. Bearing certain precedence constraints, e.g., acoustic sensors are much cheaper to operate than video sensors which allows us to check sound intensity first to detect an intrusion event with minimum

energy consumption, data may be routed in a specific way, e.g., from acoustic sensors to video sensors, to enable in-network integration for the purpose of e.g., raising an alarm or tracking an intruder. Decisions on the best architecture for these applications have to be made considering all potential cooperation between sensor nodes.

A Mediation Based Architecture for Sensor Data Management: Concurrently supporting multiple applications is complicated; it is desirable to have a middleware component bridging the applications and the network layer. There are plenty of works on WSN middleware. From the paradigm perspective, we can classify these middlewares into several categories: event-based [15], application driven [43], component based [42] and mediation based. In this chapter, we use a mediator to shield applications from underlying complexity; it is the module where the context collection process is executed. Figure 3 depicts the architectural components of a mediator based framework for context collection. The efficiency of the system depends on specific algorithms applied in each component of the framework. We describe the functionality of each component.

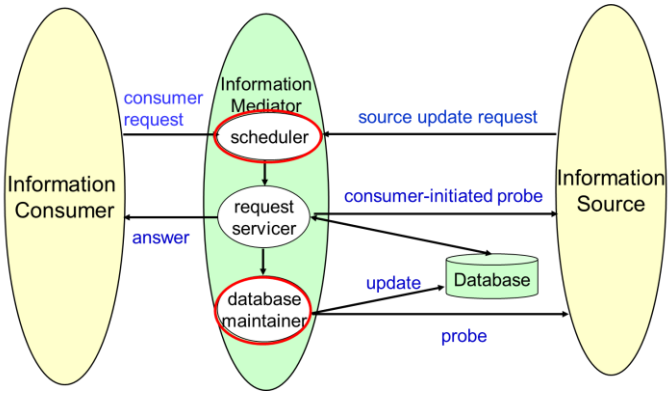

Figure 5 . A mediation-based architecture for sensor systems.

The Information Sources correspond to different components in the distributed sensor system, such as the server, link, mobile or stationary host, and sensors. The context information about sources includes network parameters (such as residual link bandwidth, end-to-end delay on links, link load, link packet drop rate.), server parameters (such as CPU utilization, butter capacity, disk bandwidth.), stationary host parameters (such as client capacity, connectivity), mobile host parameters (such as mobile host location, connectivity, power level), and any data that sensors can capture (such as temperature, humidity, habitat activities). These sources can be programmed to send out information updates periodically, to respond to value requests, or to send out notifications when their values are beyond a certain pre-specified threshold.

The Information Consumers are application and system level tasks that use the data collected from the information sources. For instance, a traffic monitoring application is an application level task that obtains data from highway sensors periodically to assist in traffic planning and routing. Consumer requests are read-only requests that obtain current context, and they are often associated with QoS and/or QoD requirements. The arrival time of one-shot consumer requests are unpredictable, while continuous consumer requests are often in the format of interest registration, i.e., they register desired context data with specified triggering conditions.

The *Data Base* is also called *Context Repository* consisting of distributed databases which hold context information from information sources. Approximate data representation in the

repository can be used to lower the cost of maintaining the repository. For example, each data item can be represented using an interval bounded by an upper and a lower value. The information in the context repository is updated based on current network conditions and user requirements using different update policies.

The *Information Mediator* serves as the decision point for the information collection process. When a number of users request dynamic data at varying requirements under constantly changing system/network conditions, mediation functionality is crucial to deliver the right data to the right user at the right time. Therefore, a key component of context collection architecture is an information mediator, which connects information sources and consumers and serves as a crux of the information collection process where collection decisions are instrumented.

Information Flow of a Context Collection Process: A typical context collection process works as follows. Information sources communicate changes in source values to the mediator, and the mediator uses the most recent value to update the context repository representation if deemed necessary. Information consumers forward their requests to the mediator which in turn retrieves the requested data from the repository. If the repository values meet user specified quality, the mediator returns the answer; otherwise, the mediator probes the sources for current values. In next section, we will show how we collect sensor data on such a mediation based system.

Distributing the Mediator Functionality: Although logically a mediator resides between applications and sensors, in reality, it can be at a very powerful server, or at a resource sufficient base station, or at resource constrained sensors. Since sensor applications may involve a large number of sensors (especially when the observed phenomena moves spatially), it is desirable to use ***distributed cooperating mediators*** to ensure system scalability. Given a large number of consumers (query points) and sensor networks, the choice of how to design the mediation based architecture has multiple challenges - it involves determining an optimal total number of mediators and their locations should be identified that minimizes the overhead involved in satisfying the functional and non-functional needs.

Given multiple mediators, techniques are required to decide where to place data from each sensor and how (e.g., at what accuracy level) the data should be maintained. In addition, we need to maintain a consistent view among all the replicas; with multiple mediators, we need to select for each user request where to retrieve the requested data in order to ensure applications' non-functional needs while balancing the load among distributed mediators. As sensor network infrastructures get more sophisticated, they must provide seamless access to data dispersed across a hierarchy of sensors, mediators, servers and archives—from sensor devices where data originates to large databases where data is stored and/or analyzed. Sensors are more than just passive beacons, but can perform useful work, so we need to consider where to store data and how to push part of the computation to sensors. Archival applications require large amounts of sensor data to be stored for future analysis. If this information is stored at sensors, then we are restricted by sensors limited storage, network and computation resources. If this information is stored at servers, the architecture must be able to cope with high data production rates, and prevent data staleness and/or wasted resources. Several research efforts have been devoted to this area. Dimensions [33] uses in-network wavelet based progressive aging of summaries in support of long-term querying in storage. DCS [77] stores data at a node determined by the name associated with the sensed data. In SDCT [74], finding locations of the nodes for caching data to minimize communication cost corresponds to finding

the nodes of a weighted Minimum Steiner tree whose edge weights depend on the edge's Euclidean length and its data refresh rate. Based on this, a dynamic distributed energy-conserving application-layer service for data caching and asynchronous multicast is presented. TSAR [31] is a storage architecture designed for multi-tier sensor network where an application comprises tens of tethered proxies, each managing tens to hundreds of untethered sensors. TSAR separates data from metadata by employing local archiving at the sensors and distributed indexing at the proxies. At the proxy tier, TSAR employs a novel multi-resolution ordered distributed index structure, the Interval Skip Graph, for efficiently supporting spatio-temporal and value queries. At the sensor tier, TSAR supports energy-aware adaptive summarization that can trade off the cost of transmitting metadata to the proxies against the overhead of false hits resulting from querying a coarse-grain index. The data placement problem should be addressed together with non-functional needs. Specifically, the goal should be to identify the most appropriate data placement strategies to achieve timeliness, accuracy, reliability needs while ensuring energy efficiency.

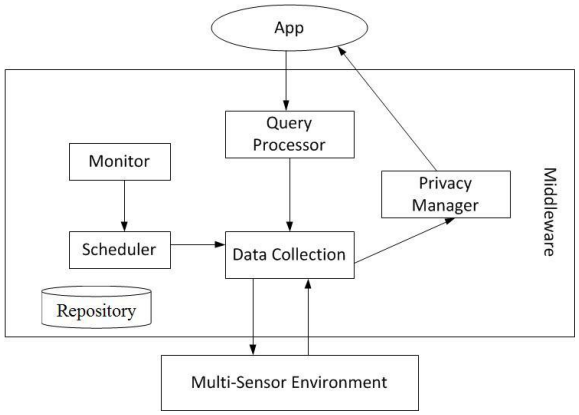

Figure 6 SATware Mediation Middleware

SATware[2] is a distributed multilevel mediation-based middleware for multi-sensor environment. If can efficiently capture, represent, process, and store information from the various data producers (e.g., cameras, motes, mesh routers) at desired levels of accuracy and granularity to meet the information quality and dependability needs of consumers (e.g., video data for surveillance or link congestion levels for routing) given storage and communication constraints. Shown in Figure 6, The data collection module can adjust the collection policy to achieve this. The privacy issues (e.g. in video data) are solved by privacy manager. At the meantime, given the application requirement and the low level sensor environment, optimal sensing schedule can be made to achieve efficient data collection. In general the architecture of SATware mediation middleware that translates application needs (expressed via queries) to a corresponding data collection plan to be executed on the multisensor environment. Such data collection techniques to deal with the multiple needs of sensor applications is the focus of the following section.

5. Sensor Data Collection

Given the importance and potential of the impact of sensor technologies, over the past decade, significant progress has been made on techniques to architect and program large-scale sensor systems. Important developments include design of light-weight operating systems for sensor devices, powerful programming frameworks that isolate application logic from the complexi-

ties of optimizing the computation over sensor networks, techniques for in-network processing that exploit computational resources at the sensors to reduce communication and preserve energy.

While substantial progress has been made, current research has primarily considered *functional* aspects of distributed sensor systems focusing on techniques to sense, capture, communicate, and compute over sensor networks. As sensor applications become more complex and diverse, *non-functional* application needs (such as timeliness, reliability, accuracy and privacy) become important. As an illustrative example, consider a network of sensors monitoring ground movement to detect presence/arrival of enemy forces in a given region in a command and control application. Timeliness and reliability of sensing (in presence of failures) might be of essence here if the countering maneuver requires immediate detection. Such timeliness and reliability requirements, however, come at certain costs, namely additional communication overheads, energy costs, etc. Furthermore, different applications over a given sensor infrastructure may have differing non-functional requirements. For instance, an online monitoring and actuation application might have real-time requirements, an analysis application over the same sensor system might only require that data be collected in a repository (eventually) at a given level of accuracy or spatial and temporal frequency. At the meantime, the more accurate the less privacy the system can provide. For example, an accurate location sensing would expose people's privacy to externals. Such differing application requirements may pose competing requirements on the underlying sensor data collection, coordination, and storage mechanisms. For instance, from the perspective of the archival application, it might be both feasible and desirable that the data be collected, temporarily stored, compressed and then transmitted to the repository. A real-time monitoring/actuation application, however, may demand low latency;

5.1 Quality-Aware Sensor Data Collection: Wireless sensor networks have typically been built with a high degree of dependency between applications and the underlying communication protocols. Such dependency is justified as necessary to achieve energy efficiency. However, it generates rigid systems with sensor networks specifically designed to suit a particular application. While providing a platform that accommodates all types of sensor applications is very difficult, one idea is to build a middleware architecture that can support a representative class of sensor applications - those with multiple performance requirements (in particular QoS, QoD, Cost). We observe that there exists a fundamental tradeoff between the overhead introduced in supporting the application and the QoS/QoD achieved. We refer to this characteristic as the QoS-QoD-Cost tradeoff. If we consider Cost as one dimension and composite performance as another dimension, the application fixes the position of one dimension, and the system is expected to maximize the position along the other dimension.

The Quality-aware Sensing Architecture (QUASAR) [54] is a framework that aims to provide end-to-end support of sensor data collection with data quality specifications [16] and varying QoS and QoD needs (Figure 7). We envision two complementary techniques for quality aware sensor data collection.

The first category includes applications that aim to maximize the QoS/QoD without exceeding the energy budget: This applies when the lifetime of a sensor network is known and the application would like to get as high-quality data as possible. Providing desired timeliness, reliability, accuracy to applications while conserving energy continues to be an important goal. For instance, in the immediate aftermath of a toxic chemical leakage, timely and accurate communication of collected data is much more important than energy efficiency, hence the

application would like to maximize QoS-timeliness and QoD subject to the constraint of remaining energy. With the finite remaining energy level on each sensor node, we are faced with a joint optimization problem when the objective of an application is to maximize more than one metric (i.e., two or three among reliability, timeliness, accuracy).

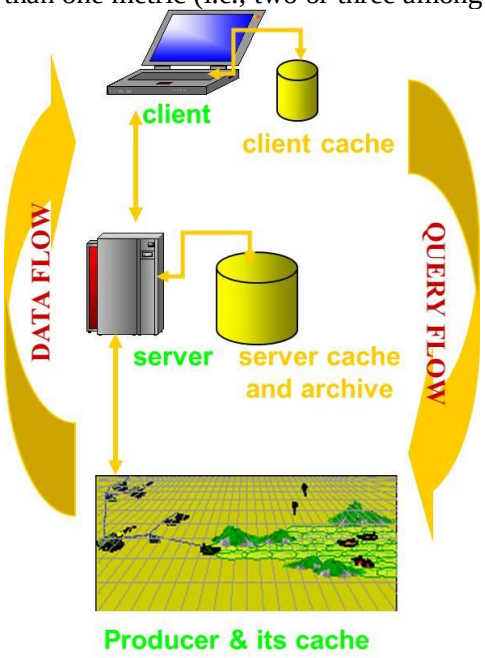

Figure 7. QUASAR Architecture

Existing work has addressed limited subsets of the problem space. Using decoupled strategies that optimize each performance goal in separate phases can unfortunately lead to very expensive data collection plans, since these performance goals are often interdependent. The decision on achieving one objective affects the decision on achieving the other. For instance, improving reliability might entail retransmission of packets, which may lead to increased latency. One strategy is to design a new composite evaluation metric. Most of the existing work focuses on one single performance metric, such as data freshness (i.e., the time elapsed from data generation time to the data collection time) [58], data fidelity (i.e., ratio of nodes participating in the collection to all the nodes in the area) [58], data accuracy measured as how much the obtained data deviates from the real value [37], or as combined spatial and time distortion [23]. We believe it is a better approach if we take into account pre-specified application preferences towards different performance goals and define a composite system performance metric to facilitate the identification of an "optimal" point that would maximize the multiple QoS or QoD requirements. Alternatively, we can investigate the tradeoff between these performance goals and develop a collection plan that optimizes all the goals. More specifically, we might be able to derive theoretical upper bounds on QoS-timeliness, QoS-reliability and QoD, given the energy constraint. We can further design online algorithms that attempt to either maximize the composite performance metric as described above, or jointly optimize multiple goals simultaneously.

An alternate formulation for quality aware data collection can be developed where the goal is to minimize energy consumption while achieving a minimum acceptable level of QoS/QoD: This applies when a sensor network needs to consume as little energy as possible in order to last longer while ensuring the satisfaction of application needs. For instance, careful energy

management of sensor nodes is critical to monitor remnant or new plumes in the days and weeks after the plume disaster; hence, the application would like to minimize energy consumption as long as QoD reaches a certain level. The multiple constraints can be any combination of requirements for QoS-timeliness, QoS-reliability and QoD, e.g., an application might want to minimize energy consumption while ensuring the minimum data accuracy and maximum tolerable latency. Since the granularity at which the sensor data is maintained at the server directly affects the amount of communication, and a sensor consumes energy even when it is idling, the total energy consumption is a function of data granularity and sensor idling time at different power saving states. Therefore, this is a multi-variable optimization problem subject to multiple constraints. An arbitrary combination of algorithms for satisfying timeliness requirements and those for meeting data accuracy needs will result in undesirable system performance, since these constraints are not orthogonal and varying one will typically affect the others. Hence, it is crucial to understand the interplay between different application needs.

In the following, we describe how individual non-functional needs have been supported in the literature.

*Dealing with Data Accuracy Requirements***.** Since sensors are resource constrained, sensor data is often collected into more powerful servers. A natural tradeoff exists between the sensor resources (bandwidth, energy) consumed and the data accuracy collected at the server. Blindly transmitting sensor updates at a fixed periodicity to the server results in a suboptimal solution due to the differences in stability of sensor values and due to the varying application needs that impose different accuracy requirements across sensors. The idea of approximate caching in traditional databases has been applied to resource-constrained sensor networks [47, 54]. The novelty of that work is the application-aware integration of data accuracy satisfaction and power management of sensors [37, 38]. They have developed an optimal data collection protocol that depicts how the server and the sensors collaborate to maintain an optimal representation of sensor data. In concert with this, an optimal algorithm was designed for managing sensor states, which determines the length of sensor idling or sleeping.

Dealing with Reliability Requirements. With the increasingly popular use of WSNs, data is available as never before in many fields of study; practitioners are now burdened with the challenge of doing data-rich research rather than being data-starved. However, in situ sensors can be prone to errors, links between nodes are often unreliable, and nodes may become unresponsive in harsh environments, leaving to researchers the onerous task of deciphering often anomalous data. To this end, the REDFLAG fault detection service is developed that is a Run-time, Distributed, Flexible, detector of faults, that is also Lightweight And Generic [86]. REDFLAG addresses the two most worrisome issues in data-driven wireless sensor applications: abnormal data and missing data. REDFLAG exposes faults as they occur by using distributed algorithms in order to conserve energy. There are also approaches to guaranteeing an application's reliability requirement [36, 69]. Further, the problem of evaluating continuous selection queries over sensor data has been considered in the presence of faults [51, 52]. Reports produced by small sensors may not reach the querying node, resulting in an incomplete and ambiguous answer, as any of the non-reporting sensors may have produced a tuple which was lost. Fault Tolerant Evaluation of Continuous Selection Queries (FATE-CSQ) is a protocol that guarantees a user-requested level of quality in an efficient manner. FATE-CSQ is designed to be resilient to different kinds of failures.

Dealing with Timeliness Requirements. The nature of many sensor applications as well as continuously changing sensor data often imposes real-time requirements on WSN protocols. Due to numerous design constraints, such as limited bandwidth, memory and energy of sensor platforms, and packet collisions that can potentially lead to an unbounded number of retransmissions, timeliness techniques designed for real-time systems and real-time databases cannot be applied directly to WSNs. In order to design a protocol for sensor applications that require periodic collection of raw data reports from the entire network in a timely manner, previous work has formulated the problem as a graph coloring problem and then developed TIGRA (Timely Sensor Data Collection using Distributed Graph Coloring)—a distributed heuristic for graph coloring that takes into account application semantics and special characteristics of sensor networks [70, 71]. TIGRA ensures that no interference occurs and spatial channel reuse is maximized by assigning a specific time slot for each node. Although the end-to-end delay incurred by sensor data collection largely depends on a specific topology, platform, and application, TIGRA provides a transmission schedule that guarantees a deterministic delay on sensor data collection.

5.2 Dealing with Composite requirements. A more challenging issue is to ensure that the multiple non-functional needs posed by application queries: timeliness, data accuracy, reliability are met in a cost-effective manner. However application requirements may conflict with each other. Consider the following cases.

- *Accuracy vs. timeliness:* Frequent updates from sensors will undoubtedly improve accuracy, however, it might also leave the server not enough time to process user queries, hence violating many time constraints. A straightforward application of existing algorithms for dynamic data management that address the cost and accuracy tradeoffs attempt to keep the database ``reasonably'' accurate. However, it ignores the arrival order and time constraints of user requests. i.e., an urgent request might have to wait for a long time before it can be processed; even though the request can be processed by merely retrieving values from the repository, the long waiting time may lead to the violation of time constraints. There are other subtle implications that arise when timing needs are ignored. For instance, handling update requests from sensors prior to retrieval requests might either improve or worsen timing and cost depending on how many update requests remain to be processed. Furthermore, the cost involved in sensor data collection is not only introduced by communication, but also sensor idling at different power saving states.
- *Timeliness vs. reliability:* In order to ensure reliability in the presence of faults, there may be a need to re-transmit data; however, this recovery can be time-consuming, hence leading to degradation in timeliness satisfaction. Directly applying existing fault tolerant techniques developed for delivering sensor data may provide certain reliability guarantees. However, those techniques typically are ignorant of timing needs of applications. For instance, in order to ensure reliable data delivery, hop-by-hop recovery is often applied; however, the final arrival of the data might lag behind significantly and become useless since the timing constraints are violated.
- *Reliability vs. accuracy:* queries may derive some missing sensor data based on historical or neighboring reports, and this might be sufficient for reliability needs; however, this implies that the derived data may not be accurate.

Existing and ongoing research has addressed the tradeoffs between two dynamic factors in different context. For instance, the tradeoffs between transaction timeliness and data freshness have been studied via different real-time scheduling algorithms in real-time and temporal databases[3,24,94]; the accuracy and cost tradeoffs are explored by developing various cache management algorithms in dynamic data management [46, 64, 40, 66, 57, 18, 62, 6, 16, 12,

54], the tradeoffs between reliability and energy efficiency in sensor networks have been investigated [60, 22, 84, 7, 88, 73, 85, 79, 89, 32, 90, 27]. However, the problem of supporting multi-dimension (possibly conflicting) non-functional needs -- timeliness, accuracy, reliability and cost-effectiveness -- simultaneously in the context of heterogeneous sensor networks remains a challenge. While real-time scheduling algorithms determine the order of request processing with the objective of increasing the number of requests with their time constraints met, they are incognizant of communication and data processing overheads involved in the data collection process in highly unreliable distributed sensor environments. The overheads come from of sampling sensors, retrieving data from repositories, updating repositories, etc. To address the cost constraint, these overheads should be kept to a minimum while addressing the timeliness/accuracy/reliability needs. In addition, the current approaches do not address the issues of how the scheduled request should be processed. For instance, should it obtain information from the repository and sacrifice accuracy for cost or should it probe the sensor and sacrifice cost for current and possibly future accuracy?

Determining an appropriate composition of the services for reliable accuracy-aware scheduling and cost-aware database maintenance is challenging due to the following reasons. Firstly, achieving optimality (minimized overall cost while meeting timelines s needs and accuracy constraints) in sensor data collection is impossible. Addressing the timeliness/accuracy/reliability/cost tradeoffs simultaneously is in fact NP-hard, since the problem is a superset of the classical scheduling problem with mutual exclusion constraints (which has been proven to be NP-hard [5, 82]). Secondly, how the composition should work needs to be determined. Should the services operate independently with no interaction (resulting in poor performance) or should each service be extended to work in concert with the others? Each service becomes more complicated when extended and composed and there are no straightforward rules for how to adjust parameters for each service in order to achieve the overall best performance. It is also unclear exactly what information is needed to enable the composition to make the parameter adjustments. Finally, there are a number of dynamic factors affecting the data collection process: the network latency and link quality are unpredictable and variable, the observed phenomena are highly dynamic, and sensors are failure prone. Hence, adapting the system while balancing the composite tradeoffs of timeliness, accuracy, reliability and cost is not straightforward. Lots of current research has primarily considered functional aspects of distributed sensor systems focusing on techniques to sense, capture, communicate, and compute over sensor networks. To support different non-functional (i.e., quality) needs of sensor data collection, most schemes are implemented at different layers such as MAC layer, routing layer, or data management layer. In fact, these non-functional needs are cross cutting issues that are better addressed by using cross-layer approaches. Motivated by the observation that various biological systems have developed mechanisms to meet conflicting requirements simultaneously, a biologically-inspired solution has been proposed to balance the tradeoffs among conflicting requirements (reliability, timeliness, energy efficiency) and govern mobile agent behavior in sensornets [13, 35] .

To ensure judicious composition, several general strategies can be exploited.

- **Error-aware Prediction:** Here, the individual sensor and mediator dynamically agree upon a model that predicts sensor measurements over the immediate future. If the sensor reading adheres to the predicted model within an error bound, the communication of the sensor readings to the mediator is avoided conserving energy. Prediction based mechanisms can exploit local compute capabilities at the sensor to further optimize communication (the efficacy of the optimization process is dependent on the accuracy of the prediction model). While prediction is useful, there are many issues that need to be resolved

in incorporating prediction models for dynamic data collection. For instance: Who determines the prediction models, the sensor or the mediator? What is the protocol by which a sensor and mediator agree on a model *M*? How does one do dynamic model switching to improve accuracy of the prediction process? What is the energy/performance overhead due to model maintenance and synchronization?

- **Spatiotemporal Sensor Correlation:** As sensor networks scale in size and density, there will be increasing redundancies (correlations) between different sensors [41]. This can be exploited in a principled manner to help recover from failures or provide a quick estimate of the sensor value when probing the sensor may result in violation of timeliness needs. We will develop a probabilistic approach where we model the correlations between different sensors using a probabilistic network (Bayesian net or Markov random field). Using a compact summary of the (joint) distribution of all the sensors, we will attempt to provide reasonably accurate answers to queries such as: Given that sensor X has value X, what is the most likely value of sensor Y? (to fill in missing values, for instance). As a by-product, such an approach can also detect failures. Consider the case where the readings of sensor X have changed significantly, but X is failing to communicate this for some reason. The shift in its data distribution will be reflected in other correlated sensors as well. If these sensors are still functioning, then the changes they signal, used in conjunction with the probabilistic model, will clearly indicate that communication is to be expected from X. The absence of such communication is a sign of the possible failure of X.
- **Application aware caching:** Caching part of the sensor data can be performed at different levels: at the sensor node, at the intermediate mediator, or at a centralized server. Cached data may be accurate enough to answer some queries, this may enable better timely results to queries. Cached data can be used to infer current sensor readings as well, which improves timeliness while avoiding the overhead of re-transmitting data from sensors. Questions that need to be answered while developing these techniques include where to cache, what to cache and how to cache (i.e., at what level of resolution).

6 Querying and Query Optimization in Sensor Networks

User tasks submitted in a high-level language appropriate to the application domain, will be mapped to appropriate data management primitives by the application software which will then be posed to the sensor database management system in a declarative (e.g., SQL-like) language. In this section, we will describe the various types of sensor queries and potential optimizations for query processing in sensor networks.

A query from a sensor application may ask for a data value produced by a sensor device at any time instant including the future, the present or the historical past. While queries about the future cannot be answered precisely at the present time by either the producer or the archiver, queries about the present can be answered by the data producer or the device in our case. For historical queries, the only choice is to answer them *approximately* at the archive since the producer discards the time series once it has sent a compressed representation to the archive. For queries seeking future data values, we seek to build evaluation strategies that allow tremendous scaling to large-scale data networks. Thus, we use an approximate representation of the actual data at the sources *within bounded quality guarantees* using a *quality-aware* client-server architecture to achieve scalable, energy and bandwidth efficient query processing. The data collection architecture as described above allows us to capture current and past data with a given quality. Applications need to be able to run on this imprecise data producing approximate results. While there are many types of queries are possible, we con-

sider three types of queries over the imprecise data representation of particular interest to sensor applications: continuous monitoring queries, general purpose aggregate and SQL queries and monitoring applications.

Continuous Queries for Monitoring Applications: Continuous queries are those in which a query runs periodically over a time period (e.g. summary of the traffic information on a freeway every 5 minutes). Such queries may be continuous aggregate queries with different precision bounds on their answers. Different queries will involve different sets of source/sensor data with possible overlaps. If the communication cost to collect a single data on any specific source is known, it is an interesting research problem to look for the optimal precisions on each data source so that the total communication cost is minimized and all query quality requirements are satisfied as well. Again the issue is given the error threshold, how to push the computation to the sensors and exploit inter-sensor communication to minimize the net communication overhead between the sensors and the server.

An example of continuous queries is the tracking of mobile objects over a period of time. The basic approach to tracking using sensors works as follows: at any instance, a set of sensors whose sensing range the object to be tracked is located in are activated. The activated sensors communicate their readings to the tracking module periodically which fuses the sensor readings to determine the location of a mobile object at any given time. The more frequent the communication, better is the track, where quality is defined as the reciprocal of the deviation from the actual trajectory. However, it incurs higher cost. Depending upon the type of sensor used, in its simplest form, the location can be triangulated from three sensor values.

In contrast to the above, an alternative tracking framework is to explore a systematic mechanism using which the energy consumption of the tracking module can be controlled based on the application's quality/accuracy requirement. Specifically, an application may specify its willingness to tolerate a bounded inaccuracy -- e.g., tolerance to within 10 meters from the target trajectory. The application tolerance can be exploited to reduce communication between sensors and the tracking module. A key aspect of this approach is the translation of application quality to measurement quality at the sensor. At any stage during tracking, each enabled sensor and the tracking module dynamically agree upon a model that predicts the object's mobility in the immediate future. Such a model can either be communicated by the tracking module to the sensor when the sensor is activated or alternatively determined by the sensor using model fitting techniques based on a set of readings. If the object adheres to the predicted model within the error bound, the communication of the sensor readings to the tracking module is avoided thereby conserving energy. Note that the correctness of the approach is independent of the efficacy of the prediction model in predicting the target motion, though the effectiveness of the strategy (in terms of reducing communication) is directly related to the model. Previous work [91] has established the feasibility of the above envisioned adaptive tracking framework on top of adaptive precision data collection mechanisms. Using even the very conservative policies to adapt the sensor precision based on application tolerance, we can get many-fold improvement in power conservation.

*Answering Value-Based Queries with Bounded Quality***:** A query may ask for a data value (or aggregation of a set of data values) produced by a device at any time instant including the future, the present or the historical past. While queries about the future cannot be answered precisely at time n (which is the present time) by either the producer or the archiver, queries about the present can be answered by the data producer or the device in our case. For historical queries, the only choice is to answer them approximately at the archive since the producer

discards the time series once it has sent a compressed representation to the archive. If a query asks for a data value at a time instant, it may be answered using any of the three following evaluation strategies:

***Probe*:** The server issues a direct request to the device for the data item at the time instant. However, this requires that the producer, or the device, maintain the samples it has not yet sent to the archive and listen continuously on the communication channel, thus making it in appropriate for the energy-constrained wireless sensor nodes

***Wait*:** The server may wait for the data item to arrive. However if the quality requirements of the query is more stringent than ε (precision at the sensor), then the answer would be incorrect.

***Predict*:** A predictive model (M, θ) is stored at the archive. Thus the archive or the server predicts the future value of the device and answers the query. The use of prediction to answer queries is motivated by the communication latency between the producer and the archiver of the time series. It does away with either doing a probe or waiting for a value to e sent by a sensor. Also, the archive can answer to interested applications, within ε_{pred} estimation of time series values before these values arrive at the archive.

Answering Set-Based Queries with Bounded Quality: For queries that return a set of answers, (e.g., sample points in space-time where temperature exceeds 100 degrees), one of the key issues is how to measure the quality of results. Many metrics to measure differences between two sets have been proposed in the literature (e.g., Earth movers distance, match-and compare, etc.). While such measures could be used *diagnostically* to quantify the closeness of results to the true answer (i.e., quality of the results) when the true answers are known, there is no straightforward mechanism to estimate the quality of result at the server using these measures in a *prescriptive* setting such as ours when the true answers are not known a priori. An ideal measure, for our setting is such that (1) it is easy to compute, (2) does not require advance knowledge of exact answers, and (3) can be used to support progressive improvements of results. We have recently shown that *precision* (that measures the purity of results) and *recall* (that measures the completeness of results) suitably adapted to measure the set discrepancy metrics satisfy the above requirements [53]. The key aspect of these measures that makes them suitable for our purpose is that even though precision and recall cannot be accurately computed without knowledge of the true answer set, a range that provides a lower and upper bound on them can be determined without explicit knowledge of the true answers. Thus, at any stage of query processing, a tight bound on accuracy of the returned results can be determined. If the bound is not sufficient to meet application needs, results can be suitably refined. Previous work has explored such a mechanism for a simple selection queries that select certain sensor readings based on whether or not they satisfy a specified predicate. One direction of future research is to generalize this approach to a larger class of SQL queries including join queries.

Energy-aware query processing: Since the most critical resources in WSNs are energy and communication capabilities, research efforts in query optimization have aimed to minimize energy consumption and communication loads. *In-network data aggregation* is one such technique; it relies on the observation that a sensor node consumes much more energy to communicate than process data. Here sensor nodes operate in multiple phases to process the received sensor data and then send out the aggregated information. During the first phase , a hierarchical tree structure rooted at an access point is discovered via techniques such as network clustering [19, 49]. For example, in TAG[60], a routing tree is formed during the query dissemination phase. A node always aggregates incoming data before sending it to its parent. Approximation techniques [25, 65] have also been explored to achieve energy-efficient

data aggregation by eliminating unnecessary messages. [93] tackles issues that arise due to shared media access by exploring sensor state scheduling for data collection and aggregation within sensor networks. They first identify potential collisions relevant to aggregate monitoring in the deployed wireless environment, and then propose TDMA scheduling algorithms based on greedy heuristics to determine sensor node operation states to achieve energy-efficient and collision-free communication for data collection and control message dissemination with short makespans. Other work [8, 78, 59] organizes the queries into groups according to their attributes so that similar queries results can be collected using the same path.

Distributed query processing: is an alternative approach to processing triggered queries. Although tree-based query processing has been shown to work well for monitoring queries in general, utilizing a fixed network topology for triggered queries is not always the best solution. SURCH (SURfing and searCHing) [92] is a fully decentralized peer-based algorithm for processing triggered queries in wireless sensor networks. SURCH combines query dissemination and processing so that a query can be partially processed on the fly in a sensor network. Partial results are delivered to the query initiator or a designated proxy for final processing. SURCH avoids the overheads of network topology construction or interaction with a server by exploiting local communication. The novelty of SURCH is four-fold. First, it is capable of processing ad hoc in-network generated queries more efficiently than existing tree-based techniques in addition to continuously monitoring queries. Second, by implementing prioritization, when only a small number of sensor nodes contribute to the query result, SURCH demands very little communication. Third, sensor workloads can be balanced to maximize the overall life-time of sensor networks through carefully designed propagation policies. Finally, SURCH has an inherent resilience to sensor failures, as it does not depend on any particular node, but works "around" failed nodes, discovering the ones that are still operational.

7 Conclusions

In this chapter, we discussed techniques and solutions for effective data management in distributed sensor networks. In particular, novel approaches to data representation, distributed sensor architectures, and query processing will trigger the development of alternative protocols/services for adaptive optimization engine that can be used over a wide range of sensor networks and applications. Sensor networks are an example of a complex technical system with multiple operational constraints and performance goals dictated by the deployed application at hand. In general, the dynamic nature of observed phenomena under varying system and network conditions implies that policies instrumented in the system should be dynamic and customizable. Multiple system and application activities can interfere with each other when they occur concurrently in distributed sensor systems. Fundamental problems arise in the concurrent execution of multiple distributed services and protocols that manage multiple needs; further complications arise in the dynamic customization of these services and protocols. Arbitrary composition of those techniques developed for addressing each individual non-functional need will result in unsatisfactory system performance, when there are multiple needs. More research is needed to address the complexity of supporting distributed sensor data collection application under multiple conflicting requirements: timeliness, accuracy, reliability and cost-effectiveness.

A widescale and effective deployment of sensing infrastructures has the potential to impact the scientific community (disaster management, ecosystem monitoring, command and control) at large; further research in this area will help generate guidelines for the development of sensor architectures for specific disciplines. Novel challenges for the new decade include sus-

tainability of large sensing infrastructures that provide intelligence to applications while preserving the privacy of individuals in sensed spaces.

References

1. http://en.wikipedia.org/wiki/Control_system, 2012
2. http://www.ics.uci.edu/~projects/SATware/, 2012
3. B. Adelberg, H. Garcia-Molina, and B. Kao. Applying update streams in a soft real-time database system. In Proceedings of ACM International Conference on Management of Data (SIGMOD), 1995.
4. Werner-Allen, Geoffrey, et al. "Deploying a wireless sensor network on an active volcano." Internet Computing, IEEE 10.2 (2006): 18-25.
5. A.K.Mok. Fundamental design problems of distributed systems for the hard-real-time environment. PhD thesis, MIT, 1983.
6. B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In Proceedings of ACM Symposium on Principles of Database Systems (PODS), 2002.
7. Brian Babcock, Mayur Datar, and Rajeev Motwani. Load shedding for aggregation queries over data streams. In Proc. of ICDE Conference, 2004.
8. Bonfils, B., & Bonnet, P. Adaptive and decentralized operator placement for in-network query processing. In Proceedings of IPSN (International conference on information processing in sensor networks) (pp. 47–62), 2003.
9. Botts, M.: OGC Implementation Specification 07-000: OpenGIS Sensor Model Language, SensorML, 2012
10. Ricardo Baeza-Yates and Berthier Ribeiro-Neto. Modern Information Retrieval. Addison Wesley, 1999.
11. C. Bisdikian, L. M. Kaplan, M. B. Srivastava, D. J. Thornley, D. Verma, and R. I. Young. Building principles for a quality of information specification for sensor information. In Proceedings of the 12th International Conference on Information Fusion, 2009.
12. Philippe Bonnet, J. E. Gehrke, and Praveen Seshadri. Towards sensor database systems. In Proceedings of IEEE International Conference on Mobile Data Management (MDM), 2001.
13. P. Boonma, Q. Han, and J. Suzuki. Leveraging biologically-inspired mobile agents supporting composite needs of reliability and timeliness in sensor applications. In Proceedings of IEEE International Conference on Frontiers in the Convergence of Bioscience and Information Technologies (FBIT), pages 851-860, 2007.
14. Eric Bouillet, Mark Feblowitz, Zhen Liu, Anand Ranganathan, Anton Riabov, Fan Ye: A Semantics-Based Middleware for Utilizing Heterogeneous Sensor Networks. DCOSS 2007: 174-188
15. E. Brewer, M. Demmer, B. Du, M. Ho, M. Kam, S. Nedevschi, J. Pal, R. Patra, S. Surana, and K. Fall. The case for technology in developing regions. In IEEE Computer, June 2005.
16. D. Carney, U. Cetintemel, M. Cherniack, C. Conveyand S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik. Monitoring streams: A new class of data management applications. In Proceedings of the 28th International Conference on Very Large Data Bases (VLDB), 2002.
17. S. Chakravarthy, V. Krishnaprasad, E. Anwar, and S.K. Kim. Composite events for active databases: Semantics, contexts and detection. In Proceedings of the Interna-

tional Conference on Very Large Data Bases (VLDB), 1994.
18. J. Chen, D. J. DeWitt, F. Tian, and Y. Wang. NiagaraCQ: A scalable continuous query system for internet databases. In Proceedings of ACM International Conference on Management of Data (SIGMOD), 2000.
19. Y. Chen and A. Liestman. Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks. In The 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2002.
20. Chintalapudi, Krishna, et al. "Monitoring civil structures with a wireless sensor network." Internet Computing, IEEE 10.2 (2006): 26-34.
21. Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. Evaluating probalistic queries over imprecise data. In ACM SIGMOD, 2003.
22. J. Considine, F. Li, G. Kollios, and J. Brers. Approximate aggregation techniques for sensor databases. In Proceedings of the IEEE International Conference on Data Engineering (ICDE), 2004.
23. R. Cristescu and M. Vetterli. On the optimal density for real-time data gathering of spatiotemporal processes in sensor networks. In Proceedings of IEEE International Conference on Information Processing in Sensor Networks (IPSN), 2005.
24. A. Datta and I. Viguier. Providing real-time response, state recency and temporal consistency in databases for rapidly changing environments. Information Systems, 22(4), 1997.
25. A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Hierarchical in-network data aggregation with quality guarantees.In EDBT, 2004.
26. A. Demers, J. Gehrke, R. Rajaraman, N. Trigoni, and Y. Yao. The cougar project: A work-in-progress report. ACM SIGMOD Record, 32(4), 2003.
27. A. Deshpande, and C. Guestrin and S. Madden and J. Hellerstein and W.Hong. Model Driven Approximate Query in Sensor Networks. The VLDB Journal, 14(4), 2005
28. Amol Deshpande, Lise Getoor, and Prithviraj Sen; Book Chapter. Graphical Models for Uncertain Data; Managing and Mining Uncertain Data, ed. C. Aggarwal, Springer, 2009
29. Amol Deshpande, Carlos Guestrin, and Samuel Madden. Using Probabilistic Models for Data Management in Acquisitional Environments. In the Proceedings of CIDR , 2005
30. Amol Deshpande, Carlos Guestrin, Samuel Madden, and Wei Hong. Exploiting Correlated Attributes in Acquisitional Query Processing. ICDE, 2005.
31. P. Desnoyers, D. Ganesan, and P. Shenoy. Tsar: A two tier storage architecture using interval skip graphs. In ACM Sensys, 2005.
32. D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. ACM Mobile Computing and Communications Review, 1(2), October 2002.
33. D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann. An evaluation of multi-resolution search and storage in resource-constrained sensor networks. In SenSys, 2003.
34. Stephane Grumbach, Philippe Rigaux, and Luc Segoufin. Manipulating interpolated data is easier than you thought. In The VLDB Journal, pages 156-165, 2000.
35. Q. Han, D. Hakkarinen, P. Boonma, and J. Suzuki. Quality-aware sensor data collection. International Journal of Sensor Networks (IJSNet), Special Issue on Data Quality Management in Wireless Sensor Networks, 7(3):127-140, 2010.
36. Q. Han, I. Lazaridis, S. Mehrotra, and N. Venkatasubramanian. Sensor data collection with expected reliability guarantees. In Proceedings of IEEE International Workshop on Sensor Networks and Systems for Pervasive Computing (PerSeNS), pages 374-

378, 2005.
37. Q. Han, S. Mehrotra, and N. Venkatasubramanian. Energy efficient data collection in distributed sensor environments. In Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS), pages 590-597, 2004.
38. Q. Han, S. Mehrotra, and N. Venkatasubramanian. Application-aware integration of data collection and power management in wireless sensor networks. Journal of Parallel and Distributed Computing (JPDC), 67(9):992-1006, September 2007.
39. Q. Han, M. B. Nguyen, S. Irani, and N. Venkatasubramanian. Time-sensitive computation of aggregate functions over distributed imprecise data. In Proceedings of IEEE International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS), 2004.
40. Q. Han and N. Venkatasubramanian. Autosec: An integrated middleware framework for dynamic service brokering. IEEE Distributed Systems Online, 2(7), 2001.
41. G. Hartl and B. Li. Infer: A bayseian inference approach towards energy efficient data collection in dense sensor networks. In Proceedings of International Conference on Distributed Computing Systems (ICDCS), 2005.
42. G. Heineman and W. Councill. Component-Based Software Engineering: Putting the Pieces Together. Addison-Wesley: Reading, MA, 2001
43. W. Heinzelman, A. Murphy, H. Carvalho, and M. Perillo. Middleware to support sensor network applications. IEEE Network, 1(18):6–114, 2004.
44. Henaut, Julien, Daniela Dragomirescu, and Robert Plana. "Fpga based high date rate radio interfaces for aerospace wireless sensor systems." Systems, 2009. ICONS'09. Fourth International Conference on. IEEE, 2009.
45. Hore, Bijit, et al. "SATware: Middleware for Sentient Spaces." Multimodal Surveillance: Sensors, Algorithms and Systems (2007).
46. Y. Huang, R. Sloan, and O. Wolfson. Divergence caching in client-server architectures. In Proceedings of the IEEE Third International Conference on Parallel and Distributed Information Systems (PDIS), 1994.
47. I. Hwang, Q. Han, and A. Misra. MASTAQ: A middleware architecture for sensor applications with statistical quality constraints. In Proceedings of IEEE International Workshop on Sensor Networks and Systems for Pervasive Computing (PerSeNS), pages 390-395, 2005.
48. Allred, Jude, et al. "Sensorflock: an airborne wireless sensor network of micro-air vehicles." Proceedings of the 5th international conference on Embedded networked sensor systems. ACM, 2007.
49. K. Kalpakis, K. Dasgupta, and P. Namjoshi. Maximum lifetime data gathering and aggregation in wireless sensor networks. In the IEEE International Conference on Networking, 2002.
50. Kim, Minyoung, et al. "A semantic framework for reconfiguration of instrumented cyber physical spaces." Workshop on Event-based Semantics, CPS Week. 2008.
51. I. Lazaridis, Q. Han, S. Mehrotra, and N. Venkatasubramanian. Fault-tolerant queries over sensor data. In Proceedings of International Conference on Management of Data (COMAD), pages 104-116, 2006.
52. I. Lazaridis, Q. Han, S. Mehrotra, and N. Venkatasubramanian. Fault-tolerant evaluation of continuous queries over sensor data. International Journal on Distributed Sensor Networks (IJDSN), 5(4):338-360, 2009.
53. Iosif Lazaridis, Sharad Mehrotra: Capturing Sensor-Generated Time Series with Quality Guarantees. ICDE 2003: 429-440
54. I. Lazaridis, Q. Han, X. Yu, S. Mehrotra, N. Venkatasubramanian, D. Kalashnikov, and W. Yang. Quasar: Quality aware sensing architecture. ACM SIGMOD Record,

33(1):26-31, March 2004.
55. C.G. Lee, A.K. Mok, and P. Kanana. Monitoring of timing constraints with confidence thresholds. In Proceedings of IEEE Real-Time Systems Symposium (RTSS), 2003.
56. Lee, Edward A. "Cyber-physical systems-are computing foundations adequate."Position Paper for NSF Workshop On Cyber-Physical Systems: Research Motivation, Techniques and Roadmap. Vol. 2. 2006.
57. L. Liu, C. Pu, and W. Tang. Continual queries for internet scale event-driven information delivery. Proceeding of IEEE Transactions on Knowledge and Data Engineering (TKDE), 11(4):610-628, July/Aug. 1999.
58. C. Lu, G. Xing, O. Chipara, C. L. Fok, and S. Bhattacharya. A spatiotemporal query service for mobile users in sensor networks. In Proceedings of International Conference on Distributed Computing Systems (ICDCS), 2005.
59. Sharaf, M. A., Beaver, J., Labrinidis, A., & Chrysanthis, P. K. Balancing energy efficiency and quality of aggregate data in sensor networks. The VLDB Journal, 13(4), 384–403, 2004.
60. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In Proceedings of USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2002.
61. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tinydb: An acqusitional query processing system for sensor networks. ACM Transactions on Database Systems (TODS), 30(1), March 2005.
62. S. R. Madden and M. J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In Proceedings of the IEEE International Conference on Data Engineering (ICDE), 2002.
63. S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In Proceedings of ACM International Conference on Management of Data (SIGMOD), 2003.
64. C. Olston, B. T. Loo, and J. Widom. Adaptive precision setting for cached approximate values. In Proceedings of ACM International Conference on Management of Data (SIGMOD), 2001.
65. C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In SIGMOD, 2003.
66. C. Olston and J. Widom. Best-effort cache synchronization with source cooperation. In Proceedings of ACM International Conference on Management of Data (SIGMOD), 2002.
67. Otto, Chris, et al. "System architecture of a wireless body area sensor network for ubiquitous health monitoring." Journal of Mobile Multimedia 1.4 (2006): 307-326.
68. G. Ozsoyoglu and R. T. Snodgrass. Temporal and real-time databases: A survey. Proceeding of IEEE Transactions on Knowledge and Data Engineering (TKDE), 7(4), 1995.
69. L. Paradis and Q. Han. A survey of fault management in wireless sensor networks. Journal of Network and Systems Management (JNSM), 15(2):171-190, June 2007.
70. L. Paradis and Q. Han. Tigra: Timely sensor data collection using distributed graph coloring. In Proceedings of IEEE International Conference on Pervasive Computing and Communication (PerCom), pages 264-268, 2008.
71. L. Paradis and Q. Han. A data collection protocol for real-time sensor applications. Pervasive and Mobile Computing (PMC), 5(1):369-384, 2009.
72. S. J. Park, Ra. Vedantham, R. Sivakumar, and I. F. Akyildiz. A scalable approach for reliable downstream data delivery in wireless sensor networks. In Proceedings of

ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), 2004.
73. S.J. Park, Ra. Vedantham, R. Sivakumar, and I. F. Akyildiz. A scalable approach for reliable downstream data delivery in wireless sensor networks. In Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), 2004.
74. K. S. Prabh and T. F. Abdelzaher. Energy-conserving data cache placement in sensor networks. ACM Transactions on Sensor Networks, 1(2), November 2005.
75. Polastre, Joseph, et al. "Analysis of wireless sensor networks for habitat monitoring." Wireless sensor networks (2004): 399-423.
76. K. Ramamritham. Real-time databases. International Journal of Distributed and Parallel Databases, 1(2), 1996.
77. 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 Mobile Networks and Applications, 2003.
78. Pattem, S., Krishnamachari, B., & Govindan, R. The impact of spatial correlation on routing with compression in wireless sensor networks. In Proceedings of IPSN (International conference on information processing in sensor networks) (pp. 28–35), 2004.
79. Y. Sankarasubramaniam, O. B. Akan, and I. F. Akyildiz. Esrt: Event-to-sink reliable transport in wireless sensor networks. In Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), 2003.
80. M. Sharaf, J. Beaver, A. Labrinidis, and P.K. Chrysanthis. Balancing energy efficiency and quality of aggregate data in sensor networks. The VLDB Journal, 13(4):384{403, December 2004.
81. A.P. Sistla and O. Wolfson. Temporal conditions and integrity constraints in active database systems. In Proceedings of ACM International Conference on Management of Data (SIGMOD), 1995.
82. J. A. Stankovic, M. Spuri, M. D. Natale, and G. C. Buttazzo. Implications of classical scheduling results for real-time systems. IEEE Computer, 28(6):16-25, 1995.
83. A. Tansel, J. Cli_ord, S. Gadia, S. Jajodia, A. Segev, and R.T. Snodgrass. Temporal Databases: Theory, Design and Implementation. Benjamin/Cummings, 1994.
84. N. Tatbul, U. Cetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker. Load shedding in a data stream manager. In VLDB Conference, 2003.
85. S. Tilak, N. N Abu-Ghazaleh, and W. Heinzelman. Infrastructure tradeo_s for sensor networks. In Proceedings of ACM International Workshop on Wireless Sensor Networks and Applications (WSNA), 2002.
86. I. Urteaga, K. Barnhart, and Q. Han. REDFLAG: a run-time, distributed, flexible, lightweight, and generic fault detection service for data-driven wireless sensor applications. Pervasive and Mobile Computing (PMC), 5(5):432-446, 2009.
87. Vaisenberg, Ronen. Towards Adaptation in Sentient Spaces. DISSERTATION. Diss. UNIVERSITY OF CALIFORNIA, 2012.
88. C.Y. Wan, A.T. Campbell, and L. Krishnamurthy. Psfq: A reliable transport protocol for wireless sensor networks. In Proceedings of ACM International Workshop on Wireless Sensor Networks and Applications (WSNA), 2002.
89. C.Y. Wan, S.B. Eisenman, and A.T. Campbell. Coda: Congestion detection and avoidance in sensor networks. In Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys), 2003.
90. F. Ye, G. Zhong, S. Lu, and L. Zhang. Gradient broadcast: A robust data delivery protocol for large scale sensor networks. ACM Baltzer Journal on Wireless Networks (WINET), 11(2), 2003.

91. X. Yu, K. Niyogi, S. Mehrotra, and N. Venkatasubramanian. Adaptive target tracking in sensor networks. In CNDS, 2004.
92. Xingbo Yu, Sharad Mehrotra, Nalini Venkatasubramanian: SURCH: Distributed Aggregation over Wireless Sensor Networks. IDEAS 2006:158-165
93. Xingbo Yu, Sharad Mehrotra, Nalini Venkatasubramanian: Sensor Scheduling for Aggregate Monitoring inWireless Sensor Networks. SSDBM 2007: 24
94. R. Zhang, C. Lu, T. F. Abdelzaher, and J. A. Stankovic. Controlware: a middleware architecture for feedback control of software performance. In Proceedings of International Conference on Distributed Computing Systems (ICDCS), 2002.