Modeling Data Consistency in Wireless Sensor Networks

Kewei Sha and Weisong Shi
Department of Computer Science
Wayne State University
{kewei, weisong}@wayne.edu

***Abstract*— With the rapid growth of wireless sensor systems deployment, data quality has become a critical issue to the success of these applications. In this paper, we first raise the data quality problem in WSN. Then, we propose a novel metric, *data consistency*, to evaluate the data quality. Our consistency models consider three perspectives of consistency: *temporal, numerical,* and *frequency*, covering both individual data and data streams. Moreover, several system protocol design issues are identified to support data consistency. Finally, we propose a set of APIs to facilitate managing data consistency.**

I. Introduction

A variety of applications such as environmental and habitat monitoring [1], [6] and emergency care [2] have been launched showing the promise of wireless sensor networks (WSN). However, their success is nonetheless determined by whether the sensor networks can provide a high quality stream of data over a long period. The inherent feature of unattended and untethered deployment of networked sensors, however, imposes challenges to the underlying systems. These challenges are further complicated by the fact that sensor systems are usually seriously energy constrained. Most previous efforts focus on devising techniques to save the sensor node energy and thus extend the lifetime of the whole WSN. However, with more deployment of real sensor systems, in which the main function is to collect interesting data at the sink, data quality has been becoming a critical issue in the design of sensor systems. We argue that the quality of data should be used as a basic performance evaluation metric, as energy efficiency does, to evaluate protocols, and envision that the quality of data should reflect the timeliness and accuracy of collected individual data, and be able to control the frequency of dramatic data changes and abnormal readings of data streams. Unfortunately, we have not seen any metric being proposed to evaluate the data quality in public literature. This paper takes an initial step.

On the other hand, data consistency models have been extensively studied in previous research. However, due to the distinct features of WSN, such as unreliable wireless communication and limited bandwidth, storage and power supply, previous consistency models are not applicable in WSN, and novel consistency models in WSN should be redefined to evaluate the quality of collected data, which is the goal of this paper.

In order to model data consistency, we first abstract the consistency requirements from temporal and numerical perspectives. Based on the observation that data streams are more meaningful than individual data from the perspective of applications, we extend the consistency requirements by including the requirement to control the frequency of dramatic data changes and abnormal readings of data streams, i.e., frequency consistency. Thus, data consistency is modeled as an application-specific concept for both individual data and data streams. Keep this in mind, we argue that the design of WSN system protocols should be revisited to cater to the consistency goal. Four open issues including an adaptive protocol, a data management algorithm, consistency-driven cross-layer protocols and consistency related APIs, are identified to support data consistency. Furthermore, we tackle the last open issue, by designing a set of APIs to check the consistency status and manage the consistency based on application scientists' decision. The contributions of our paper are summarized as follows. First, we raise the problem of data consistency to the society and analyze the consistency requirements in WSN. Second, we propose a series of data consistency models that can be used as metrics to evaluate the data consistency in different applications. Finally, a set of APIs are designed to facilitate the data consistency management.

The rest of the paper is organized as follows. We analyze consistency requirements in Section II, followed by the description of consistency models. In Section IV, we identify four open issues. APIs that facilitate managing data consistency are proposed in Section V. Finally we summarize in Section VI.

II. Consistency Requirements Analysis

Data consistency is an important problem in computer architecture, distributed systems, database, and collaborative systems [3], [4], [7]. A lot of consistency models have been proposed in these fields. However, these models are usually not applicable in WSN because of the specific characteristics of WSN. Thus, consistency models, the key to evaluate the quality of the collected data, should be remodelled for WSN applications. In this section, we first analyze the difference between WSNs and traditional distributed systems in terms of consistency; then, we abstract the data consistency requirements in WSN.

Although a WSN is an instance of a distributed system, there are several significant differences between WSNs and

traditional distributed systems. First, WSNs are resource constraint systems. Due to the constraints of the memory and the large amount of the data, the data are usually not stored in sensors for a long period, but they will form data streams to be delivered to the sinks or base stations. As a result, data consistency in WSN will not focus on the read/write consistency among multiple data replicas as in traditional distributed systems; instead, data consistency in WSN 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 in different time. Second, WSN applications may have more interests in a set of data which can depict the trends of the monitoring parameter or report an event by combining these data together. Thus, consistency models for data streams are more important than those for individual data. In this paper, consistency models for both types of data are modeled. Third, compared with traditional distributed systems, the unreliable wireless communication is common, rather than abnormal, in WSN. Although retransmission is a strategy to rectify the effect caused by the unreliable wireless communication, there is no simple technique that can guarantee the successful delivery of a message. Thus, in the consistency model, the data loss due to wireless communication should also be considered. Furthermore, in previous definition of the data consistency [7], the effect of channel noises and intended attacks are neglected. We argue that attacks are normal nowadays, and the security technologies should be integrated in the system design to prevent attacks.

In summary, we conclude that consistency models in traditional distributed systems that basically discuss the read/write consistency among different replicas are not sufficient to be applied in WSNs. Given the specific features of resource constraints and unreliable communication, consistency models in WSN should be remodelled.

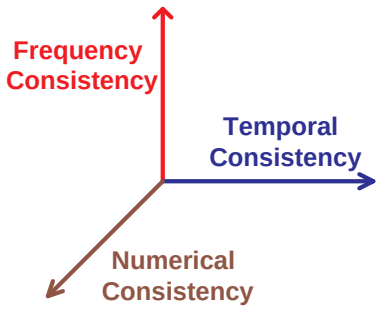

Fig. 1. A three-dimension view of consistency requirements.

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 Figure 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 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 application may have various consistency requirements for them respectively. For example, in a patient monitoring system, emergency conditions of a patient should be reported to the control panel or caregivers as quick as possible. Otherwise, the patient may be in a dangerous condition. Thus, most 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 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.

III. Consistency Models

We are in a position to propose data consistency models to evaluate the different data quality. In this section, we first define a general data format used in our models. Then, we

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 [5] is used to fill these data at the sink.

A. 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 is derived from the numerical values of the data, we define the data format used in consistency models as follows,

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

where p_i denotes that the data is from the i^{th} 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 i^{th} 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 consistent requirements and the value of T_{stmap}, Seq_Ran, and Val are set locally by the i^{th} node.

As we analyzed in the above subsection, different applications can 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 minutes 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 $Rang$ 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 predistributed by other protocols (Section IV). 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. Next, we formally model data consistency by using these abstracted semantics.

B. Consistency Models for Individual Data

For each individual data, we model three types of data consistency, *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 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 below,

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

where, $InSemantics$ judges the numerical consistency by checking whether the new arriving value and the cashed 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 will reduce the source usage while providing 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 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,

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

Similar to functions in the hop-consistency, $InSemantics$ checks the numerical consistency in terms of the consistency semantics. 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. 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 applcaitions.

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

Compared with the single-path consistency, the multiple-path consistency has one more requirement about the number of copies for the same data, denoted by $Count(P_i, Seq\_Ran)$.

C. Consistency Models for Data Streams

In WSN, data are usually collected in the format of data streams. Individual data may not have significant meaning, while they are useful when the set of the data are considered together. Thus, we argue that consistency models for a set of data, data streams, denoted as $D = \{d_1, d_2, ..., d_n\}$, are the same important, if not more important, as the models for individual data. For data streams, we propose six types of consistency models 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. 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. To satisfy the strict consistency, 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 in terms of consistency semantics is maintained, 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

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

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 WSN. 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

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

where $InSemantics$ checks the numerical consistency as before. All the received data are temporal consistent and the number of total received data is large 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 high 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, 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

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

The partial consistency is similar to the α-loss consistency except that the temporal consistency requirement is released. This consistency model is useful in applications where aggregation applies, which have numerical consistency requirements but low temporal consistency requirements.

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

$$TrendConsist = (TrendSatisfy(D)) - - - (7)$$

where $TrandSatisfy$ 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 II) 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 pre-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,

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

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. 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,

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

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 [5]. The observation of violation of this consistency may also result in a request of application scientists involvement.

In summary, we propose a set of basic but powerful consistency models for data quality measurement in WSN 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 or 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 the above basic proposed consistency models to form complicated models. 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 are also composable to set all numerical, temporal and frequency consistency requirements. Furthermore, various applications should make a trade off between the energy efficiency and data consistency based on their energy budget, which remains an open problem in the community.

IV. Open Issues

We argue that WSN system protocols should be revisited to achieve the goal of consistency; moreover, these designs should keep the goal of energy efficiency in mind. In this section, we identify four open issues in WSN system design as follows.

- An *adaptive protocol* to improve the quality of collected data and take advantage of data consistency by considering *data dynamics*. We observe that data consistency and energy efficiency are closely related to data dynamics. Thus, models for data dynamics are required. A protocol that automatically adapts the data sampling rate according to the data dynamics in the data field is necessary to improve the quality of collected data and energy saving.
- An intelligent *data management algorithm*. To take advantage of data consistency, data management algorithms are needed to control the amount of traffic in an energy efficient way, e.g., we can drop data whose value is within the consistency semantics. And more intelligent algorithms are expected from other research field such as signal processing to filter unnecessary data.
- *Consistency-driven cross-layer protocols* to achieving data consistency. Diverse data consistency requirements and variant data traffic of nodes resulting from aforementioned adaptive protocols bring the new challenging for the node scheduling, MAC protocols, and routing protocols. Cross-layer protocols should be designed to filter unnecessary data, control traffic, and route packets while maintaining consistency. Moreover, adaptive resource allocation and topology control are also expected to be integrated into these protocols.
- A set of *APIs to manage data consistency*. Data consistency requirements should be distributed to the monitoring area when the sensor network is deployed. Later on, the consistency requirement should be updated according to the observed data consistency from recent collected data. APIs are essential to support the functionality of checking consistency status, and to distribute consistency requirements. These APIs have to provide interfaces for lower layer data collection protocols to efficiently transfer the data to the sink.

All four issues described above should be addressed to take full advantage of data consistency models. In this paper, we take an initial step to design a set of APIs to manage data consistency as described in detail in next Section, and leave the rest three as our future work.

V. APIs for Managing Data Consistency

There is a gap between the lower layer protocols designed to support the consistency goal and the higher layer consistency requirements from applications. It is critical to provide user-friend interfaces for application scientists to take advantage of these models. Our APIs are designed for the purpose of the data quality management, and differ from the APIs proposed in [8]. First, our APIs lie at the higher layer (for application scientists) than theirs (for system programmers). Second, the design goals are different too. We believe that our APIs can take advantage of theirs in the real implementation. To manage the consistency, the APIs must have the following functions, *checking the*

APIs	Function Descriptions
$CheckStatus(P, Set(N), T_{ran}, ConMode)$	Check the consistency of the data from the nodes in $Set(N)$ during the time period T_{ran} according to the data consistency requirements to that data.
$SetReq(P, Set(N), T_{req}, V_{req}, Range, Max\_OutRange, Max\_Diff, Max\_CHG, ConMode, CommPtn)$	Set the temporal or numerical consistency requirements for parameter P to the set of nodes $Set(N)$ using the specified communication pattern to distribute the requirements.
$UpdateReq(P, Set(N), \delta(T), \delta(V), Range, Max\_OutRange, Max\_Diff, Max\_CHG, ConMode, CommPtn)$	Update the temporal or numerical consistency requirements for parameter P at the set of nodes $Set(N)$, and use the specified communication pattern to distribute the updating.

TABLE I

APIS FOR DATA CONSISTENCY MANAGEMENT.

current consistency status ($CheckStatus$), *setting consistency requirements* for new parameters ($SetReq$), *updating consistency requirements* ($UpdateReq$), and getting support from lower layer protocols, as listed in Table I, where $CommPtn$ denotes the communication pattern to distribute the consistency requirements; $ConMode$ depicts the name of consistency model. Together with V_{req} , T_{req} and other parameters, $ConMode$ also specifies the consistency semantics; and $Set(n)$ depicts the set of destination nodes.

Several protocols and algorithms are needed to support the above proposed APIs. For example, consistency checking algorithms are needed when the $CheckStatus$ API is called. Various algorithms are needed to check consistency in different models. While in $SetReq$ and $UpdateReq$, different protocols are used depending on the size of $Set(n)$. If there is only one node in $Set(n)$, a point-to-point communication pattern is adopted to deliver the consistency requirements. When $Set(n)$ contains all the sensors in the field, broadcast is launched to distribute the requirements. If $Set(n)$ contains nodes located in one area, area multi-cast is used to disperse the requirements. Hence, various routing protocols are needed for different communication patterns.

The process of consistency management consists of three steps. First, the consistency requirements are distributed. Second, the consistency will be checked at the sink. Third, if the sink finds that current consistency cannot be satisfied because of 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.

Next, we give an example of how to use these APIs. In a habitat monitoring application, if an irregular animal movement is observed in some area, a request to monitoring the temperature of that area is issued, i.e., $SetReq(temp, Set(area), T_{req}, Val_{req}, 0, 0, 0, 0, 10\% - LossConsist, area - cast)$, 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 $area - cast$. T_{req}, Val_{req} and $10\% - LossConsist$ specify the consistency semantics. Four zeros denote no specific frequency consistency requirements. After the application scientist collects some 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 mode, $10\% - LossConsistency$. Based on the result of the call, the application scientist makes a decision to tune consistency. For example, if the application scientist thinks that the quality of the data is good enough, he will do nothing to change it; otherwise, he will update the new consistency requirements by calling $UpdateRqe(temp, Set(area), \delta(T)), \delta(Val), 0, 0, 0, 0, 5\% - LossConsist, area - cast)$. Then, when receiving the new update request, the node will update the consistency requirements locally. The whole process forms a close-loop feedback control. In this way, high quality data could be collected in an energy efficient way.

VI. SUMMARY

In this paper, we propose to use data consistency as a metric to evaluate the quality of data in WSN. Several formal consistency models are defined. We also identify four consistency related system design issues, and propose a set of APIs for consistency management. This is our initial step to investigating the data quality problem in WSN. The authors' hope is that this position paper will stimulate further work on the problem of data quality in sensor networks using data consistency models.

REFERENCES

[1] M. Batalin et al. Call and responses: Experiments in sampling the environment. *Proc. of ACM SenSys 2004*, Nov. 2004.

[2] T. Gao, D. Greenspan, and M. Welsh. Improving patient monitoring and tracking in emergency response. *Proc. of the International Conference on Information Communication Technologies in Health*, July 2005.

[3] L. L. Peterson and B. Davie. *Computer Networks: A Systems Approach*. Morgan Kaufmann, 2003.

[4] R. Ramakrishnan. *Database Management Systems*. WCB/McGraw-Hill, 1998.

[5] K. Sha and W. Shi. On the effects of consistency in data operations in wireless sensor networks. *Proceddings of the 2006 International Conference on Parallel and Distributed Systems (ICPADS'06)*, July 2006.

[6] R. Szewczyk et al. Habitat monitoring with sensor networks. *Communications of the ACM* 47(6):34–40, June 2004.

[7] A. Tanenbaum and M. Steen. *Distributed Systems: Principles and Paradigms*. Prentice-Hall, 2002.

[8] M. Welsh and G. Mainland. Programming sensor network using abstract regions. *Proceedings of the First USENIX/ACM Networked System Design and Implementation*, Mar. 2004.