J. Parallel Distrib. Comput. ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Consistency-driven data quality management of networked sensor systems

Kewei Sha, Weisong Shi *

Department of Computer Science, Wayne State University, 420 State Hall, 5143 Cass Ave, Detroit, MI 48202, USA

ARTICLE INFO

Article history:
Received 19 April 2007
Received in revised form
26 August 2007
Accepted 3 June 2008
Available online xxxx

Keywords:
Data quality
Consistency models
Wireless sensor networks
Energy efficiency
Adaptation

ABSTRACT

With more and more real deployments of wireless sensor network applications, we envision that their success is nonetheless determined by whether the sensor networks can provide *a high quality stream of data* over a long period. In this paper, we propose a consistency-driven data quality management framework called *Orchis* that integrates the quality of data into an energy efficient sensor system design. Orchis consists of four components, *data consistency models*, *adaptive data sampling and process protocols*, *consistency-driven cross-layer protocols* and *flexible APIs to manage the data quality*, to support the goals of high data quality and energy efficiency. We first formally define a consistency model, which not only includes *temporal consistency* and *numerical consistency*, but also considers *the application-specific requirements of data* and *data dynamics in the sensing field*. Next, we propose an adaptive lazy energy efficient data collection protocol, which adapts the data sampling rate to the data dynamics in the sensing field and keeps lazy when the data consistency is maintained. Finally, we conduct a comprehensive evaluation to the proposed protocol based on both a TOSSIM-based simulation and a real prototype implementation using MICA2 motes. The results from both simulation and prototype show that our protocol reduces the number of delivered messages, improves the quality of collected data, and in turn extends the lifetime of the whole network. Our analysis also implies that a tradeoff should be carefully set between data consistency requirements and energy saving based on the specific requirements of different applications.

1. Introduction

As new fabrication and integration technologies reduce the cost and size of wireless micro-sensors, we are witnessing another revolution that facilitates the observation and control of our physical world [1,6,5,22], just as networking technologies have changed the way individuals and organizations exchange information. Micro-sensors such as Motes from Intel and Crossbow [4] have been developed to make WSN applications possible; TinyOS [10,12] has been designed to provide adequate system support to facilitate sensor node programming; Several applications, such as habitat monitoring [35], ZebraNet [25], Counter-sniper system [36], environment sampling [2], target tracking [34], and structure monitoring [42], have been launched, showing the promising future of wide range of applications of wireless sensor networks (WSNs).

With the main function of collecting interesting and meaningful data, the success of WSN applications is nonetheless determined by whether they can provide a *high quality stream of data over a long period*. The inherent feature of unattended and untethered deployment of WSN in a malicious environment, however, imposes challenges to the underlying systems. These challenges are further complicated by the fact that WSNs are usually seriously energy and storage constrained. However, most previous efforts focus on devising techniques to save the sensor node energy and thus extend the lifetime of the whole WSN. We envision that data quality management has been becoming a more and more important issue in the design of WSNs.

In principle, the quality of data should reflect the timeliness and accuracy of collected data that are presented to interested recipients who make the final decision based on these data. Complementing to the work on the sensor design that improves the accuracy of sensing, in this paper, we intend to study the relationship between data quality and energy-efficient design of WSNs. To integrate and manage data quality in WSNs, we propose a framework named *Orchis*, which includes a set of data consistency models customized to WSNs, a set of APIs to management the quality of collected data, an adaptive protocol for data sampling, a set of consistency-driven cross layer protocols to support achieving the goals of data consistency and energy efficiency. The novelty of this work is that we propose to use consistency models, including temporal, numerical, and frequency three perspectives, as metrics to measure the quality of the collected data in wireless sensor networks, and based on these models, we propose a framework to manage data quality of WSNs. To the best of our knowledge, we are the first to propose consistency models in wireless sensor networks and to try to manage the data quality

* Corresponding author.
E-mail addresses: kewei@wayne.edu (K. Sha), weisong@wayne.edu (W. Shi).

0743-7315/$ – see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2008.06.004

from the viewpoint of data consistency. Among these components, we address two of them, consistency models and the Alep protocol in detail in this paper. First, we formally define a new metric, data consistency model, to evaluate the data quality. Intuitively, most people think that the higher requirements of data quality, the more energy will be consumed. However, we find that this intuition is not necessarily held, and that the energy can be saved if we consider data consistency and data dynamics together. This fact in turn inspires us to attack the problem from the perspective of data consistency and data dynamics, and exploit the data consistency in system protocol design. Thus, an adaptive, lazy, and energy-efficiency data collection protocol called *Alep* is proposed. Finally, our comprehensive performance evaluation based on both simulation and prototype implementation shows that Alep improves the quality of data, saves energy, and extends the lifetime of WSNs.

The contributions of this paper are four-fold. First, to the best of knowledge, we are the first to propose a general framework that integrates data quality management in the design of WSNs. Second, we formally define data consistency models as metrics to evaluate data quality. Third, we propose an adaptive, lazy, energy-efficiency protocol to improve data quality and save energy. Finally, a comprehensive performance evaluation has been undertaken based on both TOSSIM [12] and a prototype implementation using 13 MICA2 Motes. The rest of the paper is organized as follows. We first analyze the importance of data quality to WSN applications, then, abstract the specific consistency-related features of WSNs and their applications in Section 2. Section 3 depicts a consistency-driven data management framework. In Section 4, we present the formal definition of data consistency and data dynamics. An adaptive lazy energy-efficient protocol for data collection is described in Section 5. And Sections 6 and 7 report a comprehensive performance evaluation based on TOSSIM simulator and a prototype implementation of 13 MICA2 motes respectively. Finally, related work and conclusion remarks are discussed in Sections 8 and 9 respectively.

2. Data consistency analyses

Data consistency plays a very important role in the success of WSN applications. For example, researchers in sleeping start to consider leveraging wireless sensors to collect the environment information of patients. In order to convince the sleeping research community that WSNs are indeed a viable approach for their traditional self-report survey based approach, researchers usually collect two types of data. One is reported records by the patient, the other is data collected from sensors. We learned that these two sets of data do not match very well. As a result, they do not know which set of data is more appropriate for their research.[1] Thus, we argue that it is vital to have a mechanism to manage and control the quality of data collected by WSNs. This in turn will guarantee the fidelity brought by this promising new approach, and pave the way of wide acceptance in different applications. We conjecture that data consistency is a good metric that can be used to evaluate and control the data qualify. Unlike data consistency in traditional distributed systems, data consistency in WSNs, an application specific concept, has to consider specific features of WSNs and specific requirements of applications. In this section, we first analyze the special consistency related WSN features and then abstract the consistency requirements from the perspective of applications.

[1] Based on oral communication with our colleagues at College of Nursing, Wayne State University. This also inspires us to undertake this research.

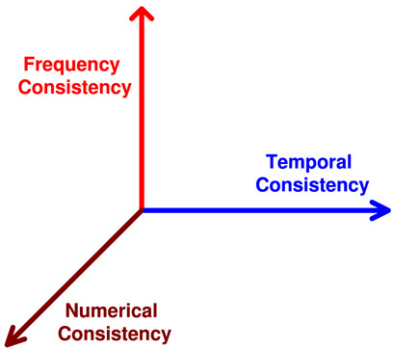


Fig. 1. A three-dimension view of consistency requirements.

2.1. Consistency related WSN features

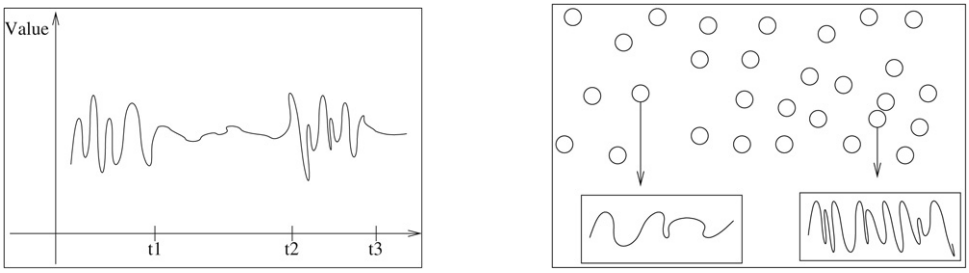
Although a WSN is an instance of a distributed system, there are several significant differences between them. First, WSNs are a resource-constrained system. Due to the constraints of the memory size and the large amount of sampled data, data is usually not stored in sensors for a long period, but it will form data streams to be delivered to the sink(s) or base station(s). As a result, data consistency in WSNs will not focus on the read/write consistency among multiple data replicas as in traditional distributed systems. Instead, data consistency in WSNs is more interested in the spatial and temporal consistency of the same data, i.e. the consistency among several appearances of the data at different locations and at different time. Second, WSN applications may have more interests in a set of data which can depict the trend of the parameter being monitored or report an event. Thus, consistency models for data streams are more important than those for individual data. Third, compared with traditional distributed systems, the unreliable wireless communication is common, rather than exceptional, in WSNs. Thus, in consistency models, the data loss resulting from unreliable wireless communication should also be considered. Furthermore, in previous definition of data consistency [37], the effect of channel noises and attacks are neglected. We argue that attacks are normal nowadays and the security measures should be integrated in the system design from the initial stage.

2.2. Consistency requirements and data dynamics

WSNs are mostly application-specific systems that are widely used in variant applications, which have different data consistency requirements. Besides, WSNs are also data-centric systems, so that data consistency is closely related with data dynamics in the data field.

Considering both individual data and data streams, we argue that the quality of the data should be examined from three perspectives: *the numerical consistency*, *the temporal consistency*, and *the frequency consistency*, as shown in Fig. 1. The *numerical consistency* requires that the collected data should be accurate. Here we have two kinds of concerns on numerical errors: *absolute* and *relative*. Absolute numerical error happens when the sensor reading is out of normal reading range, which can be pre-set by applications. In the case of absolute numerical error, we can remove it and estimate a reasonable value for it. Relative numerical error depicts the error between the real field reading and the corresponding data at the sink. To trade off the resource usage and data accuracy, we can leverage estimation technologies to estimate readings at the sink while still providing the data with the same level of accuracy. As a result, some sensor readings can be dropped to save resource usage. Subsequently, there are relative numerical errors between the real ground truth and the collected data at the sink. The *temporal consistency* means that the data should be

Please cite this article in press as: K. Sha, W. Shi, Consistency-driven data quality management of networked sensor systems, J. Parallel Distrib. Comput. (2008), doi:10.1016/j.jpdc.2008.06.004



(a) Temporal data dynamics. (b) Spatial data dynamics.

Fig. 2. Various data dynamics.

delivered to the sink before or by it is expected. The *frequency consistency* controls the frequency of dramatic data changes and abnormal readings of data streams, i.e. the number of dramatic data changes and the number of readings out of normal reading range in one time interval should be limited by the application specific consistency requirements. Given this definition, we can see that the quality of each individual data is determined by the numerical and temporal consistency, while the quality of data streams is depicted by the combination of three consistency perspectives.

All these three types of consistency are application-specific concepts, thus different applications may have various consistency requirements respectively. For example, in a patient monitoring system, emergency conditions of a patient should be reported to the control panel or care givers as quickly as possible. Otherwise, the patient may be in a dangerous condition. Thus, systems that need quick response or have high real-time requirements usually have high requirements on the temporal consistency. Other systems may have no strict time requirements on the collected data. For instance, a roadside traffic monitoring system that counts the number of passed vehicles in one area may only need the data to be reported twice a day. In this case, data aggregation is more possible because some aggregation functions need to wait until sufficient data are available. However, these kinds of systems may have high accuracy requirements (i.e. numerical consistency requirement) on the collected data. And the applications that have high accuracy requirements may have strict requirements on high frequency requirements for the purpose of quick system reaction when some abnormal data is detected, e.g., some event-detection applications may care more on the dramatic data changes and abnormal readings, which usually represent the occurrence of some interesting events or attacks.

The data consistency should also be integrated with the feature of data dynamics. Here, *data dynamics* means the trend and frequency of data changing. Usually, data dynamics comes from two dimensions, *temporal data dynamics* and *spatial data dynamics*. In the temporal dimension, data changing frequency varies at different time periods. Fig. 2(a) shows the data changing in terms of the time, where data changes very fast before time t1 and between time t2 and t3, while it keeps almost stable between time t1 and time t2. Thus, if we keep the constant data sampling rate, we will get different data consistency (e.g., data inconsistency factor defined in Section 6) during different periods with various data dynamics. On the other hand, in the spatial dimension, the data dynamics differs from area to area. An example of data changing spatially different is shown in Fig. 2(b), where data changes quickly in the right part of the sensor field and slowly in the left part. If we use the same data sampling rate in different locations, we will get different data accuracy, i.e. the collected data may be accurate in the area with lower data dynamics, but not accurate enough for the area with higher data dynamics. Furthermore, the temporal data dynamics and spatial data dynamics affect data consistency at the same time. Thus to maintain the quality of collected data, we should adapt the data sampling rate based on the extent of data dynamics from time to time and from area to area. For example, a simple strategy would be that it samples more data when data dynamics is higher and samples less data when data dynamics is less.

3. Orchis: A consistency-driven data quality management framework

To integrate data consistency requirements and system support for energy-efficiency in WSN design, we propose Orchis [31], a consistency-driven data quality management framework, which consists of four components, including a set of data consistency models customized to WSNs, a set of APIs to management the quality of collected data, an adaptive protocol for data sampling, a set of consistency-driven cross layer protocols to support achieving the goals of data consistency and energy efficiency, as shown in Fig. 3. Next we give an overview of each component one by one.

First, as in traditional distributed systems, it is necessary to define a set of consistency models to evaluate the quality of collected data. With these consistency models, data consistency can be checked both at a single node and at a cluster head or base station after a series of data are collected. Moreover, data consistency models should be application-specific and take into consideration the special characteristics of wireless sensors as abstracted in Section 2.2, such as external inference, constrained resources, and unreliable wireless communication, and environment noises.

Second, we need to develop a set of APIs to manage data consistency. These APIs are essential and enable application scientists to disseminate consistent requirements, check consistency status, manage consistency requirements reactively, and detect and filter deceptive data. These APIs are connected with defined consistency models so that data quality can be evaluated. Also, these APIs provide interfaces for lower layer data collection protocols to efficiently transfer data to the sink.

Third, due to the various data dynamics in different WSN applications, we have to either collect a large amount of data, which is energy inefficient, or devise an adaptive protocol to improve the quality of collected data and take advantage of data consistency by considering data dynamics. The protocol that automatically adapts the data sampling rate according to data dynamics in the data field is necessary to improve data quality and to save energy. Furthermore, the zoom-in feature of the adaptive protocol helps us detect deceptive data and improve the quality of sensed data significantly.

Finally, a set of consistency-driven cross-layer protocols are needed to support the goal of both high data quality and energy efficiency. Diverse data consistency requirements and changing data traffic resulting from adaptive protocols make it difficult to deliver all the messages timely and at the same time save energy.

Please cite this article in press as: K. Sha, W. Shi, Consistency-driven data quality management of networked sensor systems, J. Parallel Distrib. Comput. (2008), doi:10.1016/j.jpdc.2008.06.004

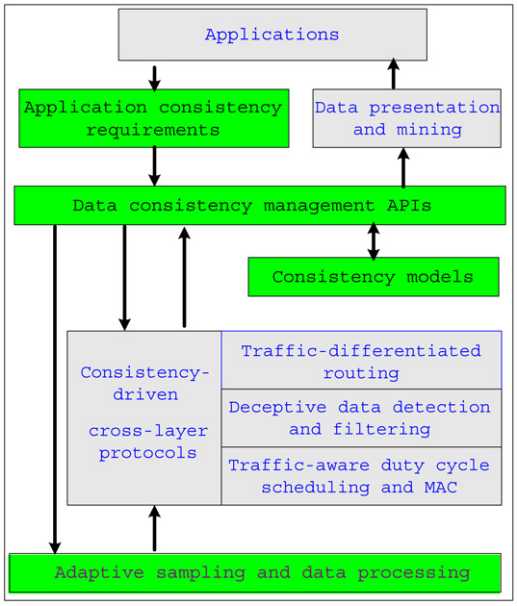


Fig. 3. An overview of the Orchis framework.

We plan to design a suite of cross-layer protocols that allow the system to filter unnecessary data, control traffic, and route packet while keep consistency, including adaptive resource allocation, duty cycle scheduling, and application-aware MAC protocol, to support the cross-layer design. In this paper, we take the first step to propose an adaptive, lazy, energy-efficient data collection protocol, rather than propose the whole suite of protocols.

In our Orchis framework [31], data consistency is controlled in the following ways. At first the consistency requirements are set by application scientists, which are translated into different consistency models. Then, when data is received, the consistency is checked based on consistency models. If the application finds that the consistency is satisfactory, it will continue to use the current parameter in all protocols. Otherwise, a modification to protocol parameters will be enforced to all the sensors through the consistency management APIs. In this way, the management of data quality forms a close loop feedback control and data quality can be bounded by consistency models or application requirements.

Last but not least, we discuss the function deployment of Orchis. In the Tenet architecture [7], researchers from University of Southern California and University of California at Los Angeles observed that the tiered principle is the best way to organize the sensor network. We believe that this hierarchical architecture will be very popular in the future WSN applications. Orchis fits this new architecture very well, because consistency checking may require intensive computation and large storage, thus should be executed at the masters level. From the figure, we can see that consistency models are the core of the Orchis framework. In this paper, we will focus on consistency models and an adaptive protocol that takes advantage of these models. APIs and other protocols in the Orchis framework are not the focus of this paper.

4. Consistency models

A lot of consistency models have been proposed in computer architecture, distributed systems, database, and collaborative systems [20,23,37]. However, these models are usually not applicable in WSNs because of the specific characteristics of WSNs as we analyzed in Section 2. In this section, we define consistency models by considering the consistency among the multiple appearances of the same sampled data at different locations and time, which we consider as replicas of the sampled data. We argue that the consistency among those replicas could be an interesting metric to evaluate the quality of collected data, and consistency models should be defined based on the three consistency dimensions defined in Section 2.2. To this end, we define a general data format first, then model data consistency for individual data followed by data consistency models for data streams. Note that we assume that unnecessary data are detected and filtered by detecting protocols, and an estimation technique [32] is used to fill these data at the sink.

4.1. Data format

Before we formally model data consistency, we define a general data format that will be used in consistency models. Considering both the temporal and numerical consistency requirements and noticing that the frequency consistency requirement (i.e., limits on the number of changes in a certain period [31]) is derived from the numerical values of the data, we define the data format used in consistency models as follows,

$$(p_i, T_{\text{stamp}}, Seq\_Ran, Val, ReT)$$

where p_i denotes that the data is from the ith sensor for parameter p; T_{stamp} specifies the time when the value is sampled and *Seq_Ran* is the range of the sequence number of the reading at the ith sensor for parameter *p*. *Seq_Ran* contains only one number where there is no two continuous readings that have the same value. *Val* is the value of the reading with sequential number in *Seq_Ran*, while *ReT* depicts the remaining time before it is expected by the sink. The initial value of *ReT* is set the same as temporal consistency requirements and the value of other parameters are set locally by the ith node.

As mentioned before, different applications might have various consistency requirements. An example of a consistency requirement is given here, ($NE = 0.2, Max\_T = 3m, Max\_Diff = 1.2, Max\_CHG = 5, Range = [1, 4], Max\_OutRange = 3$), where $NE = 0.2$ means that the application can endure ± 0.2 numerical error; $Max\_T = 3m$ denotes the data should be received at sink in 3 min after it is sampled; $Max\_Diff = 1.2$ and $Max\_CHG = 5$ together define that the number of dramatic changes, the difference between two continuous readings exceeds ± 1.2, should not exceed five; and *Range* and *Max_OutRange* requires that the number of readings with value not in the *Range* should be less than *Max_OutRange*. These consistency requirements are pre-distributed by other protocols. Based on these requirements, we can abstract two consistency semantics: (1) the difference between any two continuous meaningful readings (at the sink) should be larger than 0.2; and (2) if the number of dramatic changes in one minute exceeds five, the collected data is not good enough because the detail of the changing is ignored.

4.2. Consistency models for individual data

We consider consistency for both individual data and data stream. For each individual data, we want to keep the collected data timeliness and accurate, so we will check both temporal consistency and numerical consistency. Moreover, the consistency can be checked at different locations and the data may be delivered by various protocols according to different application consistency requirements, so we model three types of data consistency for individual data, including *the hop-consistency*, *the single-path consistency*, and *the multiple-path consistency*. The hop-consistency means that the data should keep consistency in each hop, while the single-path consistency and the multiple-path consistency imply that data consistency holds when the data is transmitted from

Please cite this article in press as: K. Sha, W. Shi, Consistency-driven data quality management of networked sensor systems, J. Parallel Distrib. Comput. (2008), doi:10.1016/j.jpdc.2008.06.004

the source to the sink using a single path and multiple paths, respectively. *The hop-consistency* is checked at each hop when the data is transferred and it is maintained if the data packet still has sufficient time to be transferred from current node to the sink and the value of the new arrive reading is within the range of the consistency semantics. We define it as follows:

$$HopConsist = (InSemantics(Val, Val_{last})\&OnTime(ReT, EsT_t)) \quad (1)$$

where, *InSemantics* judges the numerical consistency by checking whether the new arriving value and the cached last value follow the pre-defined consistency semantics, and *OnTime* checks the temporal consistency, i.e. $ReT \geq EsT_t$ denotes that the node has enough time to deliver the data to the sink. Because the hop-consistency is checked at each hop along the path, so it is very useful to detect attacks on data and filter redundant aggregated data when data aggregation is applied by using estimation technologies. This reduces the source usage but provides the same level of data accuracy.

The hop-consistency defines the consistency only at each hop, however, the end-to-end consistency between data source and data destination is more important from the viewpoint of applications. We define this type of consistency as the path-consistency, which is usually checked at the sink. According to the different routing strategies and application requirements, we define two types of path-consistency, the single-path consistency using single path routing and the multiple-path consistency using the multiple-path routing or flooding. The difference between them lies in that at most one copy of the same data is reported to the sink in the single-path consistency, while several typically copies of the same data will be received at the sink in the multiple-path consistency. Both types of path-consistency consider two concepts, temporal consistency and numerical consistency. We model the single-path consistency as below,

$$SPathConsist = (InSemantics(Val, Val_{last})\&(ReT \geq 0)). \quad (2)$$

Similar to functions in the hop-consistency, *InSemantics* checks the numerical consistency in terms of the consistency semantics, which is abstracted from consistency requirements. The temporal consistency is reflected by the condition that *ReT* is larger than 0 when the data arrives at the sink. In the multiple-path consistency, several copies of same data will be delivered to the sink to reach a majority agreement. Thus the sink will check the consistency as *k-consistency*, which means at least k copies of the same data should be reported to the sink in time. The multiple-path consistency modeled as below is very useful to detect the faked readings inserted by malicious nodes (i.e. fault detection), which might be very important to some applications.

$$MPathConsist = (InSemantics(Val, Val_{last})\&ReT \geq 0 \quad \&Count(P_i, Seq\_Ran) \geq k). \quad (3)$$

Compared with the single-path consistency, which shares the similar numerical and temporal consistency with the multiple-path consistenncy, the multiple-path consistency has one more requirement on the number of copies with the same value for each data, denoted by $Count(P_i, Seq\_Ran)$. With more than k copies of the same value for the data, a major agreement can be reached to protect malicious data insertion at a very high probability.

4.3. Consistency models for data streams

In WSNs, data are usually collected as data streams. Individual data may not have significant meaning, while they are useful when the set of the data are considered together. For a data stream, denoted as $D = \{d_1, d_2, \ldots, d_n\}$, we propose six types of candidate consistency models to satisfy different consistency levels, including *the strict consistency, the α-loss consistency, the partial consistency, the trend consistency, the range frequency consistency* and *the change frequency consistency*. All these consistency models are application-specific concepts, taking into consideration of application requirements from three consistency perspectives as analyzed in Section 2. Among these six consistency models, the first four consider the different levels of numerical and temporal consistency while the rest two focus on the frequency related consistency. The strict consistency has the most strict requirements to the consistency of collected data, so it can be used in applications that have extremely restrict consistency requirements. To satisfy the strict consistency, all three requirements must be satisfied. First, no data is missed during transmission, i.e. the packet with each sequence number i should be received at the sink. Second, the temporal consistency is satisfied, i.e. for all received data at the sink, $ReT_i \geq 0$. Third, the numerical consistency for the whole data set in terms of consistency semantics is maintained as denoted as $InSemantics(D)$, e.g., any two continuous readings d_i and d_{i+1} in the data set D received at the sink, are out of each other's endurance range. So the restricted consistency is modeled as

$$StrictConsist = (InSemantics(D)\&\forall iReT_i \geq 0 \quad \&\forall i \in [1, n], d_i \in D). \quad (4)$$

The strict consistency differs from the hop-consistency because it is defined based on a set of data and requires no data lose, so it is stricter than the hop-consistency from this point of view. Not all applications require the strict consistency, which may be almost impossible to achieve in a wireless communication based system such as WSNs. Data loss is normal sensor networks using wireless communication, if we allow some data loss during transmission, we get the α-loss consistency, where all received data should keep temporal consistency and at least $1-\alpha$ percent of totally sampled data should be received at the sink. So the α-loss consistency can be modeled as

$$\alpha - LossConsist = (InSemantics(D)\&\forall iReTi \geq 0 \quad \&Counter(D) \geq (1-\alpha) * \max(Seq\_Ran)) \quad (5)$$

where *InSemantics* checks the numerical consistency as before. All the received data are temporal consistent as denoted by $\forall iReTi \geq 0$ and the number of total received data is larger than $1-\alpha$ percent of the number of total sampled data, which is checked based on the sequence number of the received data. For example, if totally n pieces of data should be received based on the value of *Seq_Ran*, and the real received number is $Counter(D)$, we can check if the condition in above formula is satisfied. The α-loss consistency is suitable for applications that have restrict real-time requirements. The value of α is adjustable to cater to the numerical consistency requirements of the applications.

In addition to releasing numerical consistency requirements, we can also release the temporal consistency requirements in case that lots of applications are delay-tolerant, which results in the partial consistency. In the partial consistency, not all the data are required and the temporal consistency are not so strict, thus it is modeled as

$$ParConsist = (InSemantics(D)\&\forall iReT_i \geq -a \quad \&Counter(D) \geq (1-\alpha) * \max(Seq\_Ran)). \quad (6)$$

The partial consistency is similar to the α-loss consistency except that the temporal consistency requirement is released, which is denoted by the condition of $ReT_i \geq -a$. This consistency model is useful in applications where aggregation applies, which have strict numerical consistency requirements but low temporal consistency requirements.

Please cite this article in press as: K. Sha, W. Shi, Consistency-driven data quality management of networked sensor systems, J. Parallel Distrib. Comput. (2008), doi:10.1016/j.jpdc.2008.06.004

If we further release the numerical consistency requirement, we get another consistency model named the trend consistency, which is defined as follows,

$$TrendConsist = (TrendSatisfy(D)) \quad (7)$$

where $TrendSatisfy(D)$, modeled based on the pre-knowledge of the data trends of the application, detects if the trend of data streams is maintained. Mechanisms are needed to evaluate the valid trends. For instance, we might utilize some algorithms from the signal processing field to evaluate the quality of data streams, e.g., frequency domain features. This consistency model matches the trend requirement (Section 2) of some WSN applications very well, which could be used in attack-resilient data collection protocols.

Now we consider the abnormal data readings in data collection. In certain applications, the application scientists may have prior knowledge of the normal data range of their application. This is very helpful to filter erroneous readings, which are resulted from a variety of reasons, including intended attacks. Also, if the number of abnormal readings exceeds a certain number pre-set by the application, the application scientists may need to check the abnormal phenomenon. The notification of the abnormal phenomenon will be triggered by a violation to the range frequency consistency. Here we define the range frequency consistency as follows,

$$RangeConsist = (\forall i \in [1, \ldots, k], Count(V_i not \in Range) < Max\_OutRange) \quad (8)$$

where *RangeConsist* denotes the range frequency consistency. V_i shows a number of k readings in a time interval and *Max_OutRange* denotes the application pre-set maximum number of readings that may be out of the normal range, *Range*, in one period. This consistency can be checked both locally at each sensor and at the sink. The violation of the consistency is denoted by the condition that the number of outrange readings exceeds a pre-set maximum allowed number. Further action are usually needed by the application scientists when this type of consistency is violated.

In some other applications, application scientists may care a lot about the detail of the data changes, thus we define the change frequency consistency to detect whether the changes of the sensor reading are abnormal. The detail of the change frequency consistency is denoted as follows,

$$ChangeConsist = (\forall i \in [1, \ldots, k], Count(|V_{i+1} - V_i| > Max\_Diff) < Max\_CHG) \quad (9)$$

where ChangeConsist depicts the change frequency consistency. V_i is a set of total k readings in a time interval; *Max_Diff* is the pre-set maximum difference between two continuous readings when the consistency holds, and *Max_CHG* means the maximum number of dramatic changes, which is defined as the case that the difference between two continuous readings exceeding *Max_Diff*, in one interval. With this consistency, we can either prevent the data from changing too dramatically or dynamically change sampling rate to zoom in and observe the details [32]. The violation of the consistency is denoted by the condition that the number of dramatic changes exceeds a pre-set threshold. The observation of violation of this consistency may also result in a request of application scientists involvement.

Moreover, if we consider the spatial correlation of sensing parameters, we also need to satisfy the spatial consistency, which is implied by the geographical characteristics of collected data. For example, when we are sampling temperature, we may have some pre-knowledge of the geographical distribution of temperature in this area, thus spatial consistency should be checked when the data is collected. Finally, as we know that there may be some relationship between different parameters, e.g., the speed of vehicles may correlated with the density of the vehicles on the road, so we can further explore the data relationship among different parameters by defining consistency models for them.

In summary, we propose a set of basic but powerful consistency models for data quality management in WSNs from the perspective of temporal, numerical and frequency consistency. These models can be used as metrics to evaluate the quality of collected data both in aggregated format and non-aggregated format. With these proposed basic consistency models, various applications can find their suitable consistency models for their specific data quality requirements by adjusting the parameters in these models or composing these basic consistency models to form complicated ones. For example, the two frequency consistency models can be combined to control the dramatic data changes and the abnormal readings in a time interval. The partial consistency and the two frequency consistency can also be composed to set all numerical, temporal and frequency consistency requirements. Moreover, these consistency models can be combined with the spatial consistency as well. Furthermore, various applications should make a trade off between energy efficiency and data consistency based on their energy budget, which remains an open problem in the community.

4.4. Usage of consistency models

Above defined consistency models can be used in different applications. In the Orchis framework, we also devise a set of APIs, including *checking the current consistency status* (`CheckStatus()`), *setting consistency requirements* for new parameters (`SetReq()`), *updating consistency requirements* (`UpdateReq()`), and *getting support from lower layer protocols*, to manage data quality, the details of these APIs are referred in [31].

With the APIs, we can manage data quality of the collected data based on the proposed consistency models according to application consistency requirements. When an application needs to sample several different parameters and these parameters have different consistency requirements, we can set different models for these parameters, and check the consistency against these models. For example, in the applications of SensorMap [19], several parameters are monitored with different consistency requirements. For each parameter, we argue that a suitable consistency model can be leveraged, so different consistency requirements can be satisfied. For instance, the *k-consistency* (*MPathConsist*) may be set for alarm data, while *ParConsist* is enough for temperature data. To manage the data quality in this application, we can integrate our APIs with the DataHub in the SensorMap architecture. When the data arrives at the DataHub, the consistency model is checked. This direction itself deserves further investigation.

The process of consistency management consists of three steps. First, the consistency requirements are translated into suitable consistency models and distributed. Second, the consistency will be checked with support of consistency-checking algorithms at the sink after an amount of data are collected. Third, if the sink finds that current consistency cannot be satisfied because of the constrained resources, it might release consistency requirements. If the sink finds that the quality of current collected data is not satisfactory, it might increase consistency requirements. These update will be distributed to related nodes, who will in turn change their data collecting strategy according to the new consistency requirements.

Here, we give an example of how to use these APIs. For example, if irregular animal movements are observed in an area, a request to monitoring the temperature of that area is issued, which requires that the collected data should satisfy 10%-*LossConsistency* i.e., $SetReq(temp, Set(area), T_{req}, Val_{req},$

Please cite this article in press as: K. Sha, W. Shi, Consistency-driven data quality management of networked sensor systems, J. Parallel Distrib. Comput. (2008), doi:10.1016/j.jpdc.2008.06.004

10%-*LossConsist*, *area-cast*), where *temp* denotes the name of parameter; *Set*(*area*) and *area-cast* show that the consistency requirements will be sent to all the nodes in that *area* using *araa-cast*. T_{req}, Val_{req} and 10%-*LossConsist* specify the consistency semantics. After the application scientist collects enough data from the monitoring area, he/she will call *CheckStatus*(*temp*, *Set*(*area*), *last-one-hour*, 10%-*LossConsist*) to check whether the data received in last one hour satisfy the requirements specified by the consistency model, 10%-*LossConsistency*. Based on result of the function call, the application scientist makes a decision to tune consistency, e.g., if the application scientist thinks that the quality of the data is good enough, he/she will do nothing; otherwise, he/she will update the new consistency requirements by calling *UpdateRqe*(*temp*, *Set*(*area*), $\delta(T)$), $\delta(Val)$, 5%-*LossConsist*, *area-cast*). On the node side, the node will update the consistency requirements locally after receiving the new update request. The whole process forms a close-loop feedback control. In this way, the quality of data can be guaranteed.

5. ALEP: An adaptive, lazy, energy-efficient protocol

In this paper, we intend to save energy by estimating the value of the sensing data and adapt the data sampling rate to improve the quality of collected data. This in turn will reduce the number of delivered messages, which is the most significant energy consumption factor in WSNs [18], In this section, we first introduce the rationale of our design, then give the details of the protocol.

5.1. Rationale

Estimating data at the sink is used to save energy but it may hurt data accuracy. There are two extremes between data accuracy and energy efficiency. For energy efficiency purposes, we should gather and deliver very small amount of data. Subsequently, the gathered data cannot satisfy the consistency requirements of the application. On the other hand, if we always keep high sampling rate and deliver a lot of messages to get very accurate data, sensors will run out of energy very quickly. Thus, we should make a tradeoff between energy efficiency and data accuracy. In this protocol, we intend to achieve the goal of α-loss consistency, which provides enough quality for most WSN applications. From our observation, we find that data dynamics varies temporally and spatially. Furthermore, we also find that it is easier to get accurate estimation when data dynamics is low, however it is difficult to get accurate estimation when data dynamics is high. Thus, the sampling rate should adapt to data dynamics in both temporal and spacial ways. When the data dynamics is high, the sampling rate should be raised to improve the data accuracy, otherwise, it should be decreased to reduce the number of delivered messages. In addition to adapting the data sampling rate to data dynamics, we can improve the estimating techniques so that the number of delivered messages can be dramatically reduced by using estimated data to replace the sensing data. Besides, as mentioned in literature [16], sending a message with long length is more energy efficient than sending several messages with short length. Thus, we intend to integrate multiple short messages into one big message.

In summary, our proposed *Alep* protocol consists of three components, *adapting the sampling rate, keeping lazy in transmission based on consistency-guaranteed estimations, and aggregating and using long length packet*. These methods are described in detail in the following subsections.

5.2. Model for data dynamics

Before giving the details of the adaptive protocol, we first model data dynamics. To describe data dynamics, we define a number of windows to observe the data readings. Two parameters, $winSize$ and $winNum$ are defined to model the dynamics of data. $winSize$ denotes the number of readings in one window, $winNum$ specifies the number of windows in one observation. Thus the total number of readings in one observation is $Num_{rd} = winSize * winNum$. Since the data dynamics reflects *the frequency of data changing*, so we first define the frequency of the data changing as the number of data changing in one observation:

$$Num_{chg} = \{Cnt(i) || r_{i+1} - r_i| > B \& i \in [0 : Num_{rd}]\}$$

where, $Cnt(i)$ is the number of is satisfying the conditions; r_i and r_{i+1} are the ith and i+1th readings separately. And $B = C(p)_{bnd}$ is the accuracy bound for this parameter. Based on this definition, we define the data dynamics (DYN) as the average number of changing in one monitoring window.

$$DYN = \frac{Num_{chg}}{Num_{rd}} * winSize$$

From above definition, we can find that the data dynamics is defined based on time period, i.e., inside the window of observation. By adjusting the value of $winSize$ and $winNum$, we can get the data dynamics with various sensitivity.

5.3. Adapting the sample rate

The process of adapting the sampling rate is a process of reinforced learning based on data readings. Based on the value of DYN, we define the adaptation strategy of the sampling rate as

$$R_{smp} = \begin{cases} \left\lceil \frac{DYN - Ave_{chg}}{Df_{bnd}} \right\rceil * R_{cr}, & DYN > Ave_{chg} \\ \frac{Ave_{chg} - DYN}{Df_{bnd}} * R_{cr}, & DYN \le Ave_{chg} \end{cases}$$

where, R_{smp} is the adapted sampling rate; R_{cr} is the current sampling rate. Ave_{chg} is the normal average changes happen in one window size; and Df_{bnd} bounds maximum difference between the observed value of data dynamics and the normal average changes, i.e., if DYN is larger than Ave_{chg} and the difference exceeds the bound, the sampling rate will be increased; when DYN is much smaller than Ave_{chg}, the sample rate will be decreased. Different applications could define their specific up-bound and low-bound of sampling rates. However, these bounds cannot exceed the maximum bound and minimum bound. Here we define the maximum bound of the sampling rate as the maximum bandwidth of the sensor and the minimum bound of the sampling rate as the smallest sampling rate that satisfies the Nyquist-Shannon sampling theorem [38], which should also satisfy the requirement of collecting at least α-percent of all data. The sampling rate adaptation strategy learns from the previous data dynamics, and uses the most recent data dynamics to estimate the nearest future data dynamics. It is rational because we believe that the data dynamics will not change dramatically in most cases. Also, the data history is limited by the number of windows and the window size in one observation. We can adjust the length of history based on the window size and the number of windows to control the sensitivity of the adaption.

Please cite this article in press as: K. Sha, W. Shi, Consistency-driven data quality management of networked sensor systems, J. Parallel Distrib. Comput. (2008), doi:10.1016/j.jpdc.2008.06.004

5.4. Keeping lazy in transmission

One way to reduce the number of delivered messages is to keep lazy in transmission, i.e. only sending the messages that are necessary to be sent because we think that if the receiver can estimate an accurate enough value for the current reading, the message need not to be sent, i.e. if the data consistency requirement can be hold, the messages are not necessary to be sent.

In this protocol, every sensor caches the last transmitted reading for every parameter for all potential senders that may deliver message to it, and it uses the cached values as the estimation of the current reading if no new reading comes. To check the consistency for new samplings, the sensor will use the current reading as the real value and the cached value as the estimated value. If the difference between them is within the consistency bound, the sender will not send this piece of data, i.e., keeping lazy. For example, in an application which monitors the temperature of a sensor field, when a sensor gets a reading of value 3.7, and the cached last reading is 3.5 which is within the consistency bound of 0.3, the new reading will not be sent. And the possible receivers of the sampling use the last cached sampling to replace it by assuming the value unchanged. This approach has two advantages: easier to estimate the undelivered data locally and only keeping a copy of very small amount of data.

In the case of the aggregated data, every receiver caches a copy of the latest aggregated value calculated from senders. After it applies the aggregation function, it will compare the new calculated value with the cached value. If the difference between them is within the consistency bound, the sender will keep silent. For the aggregated data, the receiver has to wait for the new reading from all the senders for a period of time. If there are still data absent from some senders, the receiver will use the cached data to substitute the current reading and calculate the new aggregated value.

5.5. Aggregating and delaying delivery

Another aspect of the Alep protocol is to integrate several pieces of data into one message to reduce the number of messages and to delay the delivery when the data temporal consistency is not violated. In our design, each sensor maintains a data queue where the received data are stored. The data in the queue are sorted according to the application specific priority and the requirement of temporal consistency. When there are free space in the queue and the consistency is satisfied, the sensor will keep sleeping instead of sending data to its parent node. The temporal consistency is checked by comparing the estimated time to deliver the message to the sink and the time the data is expected at the sink. In our application, the expected time to deliver the message to the sink can be estimated based on the number of hops to the sink. For example, if we assume that it takes T_{dev} to transmit one message from the child to the parent, then we can estimate the time it takes for current sensor to deliver a message to the sink is $T_{dev} \times H_{js}$, where H_{js} is the number of hops from the current sensor to the sink. Then the time bound for the data is the sum of the estimated time plus one time slot, which denotes the time between two reporting points according to the TDMA schedule.

6. Performance evaluation: Simulation

To evaluate the performance of the proposed protocol, we have implemented the protocol in TinyOS using the TOSSIM [12] environment and compared with two other protocols, *Simple* which is a TDMA-based data collection protocol and *Lazy* which only has the lazy feature of the proposed protocol. In the rest of this section, we will describe the simulation setup and the performance metrics first, followed by the performance evaluation in simulation environment. The results of a prototype implementation and evaluation is reported in the following section.

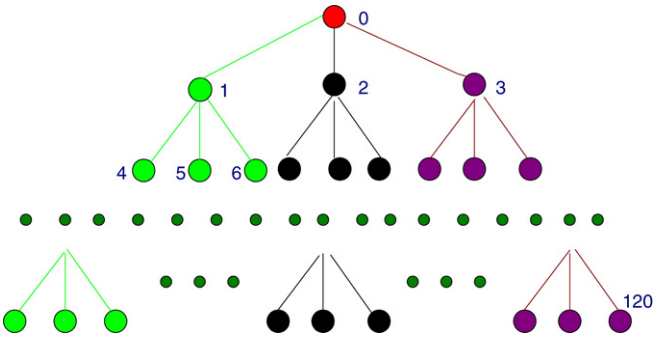


Fig. 4. A tree-structured sensor network used in the simulation and prototype. The numbers next to each node is node ID or Mote ID.

6.1. Simulation setup and evaluation metrics

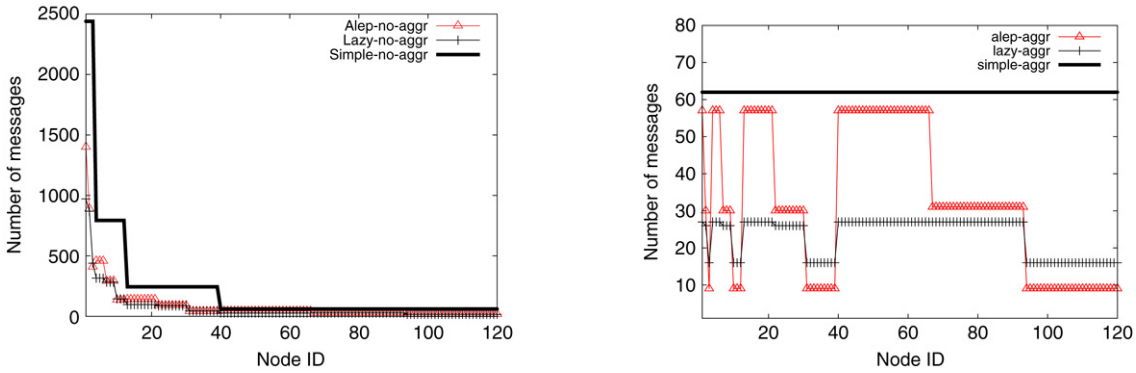
In our simulation, 121 nodes are connected forming a four layer complete tree, as shown in Fig. 4, where all the internal nodes have three children and the root acts as the sink. The sensors periodically collect data from its children and report the readings to its parent based on a TDMA schedule.[2] Note that the proposed protocol is a general principle, which can be applied in different applications and different network structures, although we evaluate it in the scenario of monitoring application with tree structured networking.

Each sensor node acts as a multiple functional sensor, which can sample three parameters: Temperature as `Temp`, Pressure as `Press`, and Rain-index as `Humid`. To evaluate the proposed protocol in different data dynamics environments, we intentionally make these three parameters have different dynamic characteristics. For example, the reading always changes faster for `Temp`, relatively stable for `Press`, while medium for `Humid`. To simulate spatial data dynamics, we intentionally separate the whole area into three sub-areas with different data dynamics as shown in different colors (gray levels in B/W print out) in the figure. The reading changes faster in the left subtree area, relatively stable in the right subtree area, and medium in the middle subtree area.

The goal of the *Alep* protocol is to save energy by reducing the number of delivered messages while satisfying the data consistency requirements. Thus, we use three metrics to evaluate our approach. To measure the energy efficient property, we count the total number of delivered messages and the dropped voltage at each sensor (in prototype evaluation), and to examine the tradeoff between the energy efficiency and data consistency. Thus, *Alep* will be examined in three ways: *Does this protocol reduce the number of the messages and extend the lifetime of WSN? Does this protocol improve the accuracy of data*? And *what is the tradeoff between the number of delivered messages and the data accuracy*? To answer the question of the effect of reduced messages to data consistency, we propose a new performance metric called *data inconsistency factor (DIF)*, which is defined as the total variance between the gathered data in the sink and real data, i.e. $V = \sum_1^n (d_{rcv} - d_{fld})^2$, where, V is the value of variance and n is the number of the collected data; d_{rcv} and d_{fld} are the reading value received at the sink and the real value sampled at the data field respectively. The more accurate the data, the smaller the variance.

[2] Ideally we should use a real trace to drive our simulation, however, at this stage we are not aware of any public available real traces. Furthermore, we decide to adopt a controlled topology rather than a random topology since controlled topology is closer to the real deployment and nature phenomenon.

Please cite this article in press as: K. Sha, W. Shi, Consistency-driven data quality management of networked sensor systems, J. Parallel Distrib. Comput. (2008), doi:10.1016/j.jpdc.2008.06.004



(a) Without aggregation.

(b) With aggregation.

Fig. 5. Comparison of number of delivered messages.

6.2. Number of delivered messages

Usually collecting more data is a way to improve the data accuracy; however, by adapting the sampling rate to fit the feature of data dynamics and keeping lazy when data is in the range of consistency, data accuracy can be improved without significantly increase the number of delivered messages. Moreover, in some cases when the data dynamics is low, the data consistency can be kept even by delivering less number of messages.

Fig. 5 lists the number of delivered messages at each sensor without and with aggregation respectively. The *x*-axis is the ID of each sensor, and the *y*-axis denotes the number of delivered messages. Note that the *y*-axis of Fig. 5(a) and (b) are at different scales. As a matter of fact, the number of delivered messages for all approaches reduces significantly when aggregation is used. From the two figures, we can see that *Simple* generally delivers the most number of messages and *Lazy* transfers almost the least number of messages in both cases of with and without data aggregation. This is because *Lazy* filters a lot of unnecessary messages.

These three approaches have totally different performance in terms of the number of messages delivered. In the case of without data aggregation shown in Fig. 5(a), the sensors are classified to four types based on the layer in the tree using *Simple*, i.e. sensors in the same layer using *Simple* delivers the same number of messages. However, using *Alep* and *Lazy*, the sensors transmit different number of messages because of the variant data dynamics in the different areas. For example, among sensors located at layer 3, sensors with ID between 13 and 21 transfer 140 messages because the high data dynamics of the monitoring area, while the sensors with ID between 31 and 39 only deliver 41 messages because the low data dynamics of the monitoring area, which is less than $\frac{1}{3}$ of that in the high dynamics area. The similar results exist in the case with data aggregation in Fig. 5(b), where all the sensors deliver the same number of messages using *Simple*, while the sensors using *Alep* and *Lazy* located at different areas transmit different number of messages, i.e. the sensors located at high dynamics area deliver 57 messages but the sensors located at low dynamics area only send 9 messages, which denotes that *Alep* does adapt the data sampling rate to the dynamics of the data.

Comparing with *Lazy*, we observe that the sensors using *Alep* send more number of messages than using *Lazy* at the area with high data dynamics (e.g., node 13–21) but send less number of messages than that of using *Lazy* at the area with low data dynamics (e.g., node 31–39). This is because the sampling rate is increased considerably in the area with high data dynamics and decreased a lot in the area with low data dynamics. From above analysis, we conclude that *Lazy* can always reduce the number of delivered messages so that it has potential to save a lot of energy, and *Alep* usually does not increase the number of delivered messages and reduce the number of delivered messages a lot when the data dynamics is low.

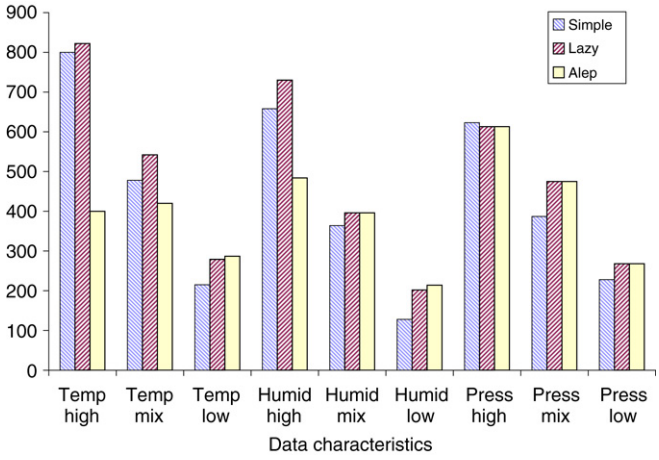


Fig. 6. Comparison of data inconsistency factor.

6.3. Data inconsistency factor

From above analysis, we can see that *Lazy* and *Alep* can largely reduce the number of delivered messages. However, delivering less message means that there are more data estimated at the sink, which may result in the degradation of the data quality. In this subsection, we examine the effect of unsent messages to the data accuracy.

We use the data inconsistency factor to evaluate the quality of the collected data. Fig. 6 reports the relationship between the data inconsistency factor and different monitoring parameters with variant data dynamics. In the figure, the *x*-axis is different data types with variant data dynamics and the *y*-axis represents the calculated data inconsistency factor of the collected data. Three types of parameters with different data dynamics are monitored, among which `Temp` has relatively higher data dynamics than `Humid` and `Press`, while `Press` has relatively lower data dynamics. Furthermore, for each parameter, data dynamics also varies according to different areas, i.e. each parameter has three types of data dynamics, high, high first then low denoted as mix, and low. Thus, there are totally nine sets of data with variant data dynamics.

In the figure, we note that when the data dynamics is higher, the value of data inconsistency factor is larger, e.g., the `Temp high` has a larger data inconsistency factor than `Temp mix` and `Temp low`, and

Please cite this article in press as: K. Sha, W. Shi, Consistency-driven data quality management of networked sensor systems, J. Parallel Distrib. Comput. (2008), doi:10.1016/j.jpdc.2008.06.004

`Temp high` also has a larger data inconsistency factor than `Humid high` and `Press high`. The reason is that when the data dynamics is high, it is more difficult for the sink to estimate the correct data. From the figure, we also find that *Alep* has much smaller data inconsistency factor than that of *Simple* and *Lazy* when the data dynamics is high, while it has larger data inconsistency factor than that of *Simple* and has the same data inconsistency factor as *Lazy* when the data dynamics is low. This result shows that *Alep* indeed makes the sampling rate fit the feature of data dynamics, i.e. when the data dynamics is high, it will use higher sampling rate to gather more data so that to make the variance small. Otherwise, it will sample less data to save energy.

Furthermore, the data inconsistency factor increases very fast with the increasing of data dynamics using *Simple* and *Lazy*, but increases slowly using *Alep*. As a result, *Simple* and *Lazy* may not collect enough accurate data when the data dynamics is high, i.e., the data inconsistency factor exceeds the data consistency requirements of the application. However, *Alep* can keep the data inconsistency factor low by adapting the data sampling rate to data dynamics. We should also notice that *Alep* improves the data accuracy meanwhile somehow reduces the number of delivered messages as shown in Section 6.2.

Comparing *Lazy* with *Simple* in terms of the accuracy of the collected data, *Lazy* has very close value of data variance as *Simple*, however, in Section 6.2 we know that *Lazy* delivered less messages than *Simple*, which means that the dropped messages are not necessary to be transferred to the sink. Thus, we conclude that lazy delivering can reduce the number of delivered messages, while the approach of adapting the data sampling rate to data dynamics can significantly improve the data accuracy. It is good to integrate those two approaches to collect accurate data in an energy-efficient way.

6.4. Successful delivery rate

We have shown that *Alep* can decrease the value of data inconsistency factor at most time, improving the numerical consistency. Here, we show that *Alep* and *Lazy* can improve the temporal consistency considerably, as shown in Fig. 7, where the *x*-axis denotes the TDMA capacity, which specifies the number of maximum delivered messages in one TDMA round; and the *y*-axis shows the successful delivery rate in terms of different TDMA capacities by using different protocols. This experiment collects data in scenario of various data dynamics using all three protocols. In the figure, we can easily see that the successful delivery rate increases with the increasing of the TDMA capacity, because more data can be delivered on time when TDMA capacity is increased. We also observe that both *Alep* and *Lazy* have higher successful delivery rate than *TinyDB*. For example, when TDMA capacity is four and the data dynamics is normal, both *Alep* and *Lazy* successfully deliver 57.1% messages while *TinyDB* only delivers 25% messages on time. With the limited TDMA capacity, a limited number of messages can be delivered via all three protocols, but less messages need to be sent using both *Alep* and *Lazy* than using *TineyDB*, so the successful delivery rate is higher when using *Alep* and *Lazy*. *TinyDB* has the same successful delivery rate in cases of different data dynamics, while *Alep* and *Lazy* have higher successful delivery rate when the data dynamics is low, because more data can be accurately estimated. For example, using *Alep*, the successful delivery rate is 100% when TDMA capacity is 4 and data dynamic is low, and it is 26.7% when TDMA capacity is 4 and data dynamic is high. This result shows that the increasing of TDMA capacity is more effective in case of *Alep* than *TinyDB*. We attribute this to the adaptation and lazy delivery in the *Alep* protocol. Compared with *Lazy*, *Alep* has a little lower successful delivery rate, however, we argue that *Alep* has much better numerical consistency than *Lazy*.

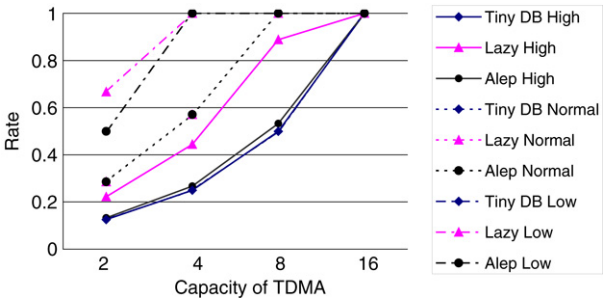


Fig. 7. Comparison of the successful delivery rate vs. TDMA capacity, the number of maximum delivered messages in one TDMA round.

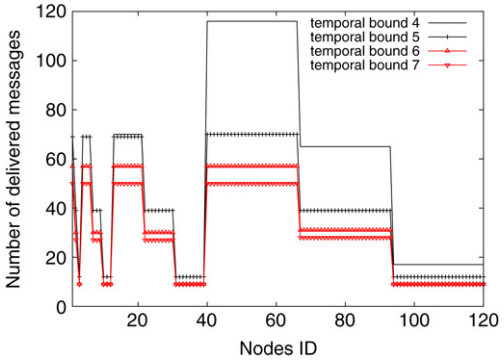


Fig. 8. Number of delivered messages with various temporal consistency bounds.

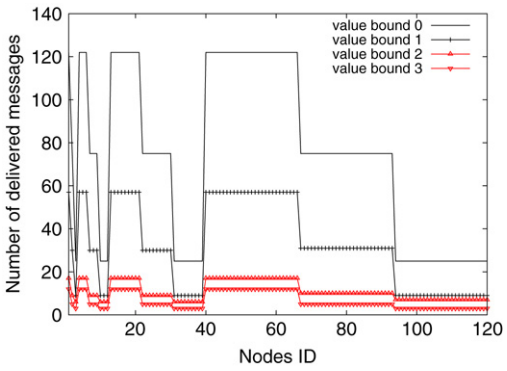
6.5. Tradeoff between energy efficiency and data consistency

The performance of Alep is largely decided by the two key factors, *the temporal consistency bound* and *the numerical consistency bound*. Here we study the effect of these two factors.

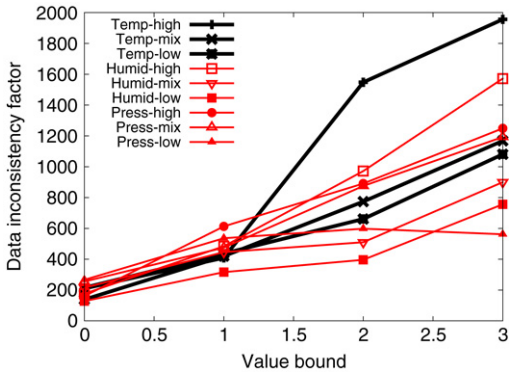
First let us consider the effect of the temporal consistency bound to the energy efficiency and data numerical consistency. If we release the temporal consistency of data, the same set of data will be delivered to the sink regardless of different arrival times. Thus changing the temporal consistency bound will not affect the data inconsistency factor of the collected data. However, releasing the temporal consistency bound does affect the number of delivered messages. Fig. 8 displays the relationship between the number of delivered messages and the different temporal consistency bounds ranging from 4 units to 7 units, which is the maximum time to transfer a message to the sink assuming each hop taking one unit time. In the figure, the *x*-axis is the ID of the sensors and the *y*-axis is the number of delivered messages. From the figure, we can see that the increasing of the bound of temporal consistency results in the decreasing of the number of total delivered messages. When the temporal consistency bound is tight as 4, some sensors deliver more than 110 pieces of messages because data combination is not possible. While the temporal consistency bound is raised to 7, sensors deliver only about 50 pieces of messages. Thus, releasing the bound of temporal consistency can reduce the number of delivered messages. Subsequently, this will reduce the energy consumption.

We also examine the effect of the numerical consistency bound. Fig. 9(a) shows the number of delivered messages with the relation to the variant value constraints. In the figure, the *x*-axis is the ID of the sensors and the *y*-axis shows the number of delivered messages. From the figure, we can see that when the numerical consistency bound is enlarged, the number of the delivered messages is decreased very fast. Fig. 9(b) shows the relationship between the data inconsistency factor and the value of the data consistency bound. The *x*-axis is the different value bounds

Please cite this article in press as: K. Sha, W. Shi, Consistency-driven data quality management of networked sensor systems, J. Parallel Distrib. Comput. (2008), doi:10.1016/j.jpdc.2008.06.004



(a) Number of messages.



(b) Data inconsistency factor.

Fig. 9. Effect of numerical consistency bound.

and the *y*-axis depicts the value of the data inconsistency factor. In the figure, when the data consistency bound is released, the data inconsistency factor increases very quickly, especially when the data dynamics is high. Thus we argue that there is a tradeoff between the data consistency and the energy efficiency. Releasing the data consistency bound results in both energy efficiency and larger data inconsistency factor, so the application should decide the data consistency bound based on its specific data consistency requirements. If the application cares little to the data consistency, it may raise the bound, otherwise, it has to use a tighter bound.

7. Performance evaluation: Prototype

Although the simulation results have shown the advantage of Alep and Lazy, we further evaluate them in a prototype implementation, where energy consumption is measured more directly and accurately. In the prototype, 13 MICA2 Motes form a tree similar to that in the simulation except that it has only two layers, the root of the tree is connected to a desktop. Other configurations of each sensor is the same as that in the simulation. For comparison purposes, three algorithms are implemented, including Alep, Lazy and TinyDB, a simplified version of TinyDB [15] without data aggregation.

In the implementation, we find that the program developed for TOSSIM could not be executed at MICA2 Motes directly because TOSSIM does not enforce the same strict memory constraints as that in MICA2 Motes. However, to compare these three protocols, we need to feed them enough data, either synthesized or real traces, to show the differences. This is a challenge in the prototype implementation because we cannot use the data sampled from the sensor board directly. As a substitution, trace based approach and a data generator are considered. However, we argue that the trace based data feeding does work in sensors smoothly. First, it is impossible to hold a long trace in the program flash memory of tiny sensors, which have only 128 kbytes in total. Second, there are also some disadvantages to store the trace file at the measurement flash of MICA2, which has 512 kbytes in total. For example, as denoted in [33], the current for reading MICA2 sensor board is 0.7 mA, while the current for reading and writing EEPROM is 6.2 mA and 18.4 mA separately. These overheads are comparable with that for receiving and sending messages. Thus, a lot of energy will be consumed by reading data from measurement flash so that the energy consumption of the protocol cannot be accurately evaluated. Furthermore, the total possible access time of the measurement flash is limited [33]. So, we decide to design a data generator for evaluation purposes.

7.1. SDGen: Sensor data generator

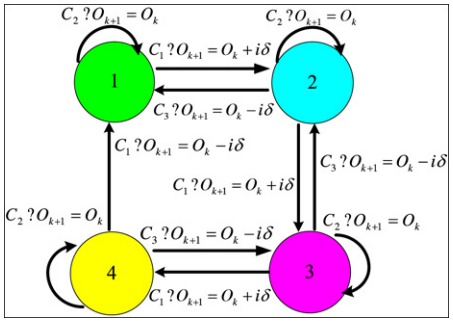
We intend to design a general sensor data generator, SDGen, which can be used by other researchers for evaluating their protocols and algorithms as well. SDGen is a simple data generator based on the a finite state machine as shown in Fig. 10(a). SDGen remembers the latest output as O_k and a state as one of the four circles in the figure. To generate a new sampling, it first generates a random number. Based on the value of the random number, the conditions, C_1, C_2, and C_3, are checked and the state of the machine is transferred. The action after the satisfied condition is executed and an new output O_{k+1} is produced based on O_k. For example, if C_1 is in the range of [0.0, 0.9], and the random number generated is 0.6, then $O_{k+1} = O_k + \delta$ is the output and the machine transfer state from 1 to 2; if C_2 is in the range of (0.9,1.0], and the random number generated is 0.95, then the output is $O_{k+1} = O_k$ and the machine state is not changed.

Our SDGen has several advantages. First, it is very easy to implement and it only needs to remember the last state and the last output, which reflects the fact that the next sampling is usually closely related with the latest previous reading. Of course, we can make it more complicated by remembering more previous readings. Second, the same set of random data can be generated if we fix the value of the random seed, which satisfies our requirements of feeding the same set of data to all protocols. Actually, in our implementation, we find that it is important to keep the order of the generated random number, because we need to gradually generate samplings for three parameters. We take the following strategy in our implementation: all samplings are read from a short array, which stores a set of latest generated data and the order of the generation is controlled by the length of the array. Third, SDGen can generate sampling series with different data dynamics by adjusting the parameters, including the value of i and the condition C_i. For instance, if we make C_2 to be in [0.0, 0.8], which means that any two continuous samplings are the same with a probability of 80%, dynamics of the generated data will be very low, while if we make C_2 to be in [0.0, 0.1], dynamics of the generated data will be much higher.

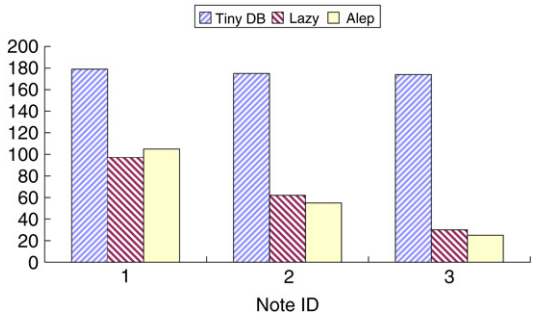
7.2. Comparison number of delivered messages

As argued in [18], the number of delivered message dominates the energy consumption in WSN applications. Thus we first compare the number of delivered messages using these three protocols. As hinted in [30], the lifetime of a WSN is decided by a set of communication intensive sensors, which are layer one nodes in a tree-based structure. Thus, we compare the number of delivered messages of these layer one nodes only.

Please cite this article in press as: K. Sha, W. Shi, Consistency-driven data quality management of networked sensor systems, J. Parallel Distrib. Comput. (2008), doi:10.1016/j.jpdc.2008.06.004



(a) State transition diagram of sensor data generator.



(b) Number of delivered messages.

Fig. 10. SDGen and comparison of number of delivered messages.

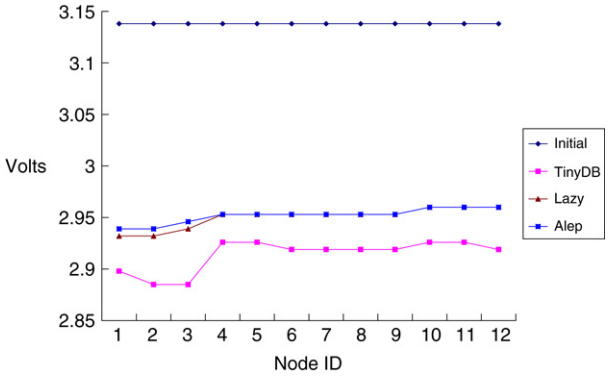


Fig. 11. Comparison of voltage dropping of different protocols.

Fig. 10(b) shows the results of number of delivered messages. In the figure, the *x*-axis is the Mote ID, which also depicts the area with specific data dynamic feature and the *y*-axis is the number of received messages at the sink from the corresponding Mote. We find that in all cases, both Alep and Lazy send much less number of messages than TinyDB. For example, when data dynamics is high, TinyDB sends about 170 messages, while Alep and Lazy send only about 100 messages. When data dynamics is low as for Mote 3, TinyDB still sends about 170 messages, but both Alep and Lazy send less than 30 messages. With the gradual decreasing of data dynamics from Mote 1 to Mote 3, TinyDB sends almost the same number of messages, while both Lazy and Alep send less messages. Compared with Lazy, Alep sends comparable number of messages in all cases; however, it sends a little more messages when data dynamics is high, and sends a little less messages when data dynamics is low. This denotes that Alep does adapt the data sampling rate to match data dynamics. This result matches the result we get from simulation very well.

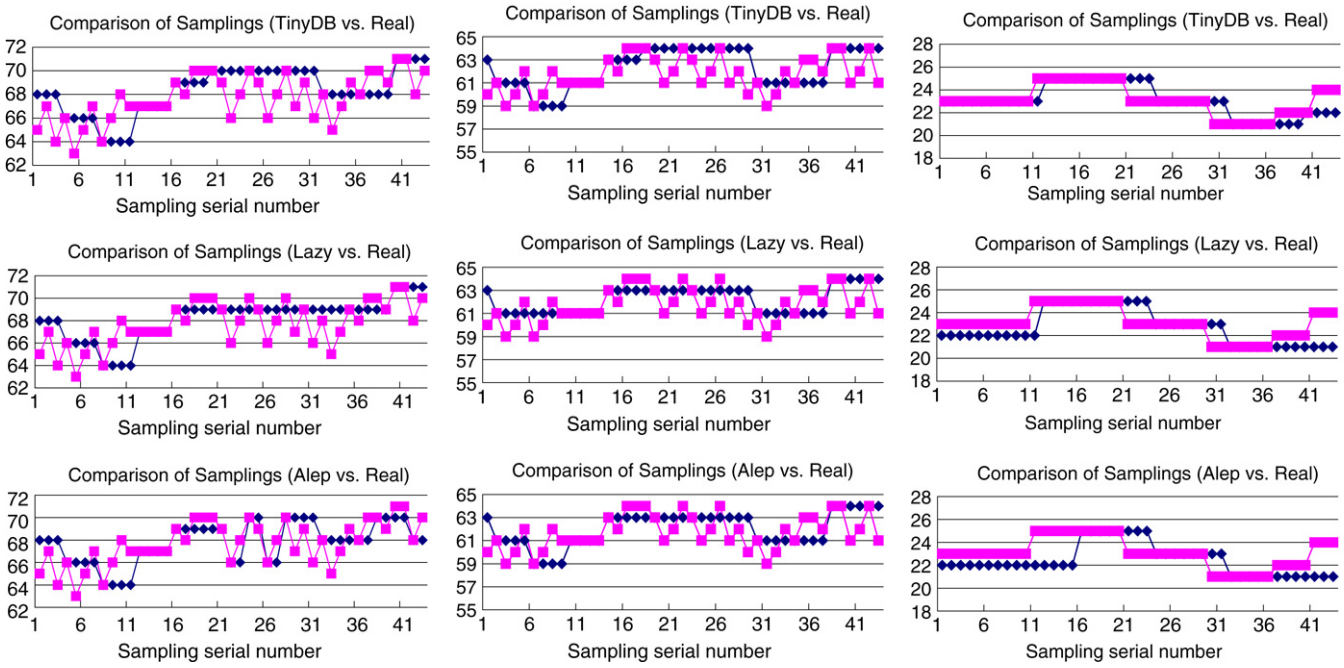
7.3. Comparison of energy consumption

We use the voltage drop to show the energy consumption in these three protocols as shown in Fig. 11, where the *x*-axis is the Mote ID and the *y*-axis is the value of the initial and final voltage read from every node using different protocols. The initial voltage is measured before the protocol is executed and the final voltage is measured after 50,000 samples are collected. We use the same brand new AA Alkaline batteries in all the experiments to get the same initial voltage, 3.138 v, and we find that different protocols result in different final voltages. TinyDB consumes much more energy than the other two. For example, the voltage drops 0.24 v at layer one nodes and 0.22 v at layer two nodes in TinyDB, while it drops only 0.20 and 0.18 v correspondingly in both Lazy and Alep. This is because TinyDB delivers much more messages than Lazy and Alep. As expected, layer one nodes consume more energy than layer two nodes because they need to forward messages for the latter. Motes in the same layer consume similar energy. The areas with different data dynamics consume almost the same amount of energy, because the number of messages does not differ too much; however, we still see that the area with lowest data dynamic does consume less energy than the Motes in other areas. Lazy and Alep have very close energy consumption, however lazy consumes less than Alep, especially for the layer one nodes. In summary, we find that the energy consumption mostly matches the result of the number of delivered messages very well. Alep and Lazy are really energy efficient, and they can extend the lifetime of WSNs considerably.

7.4. Comparison of samplings

The most intuitive way to show the quality of collected data is to compare them with the corresponding data generated by SDGen, acting as real data. In this experiment, we sample every 4 readings from all data generated by SDGen in TinyDB and Lazy, and the sampling rate is changing in Alep because of adaptation. In Fig. 12, three types of data with variant data dynamics are shown with the *x*-axis denoting the sampling serial number and *y*-axis depicting the value of the corresponding sampling. The figures should be read column by column. When data dynamic is high as show in Fig. 12(a), all three protocols perform similar at the beginning, but Alep catches the trend of the sampling better than the other two protocols because it adapts to a high sampling rate. Thus more details are observed, for example, between sampling serial number 21 and 40, Alep catches the dynamics of the data very well, while both Lazy and TinyDB miss the details. Furthermore, Lazy filters more details than TinyDB, but we argue that the degrading of data quality will not be significant because it is bounded by the consistency endurance range of Lazy. In Fig. 12(b), data dynamics is not as high as that in (a). All three protocols collect very similar data, which captures the dynamics of real data. In this case, although Alep seems not perform better than TinyDB, it delivers much less messages than TinyDB. Moreover, we can adjust the parameter in Alep so that the adaptation can be sharper, e.g., we can set small window size and decrease the threshold of adaptation. The data collected from the low dynamic area is shown in Fig. 12(c), where we observe that all three protocols can capture the trends of the sampling very well. However, Alep catches the trend a little later than TinyDB and Lazy, because Alep decreases the sampling rate when the data dynamics is low. As a result, some immediate changes are ignored and postponed to next time when the data is sampled. This will not be a problem if we set the minimum data sampling rate close to the regular data sampling rate in TinyDB. Comparing Lazy with TinyDB, we find that they are mostly comparable, especially when the data dynamic is not very

Please cite this article in press as: K. Sha, W. Shi, Consistency-driven data quality management of networked sensor systems, J. Parallel Distrib. Comput. (2008), doi:10.1016/j.jpdc.2008.06.004



(a) High dynamics. (b) Regular dynamics. (c) Low dynamics.

Fig. 12. Comparison of received data at the sink with real data. Note that the SQUARE shape represents the values of real data and the DIAMOND shape shows the value of collected data.

high, so we can save a lot of energy by using Lazy, but when the data dynamics is high, only Alep can catch the trends and the details of the monitoring parameter.

7.5. Comparison of data inconsistency factor

We have seen the advantage of the Alep protocol directly and intuitively from the sampled data readings. We also want to compare these protocols quantitatively in terms of data inconsistency factor as shown in Fig. 13, which depicts the results for three layer one nodes only. From the figure, we can see that the result is close to that in simulation. When data dynamics is high, Alep can reduce data inconsistency factor, but while data dynamics is low, Alep increases data inconsistency factor a little. However, we argue that we can control the increasing of data inconsistency factor by limiting the minimum sampling rate during adaptation, from which we still can get the advantage of reducing a large amount of messages. Furthermore, data inconsistency factor will be much less if we think the data within tolerance range as consistent in the calculation of data inconsistency factor. Compared with Lazy, Alep has larger data inconsistency factor when the data dynamics is very low, because Alep neglects a lot of details by decreasing the sampling rate.

7.6. Discussions

In summary, from both simulation and prototype evaluation, we find that Alep can improve data quality when the data dynamic is high and reduce the number of delivered messages a lot in almost all cases. We also find that the minimum sampling rate is very important to control the quality of collected data, which decreases very fast if the sampling rate is too low to get the details of data changing. Releasing the temporal consistency requirements is helpful to reduce the number of delivered messages so that to save energy, as well as to decrease the probability of communication collision and increase the possibility of data aggregation, so we expect less package loss rate in Alep and Lazy than that in TinyDB. However, it may also make the messages piled and exceed the limitation of the total memory of MICA2. Thus, we should set a low bound for temporal consistency requirements.

We also realized that trace-based approaches only work in the simulation but fail in the prototype test due to the memory constraints. Compared with TinyDB, Lazy and Alep may have more strict requirements on the correct delivery of the messages because the effect of data loss will be more severer than that in TinyDB. Retransmission strategy may be applied here. Furthermore, TinyDB's optimization when message queue is full can also be applied in Alep and Lazy, however, we argue that the probability that message queue is full in Lazy and Alep is much less than that in TinyDB because the number of total delivered messages is significantly reduced. Finally, the performance of Alep can be improved by tuning the parameters, which deserves further investigation.

8. Related work

Next, we compare our work with previous efforts in terms of energy efficiency design, data consistency, adaptive design, and data management respectively.

Energy efficiency is always one of the major WSN design goals. Thus, energy efficient protocols have been extensively explored. Previous work expects to achieve the goal of energy efficiency by designing energy efficient routing protocols [8,26,27], energy efficient MAC protocols [21,39], energy efficient clustering [40], and other energy efficient approaches. These approaches focus on finding energy efficient paths, designing better turn on/off schedules, forming energy efficient clusters, and so on. And none of them has examined the energy efficiency from the view of data quality. Recently, Tang and Xu propose an approach to extend network lifetime by differentiating the precision of data collected from different sensor nodes [41], which shares the similar idea as ours; however, they do not define consistency models, which can be used as a general metric to evaluate data quality. A lot of consistency models have been proposed in computer architecture, distributed systems, and database [20,23,37];

Please cite this article in press as: K. Sha, W. Shi, Consistency-driven data quality management of networked sensor systems, J. Parallel Distrib. Comput. (2008), doi:10.1016/j.jpdc.2008.06.004

	Mote 1			Mote 2			Mote 3		
	TinyDB	Lazy	Alep	TinyDB	Lazy	Alep	TinyDB	Lazy	Alep
High dynamics	772	748	678	495	783	409	517	515	629
Mixed dynamics	558	498	610	205	309	341	532	658	624
Low dynamics	116	346	354	87	187	229	136	234	546

Fig. 13. Comparison of data inconsistency factor.

however, these models are usually not applicable in WSNs. Ramamritham et al. propose an idea to maintain the coherency of dynamics data in the dynamics web monitoring application [28]. They model the dynamics of the data items. Our model for data consistency is more general than theirs and applied in different fields. Lu et al. propose a spatiotemporal query service in [14] to enable mobile users to periodically gather information and meet the spatiotemporal performance constraints, but they propose neither data consistency models, nor adaptive protocols. Adaptive approach is always attractive in system design. Several adaptive protocols which adapt cluster formation and duty cycle designing are proposed in literature [3,9]. Adaptive sampling rate has also been proposed from researchers of database field, sharing the same goal of our Alep protocol. Jain and Chang propose an adaptive sampling for WSNs [11]. They employ a Kalman-Filter (KF) based estimation technique and the sensor uses the KF estimation error to adapt the sampling rate. Marbini and Sacks [17] propose a similar approach to adapt the sampling rate as ours; however they do not model the data dynamics and require an internal model, which is usually difficult to find, to compare the sampled data. TinyDB [15] adapts the sampling rate based on current network load conditions, but not based on the data dynamics in the data field. Tang and Xu also propose to adapt sampling rate to extending the network lifetime in [41] and Rangwala et al. use rate adaptation mechanism [24] to achieve fair rate control, but the uniqueness of our work is proposing a framework to control data quality and consistency models as an metric to control the adaptation. Data management has been extensively explored in previous research. A recent work [13] by Li et al. uses a feedback-driven model-based-prediction approach to manage sensed data, which shares similar idea with us. Their work makes a tradeoff between the storage cost and the communication cost; however, they do not provide a general consistency model like us to evaluate data quality, nor do they dynamically adapt the sampling rate to improve data quality. Filters are also used to manage data by reducing the size of the data stream. Work by Olston et al. uses an adaptive filter to reduce the load of continuous query. Their work focuses on the adaptive bound width adjustment to the filter so that their results are helpful to analyze our lazy approach. Sharaf et al. study the tradeoff between the energy efficiency and quality of data aggregation in [29]. They impose a hierarchy of output filters to reduce the size of the transmitted data. Data prioritization in TinyDB [15] chooses the most important samples to deliver according to the user-specified prioritization function.

9. Conclusions

In this paper, we propose a consistency-driven data quality management framework *Orchis* and depict its two important components: *consistency models* and *an adaptive protocol*. To the best of our knowledge, we are the first that formally define a set of consistency models for WSNs. We also design and implement an adaptive, lazy, energy efficient data collection protocol to improve data quality and save energy. The comprehensive evaluation using both TOSSIM-based simulation and a prototype implementation shows that the proposed Alep protocol indeed reduces the number of delivered messages, improves the quality of the collected data and saves energy.

Appendix. Supplementary data

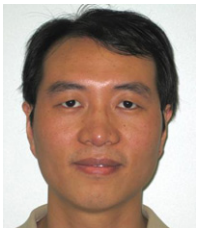
Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.jpdc.2008.06.004.

References

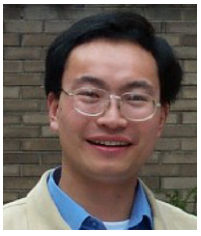
[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks, IEEE Communications Magazine 40 (8) (2002) 102–114.
[2] M. Batalin, et al. Call and responses: Experiments in sampling the environment, in: Proc. of ACM SenSys 2004, 2004.
[3] A. Cerpa, D. Estrin, ASCENT: Adaptive self-configuring sensor network topologies, in: Proc. of IEEE International Conference on Computer Communications (INFOCOM'02), 2002.
[4] Crossbow technology inc. URL http://www.xbow.com.
[5] D. Estrin, et al. Next century challenges: Scalable coordination in sensor networks, in: Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99), 1999.
[6] D. Estrin, D. Culler, K. Pister, G. Sukhatme, Connecting the physical world with pervasive networks, IEEE Pervasive Computing 1 (1) (2002) 59–69.
[7] O. Gnawali, et al. The tenet architecture for tiered sensor networks, in: Proc. of ACM SenSys 2006, 2006.
[8] B. Hamdaoui, P. Ramanathan, Energy-efficient and MAC-aware routing for data, in: Aggregation in Sensor Networks, IEEE Press, 2006.
[9] T. He, B. Blum, J.A. Stankovic, T.F. Abdelzaher, Aida: Adaptive application independent data aggregation in wireless sensor networks, ACM Transactions on Embeded Computing Systems 40 (8) (2002) 102–114.
[10] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister, System, architecture directions for networked sensors, in: Proceedings the 9th ASPLOS'00, 2000, pp. 93–104.
[11] A. Jain, E. Chang, Adaptive sampling for sensor networks, in: Proc. of the 1st International Workshop on Data Management for Sensor Networks: in Conjunction with VLDB 2004, 2004.
[12] P. Levis, N. Lee, M. Welsh, D. Culler, Tossim: Accurate and scalable simulation of entire tinyos applications, in: Proc. of ACM SenSys 2003, 2003.
[13] M. Li, D. Ganesan, P. Shenoy, PRESTO: Feedback-driven data management in sensor networks, in: Proc. of the NSDI'06, 2006.
[14] C. Lu, et al. A spatiotemporal query service for mobile users in sensor networks, in: Proceedings of the 25th International Conference on Distributed Computing Systems (ICDCS'05), 2005.
[15] S. Madden, et al. Tinydb: An acqusitional query processing system for sensor networks, ACM Transactions on Database Systems 30 (1).
[16] G. Mainland, D. Parkes, M. Welsh, Decentralized, adaptive resource allocation for sensor networks, in: Proc. of the NSDI'05, 2005.
[17] A. Marbini, L. Sacks, Adaptive sampling mechanisms in sensor networks, in: London Communications Symposium, 2003.
[18] R. Min, A. Chandrakasan, Mobicom poster: Top five myths about the energy consumption of wireless communication, Mobile Computing and Communications Review 7 (1) (2003) 65–67.
[19] S. Nath, J. Liu, F. Zhao, Sensormap for wide-area sensor webs, Computer 40 (7) (2007) 106–109.
[20] L.L. Peterson, B. Davie, Computer Networks: A Systems Approach, Morgan Kaufmann, 2003.
[21] J. Polastre, J. Hill, D. Culler, Veratile low power media access for wireless sensor networks, in: Proc. of ACM SenSys 2004, 2004.
[22] G. Pottie, W. Kaiser, Wireless integrated network sensors, Communications of the ACM 43 (5) (2000) 51–58.
[23] R. Ramakrishnan, Database Management Systems, WCB/McGraw-Hill, 1998.
[24] S. Rangwala, et al. Interference-aware fair rate control in wireless sensor networks, in: Proc. of ACM SIGCOMM'06, 2006.
[25] C. Sadler, P. Zhang, M. Martonosi, S. Lyon, Hardware design experiences in zebranet, in: Proc. of ACM SenSys 2004, 2004.
[26] K. Seada, M. Zuniga, A. Helmy, B. Krishnamachari, Energy-efficient forwarding strategies for geographic routing in lossy wireless sensor networks, in: Proc. of ACM SenSys 2004, 2004.
[27] K. Sha, J. Du, W. Shi, Wear: A balanced, fault-tolerant, energy-efficient routing protocol for wireless sensor networks, International Journal of Sensor Networks 1 (2).
[28] S. Shah, S. Dharmarajan, K. Ramamritham, An efficient and resilient approach to filtering and disseminating streaming data, in: Proc. of 29th International Conference on Very Large Data Bases, 2003.
[29] M. Sharaf, J. Beaver, A. Labrinidis, P. Chrysanthis, Balancing energy efficiency and quality of aggregate data in sensor networks, The VLDB Journal 13 (4) (2004) 384–403.

Please cite this article in press as: K. Sha, W. Shi, Consistency-driven data quality management of networked sensor systems, J. Parallel Distrib. Comput. (2008), doi:10.1016/j.jpdc.2008.06.004

[30] K. Sha, W. Shi, Modeling the lifetime of wireless sensor networks, Sensor Letters 3 (2) (2005) 126–135.
[31] K. Sha, W. Shi, Modeling data consistency in wireless sensor networks, Tech. Rep. MIST-TR-2006-013, Wayne State University (Oct. 2006).
[32] K. Sha, W. Shi, On the effects of consistency in data operations in wireless sensor networks, in: Proceedings of IEEE 12th International Conference on Parallel and Distributed Systems, 2006.
[33] V. Shnayder, et al. Simulating the power consumption of large-scale sensor network applications, in: Proc. of ACM SenSys 2004, 2004.
[34] N. Shrivastava, et al. Target tracking with binary proximity sensors: Fundamental limits, minimal descriptions, and algorithms, in: Proc. of ACM SenSys 2006, 2006.
[35] R. Szewczyk, et al., Habitat monitoring with sensor networks, Communications of the ACM 47 (6) (2004) 34–40.
[36] G. Simon, A. Ledeczi, M. Maroti, Sensor network-based countersniper system, in: Proc. of ACM SenSys 2004, 2004.
[37] A.S. Tanenbaum, M. van Steen, Distributed Systems: Principles and Paradigms, Prentice-Hall, 2002.
[38] The nyquist-shannon sampling theorem. URL http://ptolemy.eecs.berkeley.edu/eecs20/week13/nyquistShannon.html.
[39] W. Ye, J. Heidemann, D. Estrin, An energy-efficient mac protocol for wireless sensor networks, in: Proc. of IEEE International Conference on Computer Communications (INFOCOM'02), New York, NY, 2002.
[40] Q. Younis, S. Fahmy, Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach, in: Proc. of IEEE International Conference on Computer Communications (INFOCOM'04), 2004.
[41] X. Tang, J. Xu, Extending network lifetime for precision-constrained data aggregation in wireless sensor networks, in: Proc. of IEEE International Conference on Computer Communications (INFOCOM'06), 2006.
[42] N. Xu, et al. A wireless sensor network for structural monitoring, in: Proc. of ACM SenSys 2004, 2004.



Kewei Sha received B.S degree from East China University of Science Technology in 2001, and M.S. degree from Wayne State University in 2006. He is currently a Ph.D. candidate at Wayne State University. His current research focuses on distributed systems, wireless sensor networks, and vehicular networks, especially in data quality management as well as system design and modeling.



Weisong Shi is an Associate Professor of Computer Science at Wayne State University. He received his B.S. from Xidian University in 1995, and Ph.D. degree from the Chinese Academy of Sciences in 2000, both in Computer Engineering. His current research focuses on mobile computing, distributed systems and high performance computing. Dr. Shi has published more than 80 peer-reviewed journal and conference papers in these areas. He is the author of the book "Performance Optimization of Software Distributed Shared Memory Systems" (High Education Press, 2004). He has also served on technical program committees of several international conferences, including WWW, ICPP, MASS. He is a recipient of Microsoft Fellowship in 1999, the President outstanding award of the Chinese Academy of Sciences in 2000, one of 100 outstanding Ph.D. dissertations (China) in 2002, "Faculty Research Award" of Wayne State University in 2004 and 2005, the "Best Paper Award" of ICWE'04 and IPDPS'05. He is a recipient of the NSF CAREER award.

Please cite this article in press as: K. Sha, W. Shi, Consistency-driven data quality management of networked sensor systems, J. Parallel Distrib. Comput. (2008), doi:10.1016/j.jpdc.2008.06.004