Real-Time Syst (2012) 48:463–498
DOI 10.1007/s11241-012-9154-0

Quality-aware data abstraction layer for collaborative 2-tier sensor network applications

Woochul Kang · Sang H. Son · John A. Stankovic

Published online: 4 May 2012

Abstract This paper presents PRIDE, a novel data abstraction layer for collaborative 2-tier sensor network applications. PRIDE, more specifically, targets distributed real-time applications, in which multiple collaborative mobile devices have to analyze a global situation by collecting and managing data streams from massive underlying sensors. PRIDE at these devices hides the details of underlying sensors and provides transparent, timely, and robust access to global sensor data under highly dynamic and unpredictable environments of emerging sensor network applications. For transparent and efficient sharing of global sensor data, a model-based predictive replication mechanism is proposed and integrated into a conventional data management system that supports diverse types of spatial and temporal queries. In addition, for robust and timely query processing, the predictive replication scheme is extended to the problem of guaranteeing Quality-of-Service (QoS) by introducing feedback control of the accuracy bounds of models. We show the viability of the proposed solution by implementing and evaluating it on a 2-tier sensor network testbed, emulating collaborative search-and-rescue tasks with realistic workloads. Our evaluation results demonstrate that PRIDE can achieve timely sensor data sharing among a large number of devices in a highly robust and controlled manner.

Keywords Real-time collaboration · Sensor data replication · Sensor networks · Feedback control · Implementation

W. Kang (✉)
Daejon, South Korea
e-mail: wchkang@etri.re.kr

S.H. Son · J.A. Stankovic
Charlottesville, VA, USA

S.H. Son
e-mail: son@virginia.edu

J.A. Stankovic
e-mail: stankovic@virginia.edu

1 Introduction

1.1 Motivation

Recent advances in sensor technology and wireless connectivity have paved the way for next generation real-time applications that are highly data-driven, where data represent real-world status. For many of these applications, data streams from underlying sensors are managed and processed by application-specific devices such as PDAs and micro servers. Further, as sensors are deployed in increasing numbers, a single device cannot handle all sensor streams due to their scale and geographic distribution. Often, a group of such devices need to collaborate to achieve a common goal. For instance, consider a team of firefighters involved in a search-and-rescue task during a building fire. While PDAs carried by firefighters collect data from nearby sensors to check the dynamic status of the building, they have to collaborate by sharing their locally collected real-time data with peer firefighters since each individual firefighter has only limited information from nearby sensors (Sha et al. 2006; http://w3.antd.nist.gov/comm_net_ps.shtml 2008; http://fire.me.berkeley.edu/ 2008). The building-wide situation assessment requires the fusion of data from all (or most of) firefighters. As this application shows, in lots of future applications envisioned by Cyber-Physical Systems (Lee 2008; Stankovic et al. 2005), the data from underlying sensor nodes will be managed by distributed devices in cooperation. Their operating environments can be highly dynamic including mobile entities such as PDAs and moving sensor nodes. Sharing data to allow timely access to global data for each participating entity is mandatory for successful collaboration in such distributed real-time applications.

However, current tiered sensor network architectures, in which a group of devices at the upper tier manage data from the underlying sensor tier, have a limitation in satisfying the requirements of aforementioned emerging applications in terms of timeliness, flexibility, and robustness. Since sensor data is partitioned and distributed across the upper-tier devices, or proxies, queries accessing global sensor data cannot be processed in a timely manner; the queries incur unpredictable communication delays between upper-tier devices to locate and access distributed sensor data (Gnawali et al. 2006; Li et al. 2006; Desnoyers et al. 2005). Some approaches exploit indexes to reduce the time to locate distributed sensor data (Desnoyers et al. 2005). However, in highly dynamic situations including mobile entities, such approaches can severely limit the flexibility since the indexes needs to be updated frequently, offsetting the benefit of maintaining indexes. Further, data rate from the underlying sensors can be highly unpredictable. For instance, in an emergency situation, the system will be subject to a massive load increase. Potential mobility of entities can also incur unpredictable workload changes to each upper-tier devices since the number of sensors that each upper-tier device covers can change dramatically. This unpredictable workload changes can severely congest or overload the devices, incurring additional unpredictable delays in processing queries.

1.2 Our contribution

To deal with the problem of the emerging large-scale sensor network applications, this paper presents a novel data abstraction layer, called PRIDE (Predictive Replication In Distributed Embedded devices). PRIDE enables transparent access to global

sensor data in a timely, flexible, and robust manner in highly dynamic environments of emerging distributed real-time applications interacting with pervasive sensors.

The contributions of this paper are as follows:

1. *A predictive replication-based 2-tier architecture:* Data replication has been a key technique that enables each participating entity to share data and obtain a timely understanding of the global status without the need for a central server (Son 1988; Gray et al. 1996; Cook et al. 2002; Padmanabhan et al. 2008). To this end, we propose a replication-based 2-tier architecture, in which data from the sensor tier is replicated among upper-tier devices. Our replication-based approach provides several key advantages. First, since global data from the sensor tier are immediately available at each upper-tier device, spatio-temporal queries on global sensor data can be answered in a timely manner without remote communication. Further, since all upper-tier devices share the same states regarding underlying sensors, the data source tier (sensors) does not need to be tightly coupled with the upper-tier devices. This loose coupling between the tiers implies that the mobility of sensor nodes as well as the upper-tier devices can be easily accommodated.

   Despite these advantages, replication can impose a significant burden on the system (Gray et al. 1996). To mitigate such inherent overheads of replication, a predictive replication scheme is proposed, in which the models of sensor streams are replicated at upper-tier devices, instead of data themselves. Once a model for a sensor stream is created at its primary device and its peer devices, the updates from the sensor are replicated to peer devices only if the prediction from the model is not accurate enough. This model-driven approach provides the timeliness and reliability of sensor data at the upper tier devices since they can locally predict the current as well as future states of physical processes using the models without actual communication with the underlying sensor tier.
2. *A feedback controller to provide robustness against unpredictable workloads:* Even though PRIDE mitigates the high overhead of replication via the predictive replication scheme, upper-tier devices still can be overloaded or congested. During critical situations, the data rates from the sensor tier can significantly increase and exceed system capacity. If no corrective action is taken, queues will form and the latencies of queries will increase without bound. To prevent such situation, we extend the model-driven replication to the problem of guaranteeing Quality-of-Service (QoS) by introducing feedback control of the quality of sensor data. A target CPU utilization at each upper-tier device is defined as a primary QoS metric since, in soft real-time systems, the scheduler will use any cycles that are saved by the utilization control to improve the timeliness of tasks or queries in the application. A formal feedback control approach is applied to guarantee the QoS by dynamically adapting the accuracy bounds of models. By combining feedback control of QoS into the predictive replication scheme, PRIDE can achieve high predictability.
3. *Implementation and evaluation:* The proposed predictive replication mechanism and feedback control approach can be integrated into typical data managers, supporting diverse types of spatial and temporal queries. To show the viability of the proposed approach, we integrate our approach into Berkeley DB (http://www.oracle.com 2008), a popular embedded database. PRIDE is evaluated extensively

on an emulation testbed composed of Nokia N810 Internet tablets (http://www.nseries.com/ 2008), a cluster computer, and a fire simulator from NIST (http://fast.nist.gov/ 2008). The testbed emulates collaborative search-and-rescue tasks in a building fire with realistic workloads. Based on the prototype implementation, we investigate the system performance attributes such as communication/computation loads, timeliness of query processing, handling of mobility, energy efficiency, and robustness. Our evaluation results demonstrate that PRIDE can achieve timely sensor data sharing among devices in a highly robust and predictable manner.

The rest of this paper is organized as follows. Section 2 presents the overview of PRIDE. Section 3 discusses the details of the predictive replication scheme and the QoS enforcement mechanism in PRIDE. Section 4 presents the query processing mechanism of PRIDE. Section 5 discusses our prototype implementation and testbed, and Sect. 6 presents our experimental results. The related work is discussed in Sect. 7. We present conclusions and future work in Sect. 8.

2 Overview of PRIDE

2.1 System model

PRIDE envisions 2-tier sensor network systems with a *sensor tier* and a *storage tier* as shown in Fig. 1. The sensor tier consists of a large number of cheap and simple sensors; $S = \{s_1, s_2, \ldots, s_n\}$, where s_i is a sensor. Sensors are assumed to be highly constrained in resources, and perform only primitive functions such as sensing and multi-hop communication without local storage. Sensors stream data or events to a nearest storage node. These sensors can be either stationary or mobile; e.g., sensors attached to a firefighter are mobile.

The storage tier consists of more powerful devices such as PDAs, smartphones, and base stations; $D = \{d_1, d_2, \ldots, d_m\}$, where d_i is a storage node. These devices are relatively resource-rich compared with sensor nodes. However, these devices also have limited resources in terms of processor cycles, memory, power, and bandwidth. Each storage node supports multiple radios; an 802.11 radio to connect to a wireless mesh network of storage nodes and a 802.15.4 to communicate with underlying sensors. Storage nodes are supposed to form an ad-hoc mesh network that is efficient for

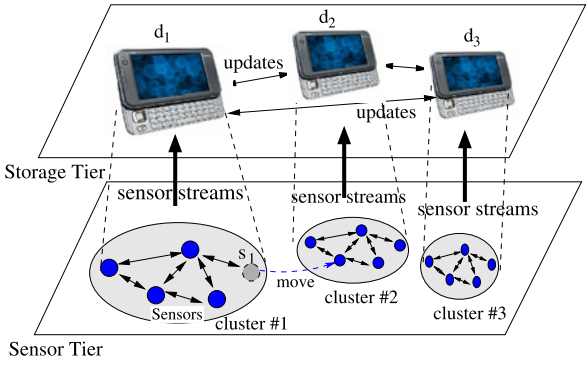


Fig. 1 A collaborative application on a 2-tier sensor network

broadcasting (Fife and Gruenwald 2003) or multicasting (de Morais Cordeiro et al. 2003) of messages. However, constructing and maintaining an ad-hoc mesh network among mobile devices is out of scope of this paper, and readers are referred to Akyildiz and Wang (2005), Raniwala and cker Chiueh (2005).

Each storage node d_i is a *primary storage node* of a set of sensors in vicinity $V(d_i)$, where $V(d_i) \subset S$, and provides in-network storage for $V(d_i)$. For the full coverage of all sensors S, the union of $V(d_i)$ is supposed to be equal to S. The primary storage node of a sensor is determined implicitly by receiving sensor updates directly from the sensor; the remaining storage nodes receiving sensor updates indirectly from the primary storage node of the sensor automatically become the peer storage nodes. This implies that the mechanisms of PRIDE at the storage tier is independent of the policies at the underlying sensor tier. Hence, PRIDE does not mandate a specific clustering and routing mechanism at the sensor tier. For example, a sensor can choose its primary storage node based on Euclidean distance, stability of radio channels, remaining energy at sensor nodes in vicinity, etc.

In PRIDE, all nodes in the storage tier are homogeneous in terms of their roles; no asymmetrical function is placed on a sub-group of the nodes. All or part of the nodes in the storage tier form a *replication group* R to share the data from underlying sensors. In this paper, we assume $R = D$. When the size of R is too large for the full replication of sensor data, they should be partitioned as in Mathiason et al. (2008). The focus of this paper is to make each replication group more scalable. When there exist multiple replication groups, sophisticated group communication schemes such as Mohapatra et al. (2004) should be introduced. But they are out of scope of this work. All devices joining the ad-hoc mesh network is supposed to be a member of the replication group, and messages are broadcast via broadcasting or multicasting protocols. Once a node joins the replication group, updates from its local sensors are propagated to peer nodes; conversely, the node can receive updates from remote sensors via peer nodes.

In the remainder of this paper, a *node* refers to a *storage node* if it is not explicitly stated.

2.1.1 Mobility of entities

Since all nodes are homogeneous and share same states regarding the underlying sensors, the sensor tier and the storage tier are loosely coupled. This loose coupling is one of the key benefits provided by PRIDE's replication-based approach. If the tiers are tightly coupled, a complex group management and hand-off procedure are required to handle the mobility of entities (Abdelzaher et al. 2004). Even though PRIDE does not mandate any particular group management and hand-off mechanisms, PRIDE can easily handle the mobility of both sensors and storage nodes. For example, in Fig. 1, when a sensor node s_1 changes its belonging cluster from cluster #1 to cluster #2, no additional handover procedure is required at the storage tier since both storage node d_1 and d_2 maintain the same state regarding s_1. Under the assumption that sensor nodes forward data streams to a nearest storage node, storage node d_2 can simply broadcast replication messages to its peer nodes from the moment sensor streams are received from the sensor s_1. This procedure is depicted in Fig. 2. It should be noted

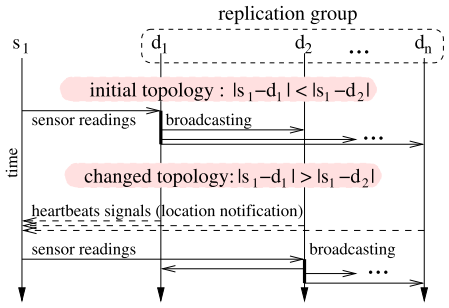


Fig. 2 An example of hand-off procedure

that no communication load is incurred at the storage tier to reflect the topology change at the sensor tier.

2.2 Usage model

Applications at each storage node are linked to the PRIDE data abstraction layer. Applications issue queries to the underlying PRIDE data abstraction layer either autonomously, or by simply forwarding queries from external users. In the search-and-rescue task example, each storage node, or firefighter's PDA, serves as both in-network data storage for nearby sensors and a device to run autonomous real-time applications for the mission; the applications at each PDA collect data by issuing queries periodically to the underlying PRIDE layer and analyzing the situation to report results to the firefighter.

PRIDE targets soft real-time applications, in which the tasks and queries have soft deadlines. Due to the inherent uncertainties of wireless communications, most real-time applications with wireless communications medium are inherently soft real-time applications. Further, since we assume mobile devices such as PDAs and smartphones that typically run by non-real-time operating systems (e.g., Android and Windows Mobile), it is almost impossible to provide hard real-time guarantees on the deadlines of tasks or queries. Instead of guaranteeing deadlines directly, PRIDE supports a predictable scheduling of the queries or tasks at the application layer by trying to minimize both computation and communication uncertainties in processing queries. The replication-based tiered architecture of PRIDE eliminates unpredictable remote communication when queries are processed. Further, the usage of CPU resource is controlled to prevent overloads against unpredictable workload changes. The details of the two mechanisms are discussed in Sect. 3.

2.2.1 Supported query types

PRIDE is characterized by the queries that it supports. PRIDE supports both temporal queries on each individual sensor stream and spatial queries on current global data. Temporal queries on sensor s_i's historical data can be answered using the model for s_i. An example of temporal query is *"What is the value of sensor s_i 5 minutes ago?"* For spatial queries, each storage node provides a snapshot on the entire set of underlying sensors (both local and remote sensors.) The snapshot is similar to a *view* in database systems. Using the snapshot, PRIDE provides traditional data organization and access methods for efficient spatial query processing. The access methods

can be applied to any attributes, e.g., sensor value, sensor ID, and location; therefore, *value-based* queries can be efficiently supported. Basic operations on the access methods such as insertion, deletion, retrieval, and the iterating cursors are supported. Special operations such as *join cursors* for join operations are also supported by making indexes to multiple attributes, e.g., temperature and location attributes. This join operation is required to efficiently support complex spatial queries such as *"Return the current temperatures of sensors located around the firefighter #6 within a radius of 10 meters."*

2.2.2 Quality of data and quality of service

In PRIDE, quality of data is defined in terms of the precision bound of sensor data. Since there exists a trade-off between the quality of data and the increased workloads, it is necessary to prevent the overload and subsequent message delays at each node while satisfying the given data quality goals. To this end, the primary QoS metric in this paper is the CPU utilization bound at each node. With CPU load control, it is assumed that any cycles that are recovered as a result of control in the PRIDE layer are used sensibly by the scheduler in the application layer to relieve the overloads or to save power (Lu et al. 2005; Tatbul et al. 2003). It can also enhance system robustness by providing overload protection against workload fluctuation.

At each node, the system specification $\langle U, \delta_{max} \rangle$ consists of a utilization bound U and the precision specification δ_{max}. The desired utilization, or QoS goal, $U \in [0..1]$ gives the required CPU utilization. The precision specification δ_{max} denotes the maximum tolerable precision bound. The precision bound defines the quality of data and subsequent query results. Note there is no lower bound on the precision as in general users require a precision bound as small as possible (as long as the system is not overloaded.)

3 Quality-aware predictive replication in PRIDE

This section discusses the architecture of PRIDE, the predictive model-based replication mechanism, and the adaptive control of sensor data precision to guarantee the desired QoS.

3.1 System architecture

The architecture of PRIDE data abstraction layer is shown in Fig. 3. Since all storage nodes in PRIDE are homogeneous, each storage node has the same architecture. The layer consists of three key components: (i) *filter & prediction engine*, which is responsible for sensor stream filtering, model update, and broadcasting of updates to peer nodes, (ii) *query processor*, which handles queries on spatial and temporal data by using a snapshot and temporal models, respectively, and (iii) *feedback controller*, which determines proper precision bounds of data for overload protection and robustness. Three components interact closely with each other.

Each data stream from a sensor has a corresponding model both at its primary node and peer nodes. Updates from a sensor are first delivered to its primary node's

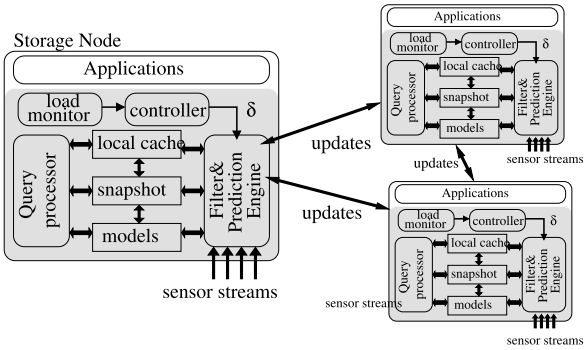


Fig. 3 The architecture of PRIDE data abstraction layer (*gray boxes*)

filter & prediction engine. The *filter & prediction engine* determines if the updates need to be broadcast to peer nodes. New updates are propagated to peer nodes only if the observed sensor values deviate from the model's predicted value by more than a specified precision bound δ. The precision bound δ is set dynamically to meet the desired QoS—the CPU load U. The load monitor periodically reports the current CPU load to the *feedback controller*, which in turn calculates the CPU utilization error, i.e., the difference between the desired CPU load and the measured CPU load at every sampling interval. Based on the error, the *feedback controller* determines if how the precision bound δ should be adjusted for the next sampling interval. To maintain the consistency among the storage nodes in the replication group, the minimum of the local δs from the nodes is taken as the global precision bound for the next sampling interval. The *query processor* processes the query requests from either internal applications or external users. Instead of directly fetching data from sensors, The *query processor* maintains a snapshot which reflects the global status of the monitored physical processes. Each data object in the snapshot is refreshed from corresponding model indirectly to maintain the freshness of data.

3.2 Filter and prediction engine

The goals of filter & prediction engine are to filter out updates from local sensors using models, and to synchronize models at each storage node. The premise of using models is that the physical phenomena observed by sensors can be captured by models and a large amount of sensor data can be filtered out using the models. In PRIDE, when a sensor stream s_i is covered by PRIDE replication group R, each storage node in R maintains a model m_i for s_i. Therefore, all storage nodes in R maintain a same set of synchronized models, $M = \{m_1, m_2, \ldots, m_n\}$, for all sensor streams in underlying sensor tier. Each model m_i for sensor s_i is synchronized at run-time by s_i's current primary storage node (note that s_i's primary node can change during run-time because of the network topology changes either at sensor tier or storage tier).

Algorithms 1 and 2 show the basic framework for model synchronization at a primary node and peer nodes, respectively. In Algorithm 1, when an update v is received from sensor s_i to its primary storage node d_j, the model m_i is looked up, and a prediction is made using m_i. If the gap between the predicted value from the model, $\hat{v}$, and the sensor update v is less than the precision bound δ (line 2), then the new data

Algorithm 1: OnUpdateFromSensor

```
   Input: update v from sensor s_i
1  v̂ = prediction from model for s_i;
2  if |v̂ − v| ≥ δ then
3      broadcast v to peer storage nodes;
4      update data for s_i in the snapshot;
5      update model m_i for s_i;
6      store to cache for later temporal query processing;
7  else
8      discard v (or store for logging);
9  end
```

Algorithm 2: OnUpdateFromPeer

```
   Input: sensor observation v from peer d_x
1  update data for s_x in the snapshot;
2  update model m_x for s_x;
3  store to cache for later temporal query processing;
```

is discarded (or saved locally for logging.) This implies that the current models (both at the primary node and the peer nodes) are precise enough to predict the sensor output with the given precision bound. However, if the gap is bigger than the precision bound, this implies that the model cannot capture the current behavior of the sensor output. In this case, m_i at the primary node is updated and v is broadcast to all peer nodes (line 3). In Algorithm 2, as a reaction to the broadcast from d_j, each peer node receives a new update v and updates its own model m_i with v. The value v is stored in local caches at all nodes for later temporal query processing.

As shown in the Algorithms, the communication among nodes happens only when the model is not precise enough.

3.2.1 Models, estimation, and prediction

Several distinctive requirements guide the choice of modeling technique in PRIDE. First, the computation and communication costs for model maintenance should be low since PRIDE handles a large number of sensors (and corresponding models for each sensor) with collaboration of multiple nodes. The cost of model maintenance linearly increases to the number of sensors. Second, the parameters of models should be obtained without an extensive learning process, because many collaborative real-time applications, e.g., a search-and-rescue task in a building fire, are short-term and deployed without previous monitoring history. A statistical model that needs extensive historical data for model training is less applicable even with their highly efficient filtering and prediction performance. Finally, the modeling should be general enough to be applied to a broad range of applications. Ad-hoc modeling techniques for a particular application cannot be generally used for

other applications. Since PRIDE is a data abstraction layer for wide range of collaborative applications, the generality of modeling is important. To this end, we choose to use a state-space modeling technique and Kalman filters (Gelb 1974; Jain et al. 2004). In Jain et al. (2004), it is show that Kalman filters can be applied to a broad range of applications.

In PRIDE, a physical process, such as temperature, measured by a sensor at a location at time k is represented in a state vector having 2 state variables, $\mathbf{x}_k = [x\ \frac{dx}{dt}]^T$, where x is the current sensor value at time k and $\frac{dx}{dt}$ is the derivative of x with respect to time t. We may need an additional state variable, such as $\frac{d^2x}{dt^2}$, to describe the physical process more accurately. However, the cost of model maintenance increases according to the number of state variables, and, hence, the current implementation of PRIDE choose to use 2 state variables for each sensor data object. We believe that the dynamics of most simple physical processes such as temperature, light intensity, pressure, etc., can be described using these 2 state variables.

The estimation of current and future states with the models follows the standard procedures of Kalman filtering technique. As a new measurement is available from a sensor, the true state of the sensor is estimated using the previous prediction from the model and weighted prediction error. Based on the prediction error, the model is updated and the estimation of future states is made from the update model. Unlike batch estimation techniques, no history of observations is required for Kalman filters. The accuracy of the parameters of Kalman filters and state variables improves gradually and adapts to the changes by having more sensor measurements. Moreover, since the parameters of models at all storage nodes are synchronized simply by exchanging an sensor update when the measured value deviates from what its model predicts, no further communication is required to synchronize the models. For detailed discussion on Kalman filters and its estimation procedures readers are referred to Gelb (1974) and Jain et al. (2004).

3.2.2 Impact of model inaccuracy

According to our scheme, the communication load incurs only when models are not accurate enough. In this section, we introduce intentional errors to show the impact of the model inaccuracy.

For the evaluation, a physical process, which has a non-negligible 3rd component, $\frac{d^2x}{dt^2} = \alpha$, is modeled with states having 2 components in PRIDE, $\mathbf{x} = [x\ \frac{dx}{dt}]^T$. When the second component $\frac{dx}{dt}$ is c at a given instant, the expected change of x after t in PRIDE is

$$\Delta\hat{x} = t \times \frac{dx}{dt} = ct. \tag{1}$$

On the contrary, the true change of x is

$$\Delta x = \int_0^t \frac{dx}{dt} dt = \int_0^t (\alpha t + c) dt. \tag{2}$$

In PRIDE, updates and communication occur only when $|\Delta\hat{x} - \Delta x| \geq \delta$,

$$|\Delta\hat{x} - \Delta x| = \left| ct - \left(\frac{\alpha t^2}{2} + ct \right) \right| \geq \delta. \tag{3}$$

Hence, the expected update rate r is

$$r = \frac{1}{t} \leq \sqrt{\frac{\alpha}{2\delta}} = \sqrt{\frac{1}{2\delta}\frac{d^2x}{dt^2}}. \tag{4}$$

This implies that the increase of update rates is proportional to the square root of the third term. For example, if the precision bound δ is 1 meter when we measure the moving distance of a vehicle, the model inaccuracy incurs 1.56 additional updates per second since the acceleration of a typical starting vehicle is known to be less than 4.9 m/s^2. Furthermore, the effect of the third component is transient for many physical processes; the third term, or $\frac{d^2x}{dt^2}$, approaches to 0 quickly after starts. We provide the data quality adaptation mechanism, which will be discussed in the following section, to handle such transient and bursty workloads incurring from the model inaccuracy, instead of using a more complex model, which incurs constant overheads.

3.3 Adaptive data quality control

Figure 4 shows the temperature changes observed by a sensor in a building fire, and how closely PRIDE estimates this temperature change when different precision bounds, $\delta = 1°$, $5°$, and $10°$, are applied. As the precision bound is getting bigger,

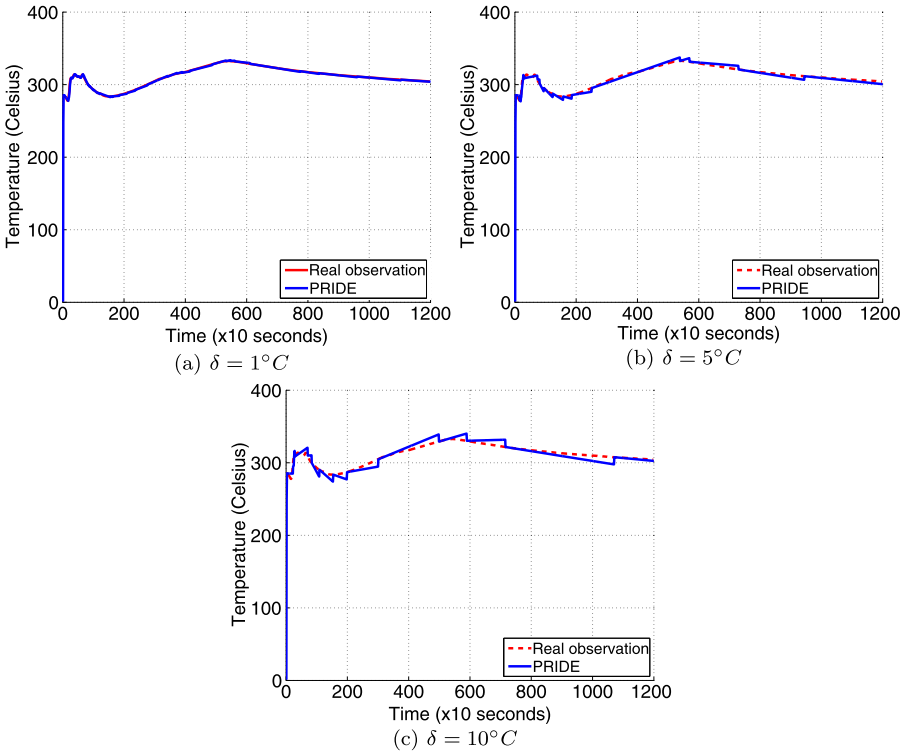


Fig. 4 Varying data precision

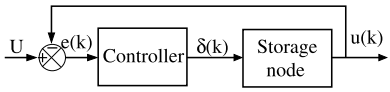


Fig. 5 The feedback control loop

the gap between the real state of the sensor (dashed lines) and the current value at the PRIDE (solid lines) increases. In the solid lines, the discontinued points are where the gap between the model prediction and the real measurement from the sensor are bigger than the precision bound, and subsequent communication is made among storage nodes for model synchronization. For applications and users, maintaining the smaller precision bound implies having a more accurate view on the monitored situation. However, the overhead also increases as we have the smaller precision bound, potentially incurring the delays in processing queries and communication messages. Given the unpredictable data arrival rates and resource constraints, it is a challenging task to determine a proper precision bound at runtime.

To this end, to guarantee the system specification without a priori knowledge of the workload or accurate system model, we apply feedback control which has shown to be very effective for a large class of computing systems exhibiting unpredictable workloads and model inaccuracies (Hellerstein et al. 2004).

3.3.1 Local feedback control to guarantee the system specification

The overall feedback control loop at each storage node is shown in Fig. 5. Let T is the sampling period. The utilization $u(k)$ is measured at each sampling instant $0T, 1T, 2T, \ldots$ and the difference between the target utilization and $u(k)$ is fed into the controller. Using the difference, $e(k)$, the controller computes a local precision bound $\delta(k)$ such that $u(k)$ converges to U.

The first step for local controller design is modeling the target system (storage node) by relating $\delta(k)$ to $u(k)$. We model the relationship between $\delta(k)$ and $u(k)$ by using profiling and statistical methods (Hellerstein et al. 2004). As shown in Sect. 6.5.1, $\delta(k)$ has higher impact on $u(k)$ as the size of the replication group increases, hence, a single linear model cannot capture the relation between $\delta(k)$ and $u(k)$. To address this problem, we employ the *gain scheduling* technique. In gain scheduling, better controller performance can sometimes be achieved by constructing separate controllers from workload-specific models and then switching to the appropriate controllers as workload changes. To this end, a set of first-order linear models are identified under different sizes of replication groups, R:

$$u(k) = a \times u(k-1) + b \times \delta(k-1), \tag{5}$$

where a and b are model parameters, $\delta(k)$ is a system input, and $u(k)$ is a system output. For computational convenience, the models are z-transformed to *transfer functions*, $G_{|R|}(z) = \frac{b}{z-a}$, where $|R| \in \{4, 8, 16, 32, \ldots\}$. With z-transform, we can easily extract key properties of models and controllers such as the settling time of a system. Table 1 shows the transfer functions for different sizes of replication groups. For instance, $G_{16}(z)$ captures the system behavior when the size of a replication group is around 16. It should be noted that a is negative, meaning that a larger δ decreases the

Table 1 Transfer functions for varying sizes of replication groups

$\|R\|$	4	8	16	32
$G_n(z)$	$\frac{-0.646}{z-0.605}$	$\frac{-0.816}{z-0.672}$	$\frac{-1.112}{z-0.604}$	$\frac{-1.499}{z-0.606}$

Algorithm 3: PrecisionBoundControl

Input: myid: my storage id number
1 /* Get local δ. */
2 measure $u(k)$ from monitor;
3 calculate $\delta_{myid}(k)$ from local controller;
4 **foreach** *peer node* d *in* $R - \{d_{myid}\}$ **do**
5 /* Exchange local δs. */
6 /* Use piggyback to save communication cost. */
7 send $\delta_{myid}(k)$ to d;
8 receive $\delta_i(k)$ from d;
9 **end**
10 /* Get the final global δ. */
11 $\delta_{global}(k) = max(\delta_i(k))$, where $i \in R$;

CPU load. Further, the gap between a and b is getting greater as the size of replication groups increases. This implies that the system is more sensitive to the change of δ accordingly as the size of replication groups grows.

After the modeling, we design a controller for the model. The goal of the controller is to ensure that the measured CPU utilization, $u(k)$, is equal to the target utilization, U. We have found that a proportional integral (PI) controller (Hellerstein et al. 2004) is sufficient in terms of providing a zero steady-state error, i.e., a zero difference between $u(k)$ and the target utilization bound. Further, a gain scheduling technique (Hellerstein et al. 2004) have been used to apply different controller gains for different size of replication groups. For instance, the gain for $G_{32}(z)$ is applied if the size of a replication group is bigger than 24 and less than or equal to 48. Due to space limitation we do not provide a full description of the design and tuning methods. For details on controller design and tuning, readers are referred to Hellerstein et al. (2004).

3.3.2 Coordination among replication group members

If each node independently sets its own precision bound, the net precision bound of data becomes unpredictable. For example, at node d_j, the precision bounds for local sensor streams are determined by d_j itself while the precision bounds for remote sensor streams are determined by their own primary storage nodes.

PRIDE takes a conservative approach in coordinating storage nodes in the group. As Algorithm 3 shows, the global precision bound for the kth period is determined by taking the maximum from the precision bounds of all nodes in the replication group. The global precision bound at kth sampling period, is the precision bound applied

for all storage nodes in the replication group for the period. We choose the greatest value among all precision bounds from each local control loop since a node that has the greatest precision bound is the most overloaded node in period k. By setting the global precision bound to the most overloaded node's, we can ensure that congestion at the most overloaded node is reduced.

One of the most important properties of a feedback control system is stability. However, the stability analysis of distributed feedback control loops is possible only in limited settings (Wang et al. 2007). Hence, we show the stability of PRIDE's feedback control mechanism in an empirical manner in Sect. 5.

4 Query processing in PRIDE

The query processor of PRIDE supports both temporal queries and spatial queries. We plan to extend PRIDE to support spatio-temporal queries.

4.1 Spatial queries

Each storage node maintains a snapshot for all underlying local and remote sensors to handle queries on global spatial data. Each element (or data object) of the snapshot is an up-to-date value from the corresponding sensor. The snapshot is dynamically updated either by new measurements from sensors or by models.[1] The Algorithm 1 (line 4) and Algorithm 2 (line 1) show the snapshot updates when a new observation is pushed from a local sensor and a peer node, respectively. As explained in the previous section, there is no communication among storage nodes when models well represent the current observations from sensors. When there is no update from peer nodes, the freshness of values in the snapshot deteriorate over time. To maintain the freshness of the snapshot even when there is no updates from peer nodes, each value in the snapshot needs to be updated by its local models either periodically or on-demand. On-demand updates are more suitable if queries do not have timing constraints and the execution of the queries are rare. With on-demand updates, resources such as CPU time can be saved. However, since PRIDE targets applications that execute queries often for the timely situation-awareness, on-demand updates can be another source of delays in processing queries. To this end, PRIDE performs periodic updates of snapshots. The overhead of periodic updates of snapshots is quantified in Sect. 6.

Each storage node can estimate the current state of sensor s_i using models without communication to the primary storage node of s_i. The period of update of data object i for sensor s_i is determined, such that the precision bound δ is observed. Intuitively, when a sensor value changes rapidly, the data object should be updated more frequently to make the data object in the snapshot valid. In the example of Sect. 3.1, the period can be dynamically estimated as follows:

$$p[i] = \delta / \frac{dx}{dt}. \tag{6}$$

[1] Note that the data structures for the snapshot such as indexes are also updated when the value of the snapshot is updated.

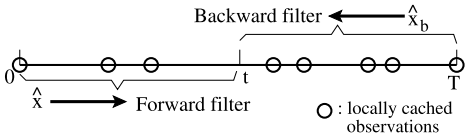


Fig. 6 Smoothing for temporal query processing

The $2 \times \delta / \frac{dx}{dt}$ is the *absolute validity interval (avi)* before the data object in the snapshot violates the precision bound, which is $\pm\delta$. The update period should be as short as the half of the *avi* to make the data object fresh (Ramamritham et al. 2004).

Since each storage node has an up-to-date snapshot, spatial queries on global data from sensors can be efficiently handled using local data access methods (e.g., B+ tree) without incurring further communication delays.

4.2 Temporal queries

Historical data for each sensor stream can be processed in any storage node by exploiting data at the local cache and linear smoother (Gelb 1974). Unlike the estimation of current and future states using one Kalman filter, the optimized estimation of historical data (sometimes called *smoothing*) requires two Kalman filters, a forward filter $\hat{\mathbf{x}}$ and a backward filter $\hat{\mathbf{x}}_b$. Smoothing is a data processing scheme that uses all measurements between 0 and T to estimate the state of a system at a certain time t, where $0 \leq t \leq T$ (see Fig. 6.) The smoothed estimate $\hat{\mathbf{x}}(t|T)$ can be obtained as a linear combination of the two filters as follows.

$$\hat{\mathbf{x}}(t|T) = \mathbf{A}\hat{\mathbf{x}}(t) + \mathbf{A}'\hat{\mathbf{x}}(t)_b, \tag{7}$$

where $\mathbf{A}$ and $\mathbf{A}'$ are weighting matrices. For detailed discussion on smoothing techniques using Kalman filters, the reader is referred to Gelb (1974).

5 Implementation and testbed

We have implemented a prototype of PRIDE on a multi-tier sensor network testbed. In this section, we describe the issues in the implementation of PRIDE and explain the details of the testbed.

5.1 Implementation details

PRIDE utilizes an extension of Berkeley DB (http://www.oracle.com 2008), which is a popular open-source embedded database. Unlike traditional database systems, Berkeley DB is embeddable to an application; Berkeley DB is linked to an application (or application infrastructure) and provides robust storage features such as diverse access methods, ACID transactions, recovery, locking, and multi-threading for concurrency. PRIDE exploits these features of Berkeley DB and extends them by providing predictive replication, snapshot management, and dynamic data precision control. Since PRIDE is intended for use in open systems, it has been built on top of a POSIX-based standard operating system (http://standards.ieee.org/regauth/posix

2009), instead of a specialized real-time operating system. PRIDE, except the original Berkeley DB, is composed of about 7000-line C codes. The binary size of PRIDE is less then 40 KB. Hence, we believe that PRIDE is applicable to a wide range of embedded devices including smart-phones.[2]

5.1.1 Snapshot management

Snapshots of PRIDE are ordinary database files of Berkeley DB. By using ordinary database files, PRIDE can exploit diverse built-in functionalities of Berkeley DB such as indexing. However, since database files are static entities, as discussed in Sect. 3.2, they need to be updated periodically to maintain the freshness of their data. In PRIDE, each data record has an implicit data field, which contains its *update period*. A timer is set to the next expiration time, which is *current time* + *update period*. On the expiration of the timer, the update period is reevaluated using (6), and the timer is set to the next expiration time. However, since potentially a large number of data records exist in database files, setting a separate timer for each data record can degrade the system performance to handle lots of timer interrupts; each timer interrupt incurs context-switching, which can be a major source of CPU load. Instead, PRIDE maintains a priority queue, which orders all timer interrupt requests for snapshot updates, and coalesces requests having close expiration times. For example, PRIDE ignores sub-second units of each timer request, and coalesces them to one timer interrupt request. When a timer expires, all data records corresponding to the coalesced timer request are updated. We believe that one context switching per second to handle timer interrupt will incur virtually no overhead. However, the detailed effects of coalescing on the performance is left for future work.

5.1.2 Feedback control

PRIDE has a dedicated control task that periodically monitors and adjusts the data precision as discussed in Sect. 3.3. The PRIDE control task is a *p-thread* belonging to the real-time class and is periodically activated every 5 seconds. The communication overhead affects the choice of sampling interval because too short sampling periods incur communication overhead for storage nodes to coordinate a global precision bound as discussed in Sect. 3.3. Conversely, if the sampling interval is too long, the speed of control will be slow. Our experiment showed that the sampling interval of 5 seconds makes good trade-offs between the two conflicting requirements.

The CPU load is the system output in PRIDE, and the effectiveness of feedback control depends on the preciseness of CPU load information. In Linux underlying PRIDE, CPU load average information via proc file system (/proc/loadavg) is based on queuing-theoretic estimation by observing the length of run queue in the CPU scheduler. This information does not provide fine-grain information on how much CPU time has been used by a specific set of tasks. Hence, instead of using the load average information, we measure the number of clock ticks that actually have been used by PRIDE both in user and kernel modes. This information is available via /proc/stat interface in Linux.

[2]The minimum footprint of PRIDE including Berkeley DB is 540 KB.

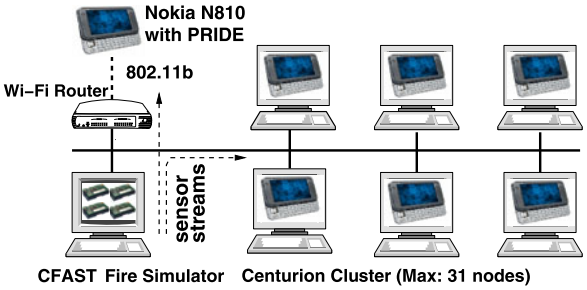


Fig. 7 PRIDE testbed

5.2 Emulation testbed

A collaborative search-and-rescue scenario in a building fire from (http://fire.me.berkeley.edu/ 2008) is adapted, and emulated on our emulation testbed. In this scenario, PDAs carried by a team of firefighters are storage nodes collecting data from sensors in a building via wireless communication.

In the testbed, the storage tier employs one Nokia N810 Internet tablet (http://www.nseries.com/ 2008) and a Centurion cluster machine as shown in Fig. 7. The N810 device is equipped with 400 MHz TI OMAP processor, 128 MB RAM, 256 MB flash memory, 802.11b Wi-Fi radio, and runs *Maemo*, which is a modified version of GNU/Linux slimmed down for mobile devices.[3] Since the number of available storage nodes is limited, the Centurion cluster with 32 computing nodes are used to enable a large-scale evaluation. Each node of the cluster is equipped with two 1.5 GHz AMD Opteron processors, 2 GB RAM. The N810 device and each node of the Centurion cluster emulate one storage node. Hence, 32 firefighters in maximum are emulated by storage nodes in the testbed (one N810 device and up to 31 Centurion cluster nodes). The N810 device and the Centurion cluster are connected via 802.11b Wi-Fi. The broadcast among the emulated storage nodes is implemented by sending a separate message to each storage node. All emulated storage nodes, either on N810 or on the Centurion cluster, perform homogeneous functionality. However, the real measurements of CPU utilization, energy consumption, and the query latency are performed in the N810 device. We believe that the N810 device represents emerging mobile computing platforms, which are expected to interact with ubiquitous sensors in the near future.

In the sensor tier, sensor streams are generated by NIST CFAST (The Consolidated Model of Fire and Smoke Transport) fire simulator (http://fast.nist.gov/ 2008). CFAST is a two-zone fire model used to calculate the evolving distribution of smoke, fire gases and temperature throughout compartments of a building during a fire. Using CFAST simulator, a wide-range of fire scenarios can be simulated in detail by configuring the input parameters, which include the geometry of the compartments and the connections between these compartments, the initial fire source and burning objects in the compartments, flow vents, and floor/wall materials. In the testbed, traces are generated from CFAST simulator off-line for repeatability and scalability of experiments. Each trace corresponds to the history of temperature changes at

[3]Maemo is based on GNU/Linux 2.6.21 kernel and compliant with POSIX standards.

Table 2 Testbed settings

Parameter	Value
# of storage nodes ($\|R\|$)	32 in maximum (One N810 and 31 cluster nodes)
Measurement device	N810
Networking	802.11b Wi-Fi
Type of sensors	Temperature
# of sensors ($\|S\|$)	4480 in maximum (80–140 per storage node)
Sensor measurement report interval	1 second
Query issue interval	0.5 second
Query operation	Accesses data ($\|S\| \times 0.1$) times
Query access pattern	50–50 (random) and 90–10

a specific location in the modeled building. For large scale experiments, additional sensor streams can be made by time-shifting the original traces. In the runtime of each simulation, these traces are replayed and sent to storage nodes by one node of the Centurion cluster with 1 second interval as shown in Fig. 7.

Each storage node is expected to be a primary storage node of up to 140 sensors; hence, 4,480 sensor streams in maximum are covered by 32 firefighters. In the simulated search-and-rescue scenario, real-time tasks running on each storage node check the status of the building such as the possibility of collapse, explosion, the existence of safe retreat path, etc. For the emulation of such real-time tasks, a query is regularly issued by an application at each storage node on every 0.5 second period.[4] The queries access sensor data in the snapshot according to x–y data access pattern. In x–y access scheme, x % of the data accesses are directed to y % of the data in the PRIDE layer. For instance, with 90–10 access pattern, 90 % of the data accesses are directed to 10 % of the data in the PRIDE layer, thus, incurring data contention on 10 % of the entire data. To demonstrate that PRIDE is robust to access pattern changes, we tested our approach by applying 2 different x–y access patterns; 90–10 and 50–50. The number of data accesses per query is proportional to the total number of sensors. The applications can use raw data returned from the queries for further analysis and decision making. However, further processing in the application layer is not modeled in the evaluation. The summary of the settings of the testbed is shown in Table 2.

6 Evaluation

In this section, the goal and background of the experiments are discussed and results are presented.

[4]Real-time queries for firefighters can be invoked on a per-second basis (Jiang et al. 2004).

Table 3 Tested approaches

Full	Full replication
Approx-caching	Replication using recently cached values
PRIDE	Model-based replication at the storage tier
PRESTO	Model-based communication between the tiers; No sensor data replication at the storage tier
PRIDE + PRESTO	Model-based communication between the tiers; Model-based sensor data replication at the storage tier

6.1 Performance evaluation goals

The objectives of the performance evaluation are (1) to determine the suitability of the light-weight replication-based 2-tier architecture of PRIDE in terms of its scalability, timeliness of query processing, handling of mobility, energy consumption, and data quality and (2) to determine if the presented algorithms can provide guarantees on target CPU loads according to a QoS specification. For the first objective, we have studied and evaluated the behavior of the algorithms under various conditions, where a set of parameters have been varied. Through Experiments #1 and #2, the data quality controller in PRIDE is turned off during the evaluation to assess the efficiency of the predictive replication alone.

The second objective is investigated in Experiment #3 by comparing PRIDE's performance adaptation while its controller is turned on and off.

6.2 Baselines

For performance comparisons, we consider five approaches shown in Table 3. In *Full*, all updates from sensor streams are fully replicated to peer storage nodes in the replication group. This is a typical replication approach for most commercial distributed databases. *Approx-Caching* is a value-driven approach, in which a storage node broadcasts updates from sensors only if the difference between current data and last broadcast data is larger than a threshold (δ). This approach is similar to the algorithm in Olston et al. (2001). *PRESTO* represents a state-of-the-art 2-tier architecture (Li et al. 2006; Desnoyers et al. 2005; Gnawali et al. 2006), in which the two tier reduce the load of communication by exploiting models. In the original PRESTO (Li et al. 2006), seasonal ARIMA model is exploited for the communication between the tiers. However, since the performance of specific models is not the focus of this paper, we use the Kalman filtering technique, which is used in PRIDE. Among the baselines, PRESTO is the only approach that does not support sensor data replication at the storage tier. Hence, senor data is partitioned across the storage nodes and each storage node has a partial view on the global situation. At each storage node, sensor data is located via an index structure, and remote sensor data is accessed on-demand. The index at each storage node is updated when the topology of either sensor tier or storage tier changes. *PRIDE* is our approach, which replicates sensor data using models and updates its snapshots to maintain the freshness of sensor data. Finally, *PRIDE + PRESTO* combines PRESTO's model-based communication scheme between the tiers and PRIDE's model-based replication technique at the storage tier.

Like PRESTO, the communication load between the tiers is reduced by exploiting models on each sensor streams. Further, like PRIDE, each storage node maintains models for all sensors and updates from a model are broadcast to all peer storage nodes. Hence, PRIDE + PRESTO does not need to access remote sensor data for processing queries. However, it should be noted that PRIDE + PRESTO, like PRESTO, needs to synchronize models between the two tiers, and it makes the tiers less independent with each other.

6.3 Experiment 1: Performance of predictive replication-based 2-tier architecture

First, we show the performance of the predictive replication scheme in PRIDE. The data precision bound δ is 1 °C for all model-based approaches and Approx-Caching. All evaluation results are based on at least 5 runs and the averages with 95 % confidence intervals are taken.[5]

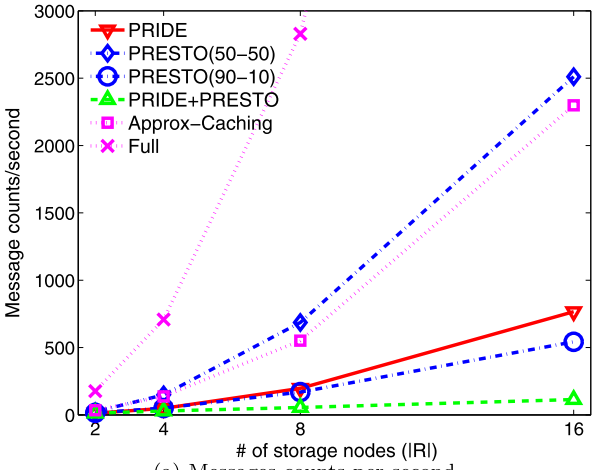
6.3.1 Scalability and timeliness

Figure 8 shows the scalability and timeliness of query processing in PRIDE and baselines when we change the number of storage nodes from 2 to 16. Each storage node receives data streams from 100 sensors. Hence, the total number of underlying sensors increases from 200 to 1,600 accordingly.

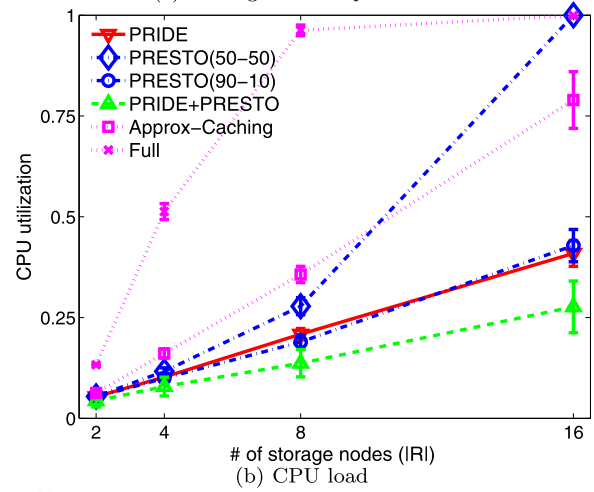
In Figs. 8(a), as the system size increases, the number of messages increases in all approaches. Among the replication-based approaches, Full and Approx-Caching generate a significant amount of messages as the network scales up. In contrast, the slope is much flatter in PRIDE compared to them since PRIDE filters out most of the incoming data from sensors as long as its models can predict the value within the precision bound. For instance, PRIDE filters out 93 % of the original data when 16 storage nodes are deployed while Approx-Caching filters out only 80 %. This gap increases as the system scales up. PRESTO, which does not support replication at the storage tier, is highly sensitive to data access patterns. For example, when 90–10 access pattern is applied, PRESTO incurs similar communication overhead as PRIDE. However, when 50–50 access pattern is applied, it incurs approximately 5.5 times more communication loads than when 90–10 access pattern is applied. This is because most communication overhead of PRESTO, unlike replication-based approaches, results from accessing remote data at the storage tier. This highly unpredictable nature of PRESTO can be a problem for real-time systems which need predictable behavior from the system. In contrast, the effect of varying access pattern is insignificant in replication-based approaches since all data accesses are handled locally. Hence, the graphs for replication-based approaches show only the results when 50–50 access pattern is applied unless they differ more than 5 %. For PRESTO, hereafter, we do not show the result of 50–50 access pattern unless it is necessary since 90–10 is the best case for PRESTO.

The least communication load is achieved when the advantages of both PRIDE and PRESTO are combined in PRIDE + PRESTO. For example, PRIDE + PRESTO

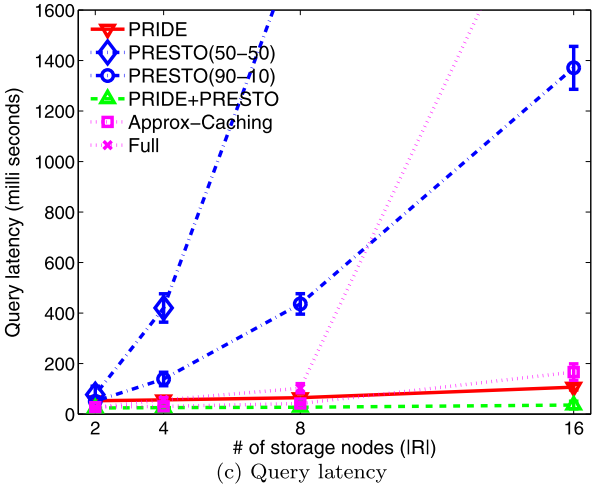
[5]The confidence interval bars are shown in the graphs.



(a) Messages counts per second



(b) CPU load



(c) Query latency

Fig. 8 Scalability and timeliness of PRIDE

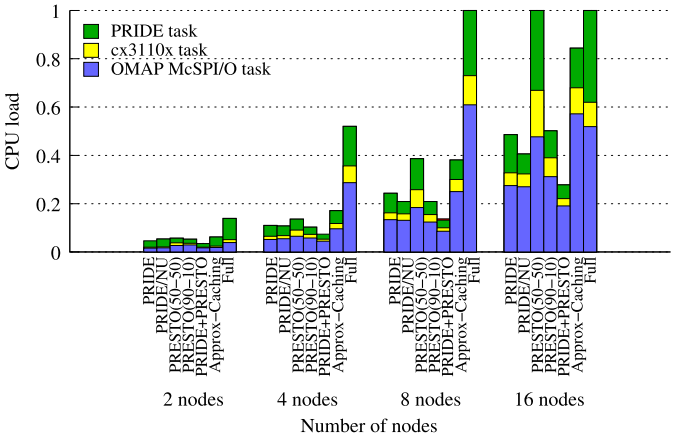


Fig. 9 The breakdown of CPU load

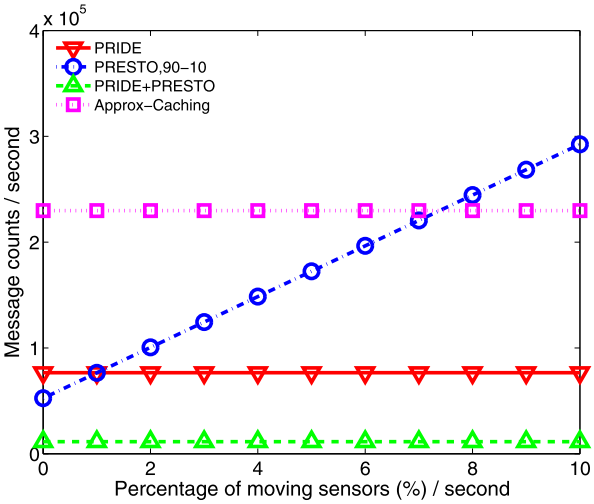
incurs only 8.8 times more messages when the number of storage nodes changes from 2 to 16. In contrast, PRIDE and PRESTO(90–10) generate 62.5 times and 34.8 times more messages, respectively, in the same environment. This is because PRIDE + PRESTO not only reduces the communication loads between the tiers using models, but it also eliminates the need for accessing remote data at the storage tier. Hence, PRIDE + PRESTO results in the least communication load as the system scales up.

Figure 8(b) shows the CPU utilization in the same experiment. The amount of communication is highly related to the CPU load since each message incurs overhead to handle it. In Fig. 8(b), the CPU loads increase proportionally to the amount of communication in all approaches. The breakdown of the CPU utilization is shown in Fig. 9. In the graph, three major tasks contribute to the overall measured CPU load; *cx3110x task* is the WI-FI driver for interrupt handling, *OMAP McSPI/O task* is the DMA transfer driver, and *PRIDE task* is the PRIDE task itself. The combined CPU load of cx3110x task and OMAP McSPI/O task represents the CPU overhead to process incoming/outgoing messages. As Fig. 9 shows, data communication is the primary source of CPU time for all approaches, and its portions increase as the size of the collaboration group increases.

In PRIDE and other model-based approaches, CPU load is not only related to the number of exchanged messages, but also to the number of underlying sensors since PRIDE updates data objects in the snapshot periodically using models of each sensor. In Fig. 9, the gap between PRIDE and PRIDE/NU quantifies the cost of the dynamic snapshot update using models. PRIDE/NU is the PRIDE approach performing the predictive replication, but the snapshot at each node is not dynamically updated by models. Figure 9 shows that PRIDE incurs less than 10 % additional CPU overhead than PRIDE/NU when 16 storage nodes are involved. This implies that the CPU overhead to maintain models in PRIDE is insignificant.

Figure 8(c) shows the query latency as the size of the system increases. In the replication-based approaches, CPU contention is the major factor determining the query latency. Once CPU load rises above a certain scheduling bound, the lengths of scheduling queues in the system increase limitlessly. For example, the query latency

Fig. 10 Effect of mobility

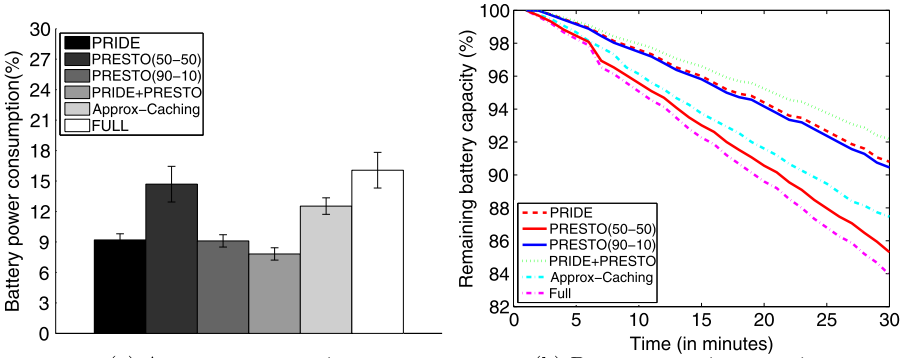


of Full approach increases without limit as CPU gets saturated. The query latencies of the other replication-based approaches increase gradually as their CPU load increases. In contrast, the query latencies of the non-replication-based approaches are mostly affected by the communication latency to access remote data. For example, even though PRESTO(90–10)'s CPU load is almost equal to PRIDE's, the query latency of PRESTO(90–10) is approximately 126 times longer than PRIDE's. This result demonstrates that the replication-based approach of PRIDE is more appropriate when timely query responses are required.

6.3.2 Mobility

In the same setting as in Sect. 6.3.1, the effect of dynamic network topology is tested when 0–10 % sensors change their primary storage node every second. The result with 16 storage nodes is shown in Fig. 10.

In PRESTO, as the more sensors change their primary storage nodes, the more communication loads are incurred to keep track of the locations of the sensor data; a new primary node of a sensor needs to broadcast the updated locations of the sensor to all peer nodes. For instance, 1 % additional topology change at the sensor tier incurs approximately 13 % increase of message counts at the storage tier to update index at each storage nodes. In practice, additional communication overheads are inevitable in non-replication-based approaches to transfer states and synchronize between the storage nodes (Abdelzaher et al. 2004); our experiment accounts for only the messages to update indexes that map the current sensor data to its primary storage node. In contrast, in PRIDE and other replication-based approaches, no additional communication is required at the storage tier when the topology changes in the sensor tier. This loose coupling between the tiers via replication at the storage tier makes PRIDE highly suitable for 2-tier sensor network applications having mobile entities.



(a) Average consumption (b) Battery capacity over time

Fig. 11 Energy consumption

6.3.3 Energy consumption

Since PRIDE targets (potentially mobile) low-end devices such as PDAs, and micro severs, energy is a critical resource for long lifetime. In this section, we compare the energy consumption of PRIDE to the baseline approaches. In the evaluation, 16 storage nodes are deployed for 30 minutes, where each storage node handles 100 sensor streams. The power is measured at the N810 device by monitoring the remaining battery power. The full capacity of the battery is 1,500 mA h.

Figure 11(a) shows the energy consumption after 30 minutes, and Fig. 11(b) shows the changes in the remaining battery power over 30 minutes. The result shows that PRIDE consumes significantly smaller energy than other approaches. For instance, PRIDE consumes 27 %, 38 %, and 43 % less energy than Approx-Caching, PRESTO(50–50), and Full approach, respectively. PRESTO's energy consumption can be reduced to the level of PRIDE's only when its data access pattern is highly skewed as much as 90–10. This result is expected since the overhead in computation and communication has net effect on the energy consumption.

6.3.4 Quality of data

In wireless sensor networks, packet losses are common due to various interferences from their environments. In this section, we evaluate the impact of lossy communication on the data accuracy at storage nodes. In the evaluation, the accuracy of sensor data at one of the storage nodes is measured while data packets from sensors are dropped with random probabilities. The accuracy of the sensor data at the storage node is quantified by *root mean square error* (RMSE) between the ground truth x and the estimated value at the storage node $\hat{x}$ over the entire emulation period n; RMSE is defined as $\sqrt{\frac{\sum_{k=1}^{n}(x_k-\hat{x}_k)^2}{n}}$.

Figure 12 shows the RMSEs of PRIDE and Approx-Caching while the packet drop ratio changes from 10 % to 90 %.[6] In Fig. 12, both approaches can satisfy the

[6]PRIDE is representative of approaches exploiting models including PRESTO and PRIDE + PRESTO. Hence, we only show the result of PRIDE.

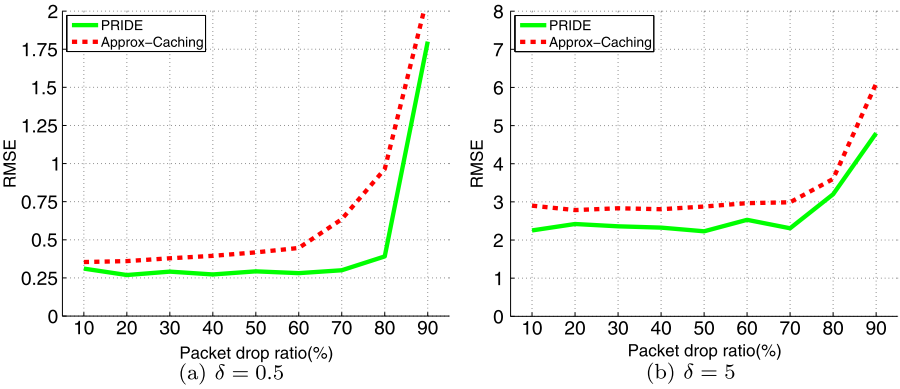


Fig. 12 RMSE vs. packet drop ratio

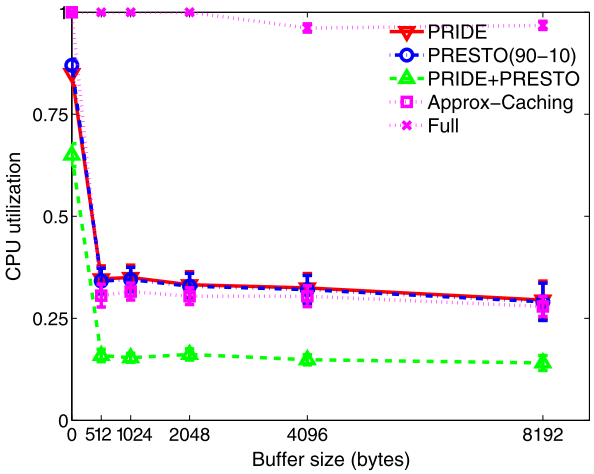
required precision bound δ even until the majority of sensor data is lost; however, PRIDE maintains the higher accuracy of sensor data (or, the lower average RMSE) despite the packet losses. For instance, in Fig. 12(a) PRIDE can maintain the required data quality, $\delta = 0.5$, until 82 % of sensor data is lost, while Approx-Caching begins to violate it when the packet drop ratio is over 63 %. When δ is greater, e.g. $\delta = 5$ in Fig. 12(b), even more packet losses can be tolerated in both approaches; the target data precision is not violated even until 93 % and 86 % of packets are lost in PRIDE and Approx-Caching, respectively.

This result shows that both approaches are highly resilient to packet losses; a fraction of data can be enough to estimate the current state of physical processes. This is because both PRIDE and Approx-Caching exploit the impreciseness of data on purpose. However, PRIDE can achieve the higher data quality of sensor data under unreliable communication. For PRIDE, it is also interesting that RMSE does not increase monotonically as the packet drop ratio increases. This is because some sensor data reflecting transient changes of physical processes can have a negative effect in modeling the long-term trends of the physical processes.

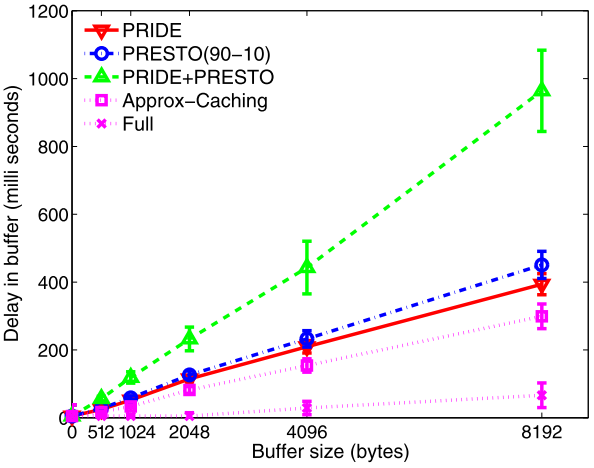
6.4 Experiment 2: Trading data freshness for scalability

PRIDE has been designed for distributed real-time applications, where maintaining high data freshness is critical. To this end, storage nodes in PRIDE deliver outgoing messages to peer nodes without delay in the buffer. However, some applications are tolerant, to some extent, to communication delays and subsequent staleness of data. For example, environment monitoring applications (Selavo et al. 2007; Fall 2003) require high scalability to cover wide areas with high precision; however, they rarely need high sensor data freshness. For such applications, data freshness can be compromised to some extent to achieve other quality aspects such as high scalability.

In this experiment, we evaluate how the scalability of PRIDE is affected when the freshness of sensor data is compromised. In the experiment, individual outgoing messages are buffered and are transmitted in one message. Instead of delivering messages immediately to peer nodes, the messages in each buffer is concatenated into



(a) Utilization



(b) Delay of buffering

Fig. 13 Trading freshness for scalability

one message and are transmitted only when the buffer is full. The size of buffer is changed from 512 bytes to 8192 bytes when 32 storage nodes are deployed. At the receiving nodes, the incoming concatenated messages are parsed and divided into original messages before their processing.

As shown in Fig. 13(a), the utilization at each storage node decreases significantly for all approaches, making them much more scalable. The gain is particularly pronounced in Approx-Caching since it has the greatest communication overhead compared to other approaches as shown in Fig. 9. For example, when the buffer size is 512 bytes, the utilization of Approx-Caching decreases from 1.0 to 0.31 while the utilization of PRIDE drops from 0.85 to 0.34. Due to this significant decrease of communication overheads, Approx-Caching outperforms PRIDE and other model-based approaches slightly except PRIDE + PRESTO. When communication overhead is low enough, PRIDE and model-based approaches have a higher computational over-

head to process models. Since the communication overhead is low enough when the size of buffer is 512 bytes, not much further gain is achieved by increasing the buffer size over 512 bytes. For example, only 15 % CPU utilization is reduced for PRIDE by increasing the buffer size from 512 bytes to 8192 bytes. However, as shown in Fig. 13(b), delay in the buffer increases linearly as the size of the buffer increases. For instance, the delay in buffer increases from 34 milliseconds to 394 milliseconds for PRIDE when the buffer size is increased from 512 bytes to 8192 bytes. Hence, the size of buffer should be chosen carefully to make best compromise between the communication overhead and latency.

6.5 Experiment 3: Adaptability

We next evaluate the adaptability of PRIDE against unpredictable workloads. First, the effects of varying precision bounds are quantified in Sect. 6.5.1 by manually changing precision bounds. In the following Sections, the adaptability of PRIDE is evaluated by deploying 32 storage nodes and turning on the data quality controller at each node. The performance goal is given by the specification $\langle 0.6, 10\ ^\circ\mathrm{C}\rangle$, which means the CPU utilization bound is 60 % and the maximum precision bound δ is 10 °C.[7] In the experiment, the sampling intervals of controllers are set to 20 seconds.

6.5.1 Effects of varying precision bounds

In PRIDE, the target CPU utilization is achieved by systematically controlling the precision bounds of models. In this experiment, we evaluate the impact of varying precision bounds on the system by manually applying different precision bounds, $\delta = 0.5$, 1, 5, and 10 to PRIDE. Controllers are turned off to prevent the dynamic changes of δ at runtime.

Since PRIDE has *n-to-n* connections, the number of messages is proportional to $\frac{n^2}{\delta}$, where n is the number of storage nodes. The implication is that precision bounds have higher impact on the system performance as the size of replication groups increases. Figure 14 shows the numbers of messages, CPU utilizations, and the query latencies when we change the number of storage nodes from 2 to 16. As expected, changing precision bounds has higher impact on the system when the size of replication group is great. For example, when the size of replication group is 16, changing δ from 10 to 1 incurs 0.194 increase in the CPU load, while 0.04 increase in the CPU load when the size of replication group is 8. This result demonstrates that the dynamics of PRIDE at different sizes of replication group can not be captured with one model. This end, PRIDE's data quality controller exploits the gain scheduling technique to choose different models and corresponding controller parameters as the size of replication group changes.

Another interesting point to note is that theoretically as δ approaches to zero, the number of messages approaches to the infinity and the quality of data increases linearly. In practice, however, the maximum number of messages and subsequent model

[7] In the simulated scenario, the observed temperatures are exploited to make the real-time prediction on fire and smoke flows, and hence high precision bound, such as below 10 °C, can be tolerated. To achieve higher data accuracy in different scenarios, the timeliness can be traded as shown in Sect. 6.4.

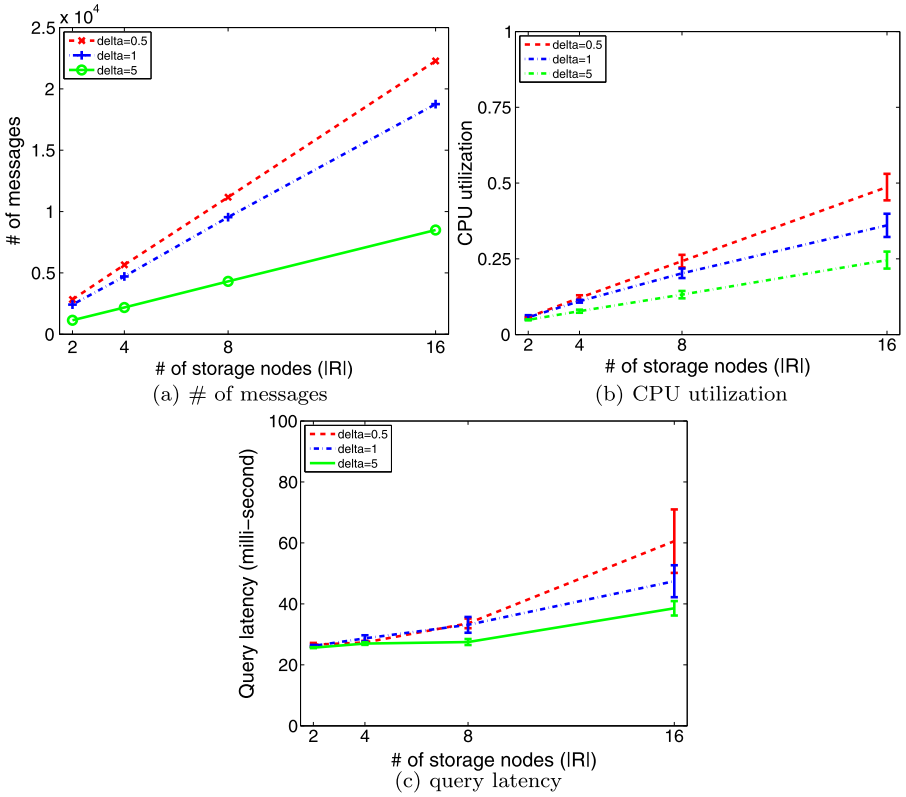


Fig. 14 The effect of different precision bounds

quality is bounded by the physical precision and sampling rates of sensors. For example, the number of messages is expected to double when δ is changed from 1 to 0.5 in theory; however, in practice, it incurs less than 20 % increase in the number of messages regardless of the size of the replication group.

6.5.2 Average performance

We evaluate the adaptability of PRIDE by changing the workload while the data quality controllers are turned on and off. The workloads are varied by changing the number of sensor streams for each storage node from 60 to 140; hence, $|S|$ changes from 1920 to 4480. For PRIDE without controllers, the precision bound is set to 3 °C.

In Fig. 15, the average performance is shown. Figure 15(a) shows that PRIDE with controller achieves the target CPU load closely in all workloads. In contrast, the CPU load fluctuates dramatically between under-utilization and over-utilization when no control is applied; the load changes between 0.28 and 1. Violating the goal in CPU load implies that the latency of application tasks and queries in PRIDE can be increased significantly. Figure 15(b) shows the changes in δ to achieve the target CPU load. In PRIDE, the precision bound δ increases linearly as the workload increases. However, PRIDE still satisfies the maximum precision bound, which is 10 °C.

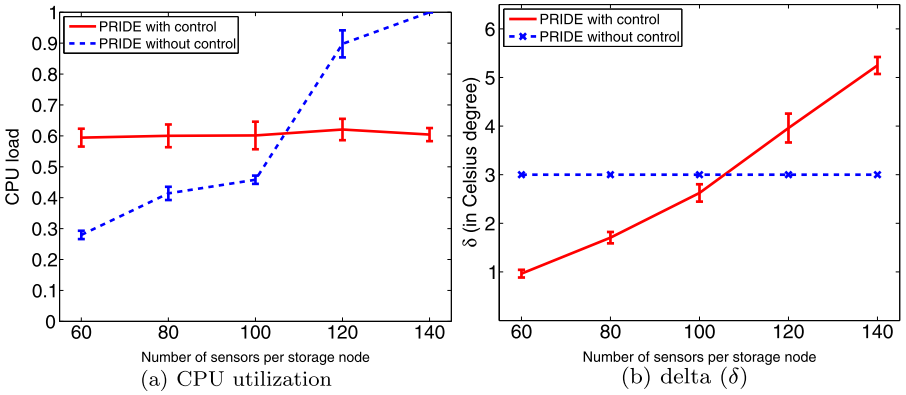
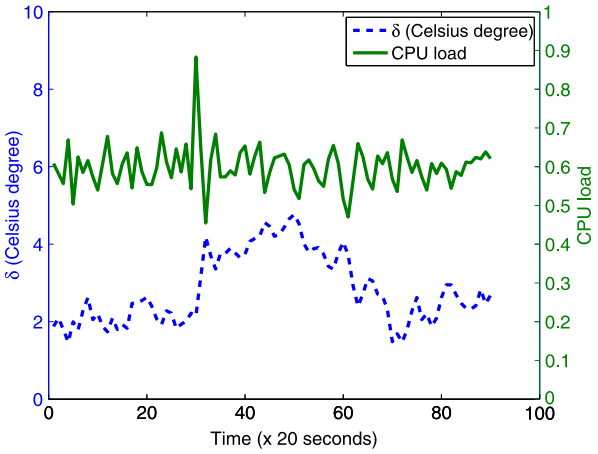


Fig. 15 Varying workload

Fig. 16 Transient behavior



6.5.3 Transient performance

The average performance is not enough to show the performance of dynamic systems. Transient performance such as settling time should be small enough to satisfy the requirements of applications. In this experiment, the workload is increased by changing the number of underlying sensors and the transient behavior is observed. For example, we can consider a situation in which a firefighter moves to a location where sensors are densely deployed. The number of sensors at one of the storage nodes (the N810 device) is increased from 80 to 140 for 30 sampling periods.

Figure 16 shows the CPU load and the precision bound δ. The increase of the workload starts at the 30th sampling period. We can see that the CPU load increases suddenly at the 30th sampling period to handle the increased workload; for example, the CPU load is surged to 0.9. However, the CPU load stabilizes to the target CPU load, which is 0.6, within 2 sampling periods; it is achieved by increasing δ from 2 to 4. When the workload decreases to the original level at the 60th sampling period,

it takes 1 sampling period for CPU load to stabilize; it is achieved by decreasing δ back to the original level.

Finally, it should be noted that even though it is not shown for the space limitation, our experiment shows that PRIDE can also closely track the target CPU utilization bound under unpredictable changes in sensor data rates.

7 Related work

7.1 Storing and querying sensor data

There have been significant research efforts devoted to the storage and querying of sensor data. Some approaches, such as Cougar (Bonnet et al. 2001) and TinyDB (Madden et al. 2005), view each sensor node as a mini data repository, and the sensor network as a database distributed across the sensor nodes. A user formulates a declarative query, which is similar to SQL, and sends it to the sensor network through a gateway. Once a query is disseminated, sensor nodes probe their sensors and propagate sensor measurements back to the gateway. These approaches are essentially centralized ones since sensor readings are funneled into a centralized gateway. This funneling effect can severely limit the scalability of these approaches (Ahn et al. 2006). Exploiting several gateways by partitioning the sensor network can mitigate the funneling effect (Abbasi and Younis 2007; Li et al. 2006). However, these gateways have only partial view on the situation since the sensor network is partitioned. In contrast, PRIDE provides global view on the physical situation in a large scale sensor network using the predictive replication mechanism at the upper storage tier.

Data-centric routing and storage approaches (Intanagonwiwat et al. 2003; Ratnasamy et al. 2002), which store sensed data within sensor networks themselves, achieve high scalability by minimizing communication overhead. Instead of propagating sensor observations to a gateway, sensed data are stored within sensor networks and queries are processed in the network; communications are postponed until the query processing time. However, the high scalability can be achieved at the cost of increased system complexity and reduced manageability (Gnawali et al. 2006). Further, they cannot provide the timeliness in data accesses and query processing due to communication delays at the query processing time. Unlike data-centric approaches, PRIDE is appropriate for real-time applications since it does not incur communication overhead at the query processing time.

PRIDE is influenced by the recent tiered sensor network architectures (Gnawali et al. 2006; Li et al. 2006; Desnoyers et al. 2005), which lie between these two extremes, centralized ones and data-centric approaches. PRESTO (Li et al. 2006) reduces the communication load between the tiers by exploiting the model-based push approach. However, since each proxy has a limited view, excessive communication overhead and delay is inevitable for processing queries on global sensor data. Further, the model parameters should be synchronized between the tiers, making the tiers dependent to each other. In PRESTO, simple multicasting of sensor data to all proxies is not enough to provide global view since each proxy cannot make predictions on

current and future sensor values that are managed by peer proxies. Models need to be replicated to all peer proxies at the proxy tier as in PRIDE to achieve timely query processing on global sensor data. Our experiment in Sect. 6 shows that combining the approaches of PRESTO and PRIDE can achieve the best performance in terms of scalability and the timeliness of query processing. TSAR (Desnoyers et al. 2005) tries to achieve both the timeliness of data accesses and high scalability through distributed indexing at the upper device tier. However, since TSAR is pull-based, it incurs communication delays at processing queries to look up the distributed index and fetch data from the sensor tier. Further, the distributed indexing scheme tightly couples the sensor tier and the upper device tier, limiting the flexibility of the system. The tight coupling between tiers in the previous approaches, including TSAR, makes it very complex to support the mobility of sensors and upper tier devices. In contrast, the predictive replication mechanism of PRIDE does not require tight coupling between tiers while supporting timely access to global sensor data.

7.2 Data reduction using models and filters

The problem of data reduction using predictive modeling techniques in sensor networks has received a great deal of attention from the research community (Goel and Imielinski 2001; Deshpande et al. 2004; Lazaridis and Mehrotra 2003; Pattem et al. 2008; Wei et al. 2011; Jiang et al. 2011). Goel and Imielinski (2001) suggested to visualize a snapshot of the sensor readings in the network as an optical image. The basic paradigm requires a base station to monitor the readings from the sensors and generate a prediction model, which is valid over a given time interval. In BBQ (Deshpande et al. 2004), a spatio-temporal prediction model is generated from historical sensor data at a base station; new data is fetched from sensors only if the confidence interval of query results from the prediction models are out of the desired range. Even though above approaches are different in technical details, they commonly try to optimize the model at a base station to reduce the communication overhead between sensors and a base station. In contrast, PRIDE aims to reduce replication overheads among upper-tier devices. Hence, models matching sensor streams are replicated to all upper-tier devices. Given replicated and distributed models, minimizing the cost of model maintenance is more important than optimizing individual models. Our approach of using models at the upper tier is not exclusive but complementary to the previous work.

A few model-based data reduction approaches try to adapt the models for the observed phenomenon (Le Borgne et al. 2007; Santini and Romer 2006). Santini and Romer (2006) proposed a modeling scheme that does not require a-priori knowledge of the observed phenomenon. The adaptive algorithms both at the data source and the sink of a data stream exploit the Least-Mean-Square algorithm to adapt their model parameters independently without help from a central server. In Le Borgne et al. (2007), proposed the racing mechanism, which allows to discard poorly performing models from a set of candidate prediction models via a cost/benefit analysis. Unlike these approaches, which advocate particular models for particular types of data or domain, the goal of our work is to build a data abstraction layer that integrates models accounting for the timeliness and robustness. Hence, previous modeling techniques

can be applied to PRIDE as long as they are light-weight, general enough to be applied to a broad range of data sources, and do not require a massive amount of historic data.

Jain et al. (2004) proposed a data reduction technique using classical linear filter theory. They introduced the Dual Kalman Filter (DKF) architecture as a general linear solution to this problem. They evaluated the filtering performance of a Kalman filter in the simulation testbed handling a single stream. PRIDE explores some of this idea in the context of more sophisticated 2-tier sensor network architectures handling massive amount of sensor streams.

In the database community, Deshpande and Madden's work (Deshpande and Madden 2006; Thiagarajan and Madden 2008) is quite related to our work. MauveDB supports a declarative language for defining model-based views, allows querying over such views using SQL, and supports several different materialization strategies. While both MauveDB and PRIDE integrate models into databases, MauveDB provides a more general framework allowing different models and materialization strategies. Unlike MauveDB, PRIDE focus on the timeliness and robustness in processing queries. Hence, it extends the model-based approach to the problem of guaranteeing the QoS using dynamic adaption of accuracy bounds of models.

Another relevant area is moving object databases (Wolfson et al. 1998; Jensen et al. 2004; Pelanis et al. 2006; Jeung et al. 2010). In these systems, an object's location is approximated using models without continuous updates. In Wolfson et al. (1998), define a specific attribute for moving objects and predicts objects' future locations based on its velocity vector. However, as far as we know this area has focused on different issues than the ones in this paper, e.g., efficient indexing mechanisms for moving objects (Pelanis et al. 2006; Jensen et al. 2004) and exploiting geographic information for better estimation of future locations (Jeung et al. 2010).

7.3 Data replication in distributed databases

Traditional replicated relational database systems focus on the problem of guaranteeing strong consistency of replicated data. Although strong consistency provides a convenient programming model, these systems are limited in scalability and availability (Gray et al. 1996). Recently, distributed real-time databases (DRTDBs) focus on data freshness and timeliness of transactions instead of providing strong logical consistency (Peddi and DiPippo 2002; Wei et al. 2003). They primarily target business application domains such as stock trades (Kang et al. 2007). PRIDE, however, targets real-time sensor network applications, in which the semantics of data from physical phenomena can be exploited.

7.4 Control-theoretic approach for QoS guarantees

Control theory, one of the most widely used mathematical frameworks to control behavior of dynamic systems (Hellerstein et al. 2004), is at the core of our sensor data quality control. Control theory has been applied to QoS management and real-time scheduling (Lu et al. 2002, 2005) due to its robustness against unpredictable operating environments. However, these work do not consider sensor data quality management

issues; therefore, they are not directly comparable to our work. Feedback control has also been recognized as a viable approach to manage the real-time database performance. They guarantee the desired transaction deadline miss ratio (Kang et al. 2004; Amirijoo et al. 2006), CPU utilization (Oh and Kang 2007), and query processing latency (Zhou and Kang 2009). They control either the freshness or precision of sensor data by selectively allowing incoming sensor data updates. Most of the existing work show the robustness of their feedback control mechanism by simulations. In contrast, our work guarantees the prediction accuracy of models, not data itself. Further, our work is also different from the existing work in that our work is implemented in a real mobile device and evaluated on a realistic testbed.

8 Conclusions and future work

This paper introduced the design, implementation, and experimental evaluation of PRIDE, a new data abstraction layer for collaborative 2-tier sensor network applications. By integrating an efficient predictive replication mechanism and adaptive data quality control, PRIDE at the upper tier devices enables transparent access to global sensor data in a timely, flexible, and robust manner in highly dynamic environments of emerging distributed real-time applications interacting with pervasive sensors. We showed that the proposed predictive replication mechanism and adaptive data control mechanism can be integrated seamlessly into a general data management system by applying it into a popular embedded DBMS. PRIDE was evaluated in a large-scale 2-tier sensor network testbed, emulating search-and-rescue tasks for a building fire with realistic workloads. Our experimental evaluation of PRIDE demonstrates the benefit and feasibility of our approach in large-scale 2-tier sensor networks.

In the future, we plan to enhance PRIDE in several different directions. First, PRIDE will be extended to include different modeling schemes. Currently, PRIDE supports only a simple modeling scheme using state vectors, without exploiting the correlation among distributed sensors. By utilizing the correlation among sensors, we may further increase the accuracy of models in PRIDE. However, the cost of using sophisticated models should be evaluated. Second, we are interested in building a testbed that is more realistic. In the original design of PRIDE, storage nodes are supposed to construct an ad-hoc mesh network. However, our current testbed uses a fixed network of both wired and wireless connections, ignoring the effect of ad-hoc routing of a mesh network. We believe that different nodes in ad-hoc mesh networks have very different resource usage (computing, communication, and power consumption). This asymmetric resource usage pattern should be considered in real situations.

References

Communication and networking technologies for public safety (2008) National Institute of Standards and Technology. http://w3.antd.nist.gov/comm_net_ps.shtml
Fire growth and smoke transport modeling with CFAST (2008). http://fast.nist.gov/
Fire information and rescue equipment (FIRE) project (2008). http://fire.me.berkeley.edu/
Nokia N-series (2008). http://www.nseries.com/
Oracle Berkeley DB (2008). http://www.oracle.com

IEEE portable applications (2009). http://standards.ieee.org/regauth/posix
Abbasi AA, Younis M (2007) A survey on clustering algorithms for wireless sensor networks. Comput Commun 30:2826–2841
Abdelzaher T, Blum B, Cao Q, Chen Y, Evans D, George J, George S, Gu L, He T, Krishnamurthy S, Luo L, Son S, Stankovic J, Stoleru R, Wood A (2004) Envirotrack: towards an environmental computing paradigm for distributed sensor networks. In: Proceedings of the 24th international conference on distributed computing systems (ICDCS'04)
Ahn GS, Hong SG, Miluzzo E, Campbell AT, Cuomo F (2006) Funneling-mac: a localized, sink-oriented mac for boosting fidelity in sensor networks. In: Proceedings of the 4th international conference on embedded networked sensor systems (SenSys '06)
Akyildiz I, Wang X (2005) A survey on wireless mesh networks. IEEE Commun Mag 43(9):S23–S30
Amirijoo M, Hansson J, Son SH (2006) Specification and management of QoS in real-time databases supporting imprecise computations. IEEE Trans Comput 55(3):304–319
Bonnet P, Gehrke J, Seshadri P (2001) Towards sensor database systems. In: Proceedings of the second international conference on mobile data management (MDM '01)
Cook SA, Pachl JK, Pressman IS (2002) The optimal location of replicas in a network using a read-one-write-all policy. Distrib Comput 15(1):57–66
Deshpande A, Guestrin C, Madden SR, Hellerstein JM, Hong W (2004) Model-driven data acquisition in sensor networks. In: Proceedings of the 30th VLDB conference, Toronto, Canada
Deshpande A, Madden S (2006) Mauvedb: supporting model-based user views in database systems. In: Proceedings of the 2006 ACM SIGMOD international conference on management of data (SIGMOD '06). ACM, New York, pp 73–84
Desnoyers P, Ganesan D, Shenoy P (2005) TSAR: a two tier sensor storage architecture using interval skip graphs. In: Proceedings of the 3rd international conference on embedded networked sensor systems (SenSys '05)
Fall K (2003) A delay-tolerant network architecture for challenged Internets. In: Proceedings of the 2003 conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM '03), pp 27–34
Fife LD, Gruenwald L (2003) Research issues for data communication in mobile ad-hoc network database systems. SIGMOD Rec 32:42–47
Gelb A (ed) (1974) Applied optimal estimation. MIT Press, Cambridge
Gnawali O, Jang KY, Paek J, Vieira M, Govindan R, Greenstein B, Joki A, Estrin D, Kohler E (2006) The tenet architecture for tiered sensor networks. In: Proceedings of the 4th international conference on embedded networked sensor systems (SenSys '06)
Goel S, Imielinski T (2001) Prediction-based monitoring in sensor networks: taking lessons from mpeg. Comput Commun Rev 31:82–98
Gray J, Helland P, O'Neil P, Shasha D (1996) The dangers of replication and a solution. In: SIGMOD '96
Hellerstein JL, Diao Y, Parekh S, Tilbury DM (2004) Feedback control of computing systems. Wiley/IEEE Press, New York
Intanagonwiwat C, Govindan R, Estrin D, Heidemann J, Silva F (2003) Directed diffusion for wireless sensor networking. IEEE/ACM Trans Netw 11(1):2–16
Jain A, Chang EY, Wang YF (2004) Adaptive stream resource management using Kalman filters. In: Proceedings of the 2004 ACM SIGMOD international conference on management of data (SIGMOD '04). ACM Press, New York, pp 11–22
Jensen CS, Lin D, Ooi BC (2004) Query and update efficient b+-tree based indexing of moving objects. In: Proceedings of the thirtieth international conference on very large data bases (VLDB '04), vol 30, pp 768–779
Jeung H, Yiu ML, Zhou X, Jensen CS (2010) Path prediction and predictive range querying in road network databases. VLDB J 19:585–602
Jiang H, Jin S, Wang C (2011) Prediction or not? An energy-efficient framework for clustering-based data collection in wireless sensor networks. IEEE Trans Parallel Distrib Syst 22(6):1064–1071
Jiang X, Chen NY, Hong JI, Wang K, Takayama L, Landay JA (2004) Siren: context-aware computing for firefighting. In: Proceedings of second international conference on pervasive computing
Kang KD, Oh J, Son SH (2007) Chronos: feedback control of a real database system performance. In: RTSS
Kang KD, Son SH, Stankovic JA (2004) Managing deadline miss ratio and sensor data freshness in real-time databases. IEEE Trans Knowl Data Eng 16(10):1200–1216
Lazaridis I, Mehrotra S (2003) Capturing sensor-generated time series with quality guarantees. In: Proceedings of 19th international conference on data engineering, 2003, pp 429–440

Le Borgne YA, Santini S, Bontempi G (2007) Adaptive model selection for time series prediction in wireless sensor networks. Signal Process 87:3010–3020
Lee EA (2008) Cyber physical systems: design challenges. Tech rep UCB/EECS-2008-8, EECS Department, University of California, Berkeley
Li M, Ganesan D, Shenoy P (2006) Presto: feedback-driven data management in sensor networks. In: Proceedings of the 3rd conference on networked systems design & implementation (NSDI'06)
Lu C, Stankovic JA, Son SH, Tao G (2002) Feedback control real-time scheduling: framework, modeling, and algorithms. Real-Time Syst 23(1–2):85–126
Lu C, Wang X, Koutsoukos X (2005) Feedback utilization control in distributed real-time systems with end-to-end tasks. IEEE Trans Parallel Distrib Syst 16(6):550–561
Madden SR, Franklin MJ, Hellerstein JM, Hong W (2005) Tinydb: an acquisitional query processing system for sensor networks. ACM Trans Database Syst 30(1):122–173
Mathiason G, Andler SF, Son SH (2008) Virtual full replication for scalable and adaptive real-time communication in wireless-sensor networks. In: Proceedings of the second intl conference on sensor technologies and applications (SENSORCOMM 2008)
Mohapatra P, Gui C, Li J (2004) Group communications in mobile ad hoc networks. Computer 37(2):52–59
de Morais Cordeiro C, Gossain H, Agrawal D (2003) Multicast over wireless mobile ad hoc networks: present and future directions. IEEE Netw 17(1):52–59
Oh J, Kang KD (2007) An approach for real-time database modeling and performance management. In: Real time and embedded technology and applications symposium, 2007 (RTAS '07). 13th. IEEE Press, New York, pp 326–336
Olston C, Loo BT, Widom J (2001) Adaptive precision setting for cached approximate values. SIGMOD Rec 30(2):355–366
Padmanabhan P, Gruenwald L, Vallur A, Atiquzzaman M (2008) A survey of data replication techniques for mobile ad hoc network databases. VLDB J 17:1143–1164
Pattem S, Krishnamachari B, Govindan R (2008) The impact of spatial correlation on routing with compression in wireless sensor networks. ACM Trans Sens Netw 4:24:1–24:33
Peddi P, DiPippo LC (2002) A replication strategy for distributed real-time object-oriented databases. In: Symposium on object-oriented real-time distributed computing, pp 129–136
Pelanis M, Šaltenis S, Jensen CS (2006) Indexing the past, present, and anticipated future positions of moving objects. ACM Trans Database Syst 31:255–298
Ramamritham K, Son SH, Dipippo LC (2004) Real-time databases and data services. Real-Time Syst 28(2–3):179–215
Raniwala A, cker Chiueh T (2005) Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. In: Proceedings of 24th annual joint conference of the IEEE computer and communications societies (INFOCOM 2005), vol 3. IEEE Press, New York, pp 2223–2234
Ratnasamy S, Karp B, Yin L, Yu F, Estrin D, Govindan R, Shenker S (2002) Ght: a geographic hash table for data-centric storage. In: Proceedings of the 1st ACM international workshop on wireless sensor networks and applications (WSNA '02)
Santini S, Romer K (2006) An adaptive strategy for quality-based data reduction in wireless sensor networks. In: Proc INSS
Selavo L, Wood A, Cao Q, Sookoor T, Liu H, Srinivasan A, Wu Y, Kang W, Stankovic J, Young D, Porter J (2007) Luster: wireless sensor network for environmental research. In: Proceedings of the 5th international conference on embedded networked sensor systems (SenSys '07)
Sha K, Shi W, Watkins O (2006) Using wireless sensor networks for fire rescue applications: requirements and challenges. In: IEEE international conference on electro/information technology
Son SH (1988) Replicated data management in distributed database systems. SIGMOD Rec 17(4):62–69
Stankovic JA, Lee I, Mok A, Rajkumar R (2005) Opportunities and obligations for physical computing systems. Computer 38(11):23–31
Tatbul N, Çetintemel U, Zdonik S, Cherniack M, Stonebraker M (2003) Load shedding in a data stream manager. In: Proceedings of the 29th international conference on very large data bases (VLDB '2003)
Thiagarajan A, Madden S (2008) Querying continuous functions in a database system. In: Proceedings of the 2008 ACM SIGMOD international conference on management of data (SIGMOD '08). ACM, New York, pp 791–804
Wang X, Jia D, Lu C, Koutsoukos X (2007) DEUCON: Decentralized End-to-End Utilization Control for Distributed Real-Time Systems. IEEE Trans Parallel Distrib Syst 18(7):996–1009
Wei G, Ling Y, Guo B, Xiao B, Vasilakos AV (2011) Prediction-based data aggregation in wireless sensor networks: combining grey model and Kalman filter. Comput Commun 34(6):793–802

Wei Y, Son SH, Stankovic JA, Kang KD (2003) Qos management in replicated real time databases. In: Proceedings of the 24th IEEE international real-time systems symposium (RTSS '03)

Wolfson O, Chamberlain S, Dao S, Jiang L, Mendez G (1998) Cost and imprecision in modeling the position of moving objects. In: Proceedings of the fourteenth international conference on data engineering (ICDE '98). IEEE Comput Soc, Los Alamitos, Washington, DC, USA, pp 588–596

Zhou Y, Kang KD (2009) Integrating proactive and reactive approaches for robust real-time data services. In: Proceedings of the 30th IEEE international real-time systems symposium (RTSS '09)

Woochul Kang received the Ph.D. degree in computer science from the University of Virginia, Charlottesville, in 2009. He is a Research Scientist at the Electronics and Telecommunications Research Institute (ETRI), Korea. Currently, he is investigating a distributed middleware architecture that enables efficient and timely access to realtime sensor data in large-scale distributed cyber-physical systems (CPS). His research interests include cyber-physical systems, real-time embedded systems, large-scale distributed systems, sensor networks, and feedback control of computing systems.

Sang H. Son received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, the M.S. degree from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, and the Ph.D. degree in computer science from the University of Maryland, College Park, in 1986. He is a Professor with the Department of Computer Science, University of Virginia, Charlottesville. His research interests include real-time and embedded systems, database and data services, QoS management, wireless sensor networks, and information security. Prof. Son is on the Executive Board of the IEEE Technical Committee on Real-Time Systems, for which he served as the Chair during 2007-2008. He is currently serving as an Associate Editor for the IEEE TRANSACTIONS ON COMPUTERS and the Real-Time Systems Journal.

John A. Stankovic is the BP America Professor in the Computer Science Department at the University of Virginia. He served as Chair of the department for 8 years. He is a Fellow of both the IEEE and the ACM. He won the IEEE Real-Time Systems Technical Committee's Award for Outstanding Technical Contributions and Leadership. He also won the IEEE Technical Committee on Distributed Processing's Distinguished Achievement Award (inaugural winner). He has five Best Paper awards, including one for ACM SenSys 2006. Stankovic has an h-index of 87 and an i10 index of 225. He has also won Distinguished Faculty Awards at the University of Massachusetts and the University of Virginia. He has given more than 30 Keynote talks at conferences and many Distinguished Lectures at major Universities. He was the Editor-in-Chief for the IEEE Transactions on Distributed and Parallel Systems and was founder and co-editor-in-chief for the Real-Time Systems Journal. His research interests are in real-time systems, distributed computing, wireless sensor networks, and cyber physical systems. Prof. Stankovic received his PhD from Brown University.