Reliability-aware Sensor Node Clustering Schemefor Reliable IoT Data Services

Siwoo Byun

ABSTRACT--- Sensor data management is a major component of the Internet of Things environment. The huge volume of data produced and transmitted from sensing devices can provide a lot of useful information but is often considered the next big data for future businesses. New col-umn-wise compression technology is mounted to the large data server because of its superior space efficiency. First, we bring forth a short overview through providing an analysis on IoT sensor net-works. Second, we discuss concept of efficient sensor data management to cope with unreli-able sensor nodes and communication failures. In this study, reliability-based dynamic clus-tering scheme and shadow copy management scheme are proposed to improve service relia-bility and performance for IoT data collection and motoring servers. In addition, column-wise compressed mirroring and modified synchronizing skills are used for shadow copy management to improve storage efficiency.The proposed dynamic clustering and shadow data management skills minimizes battery consumption and communication failures of unstable sensor nodes. Consequently, unneces-sary communication costs and battery energy can be reduced, and overall energy balance can be maintained in the sensor data network. We conclude that proposed cluster control scheme outperforms the previous storage control by performance and reliability factor.

Keywords—IoT, shadow copy, clustering, data compression, sensor device, sensor database.

1. INTRODUCTION

Recently, *Internet of Things* (IoT) sensor network has received significant attention in smart system areas. Small IoT devices can be designed with on-board calculations, wireless communications and sensor detection abilities. Recent IoT applications include energy usage monitoring and planning ener-gy conservation in buildings, military and private surveillance, natural habitats monitoring for under-standing environmental dynamics, and collecting data for learning environments. Basic sensor nodes contain a small battery pack, an 8-bit RISC processor, a low baud-rate radio, several megabytes, sev-eral gigabytes Flash Memory, and MEMS sensors for detecting temperature, ambient light, and vibra-tion(Fig.1)[1,2].

IoT sensor network is different from the traditional wired and wireless networks since the applica-tion described above automatically operates unattended. Since sensor device uses battery and wireless channel, the management efficiency of limited battery and substantial transmission energy may be a key design issue as compared to the traditional unconstrained system. This means that reliability of sensor nodes and wireless communication is also an important design issue.

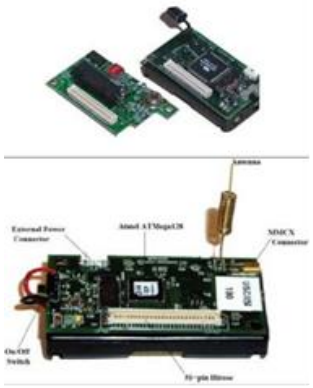


Figure 1. Example of Tiny Sensor Node

Sensor networks are *data-centric* since they are designed in terms of measured value rather than identified value such as IP address in conventional network communication. That is, only data is im-portant in sensor data networks. From this architectural point of view describe above, sensor network is treated as a huge global database called *sensor database*[3]. That is, sensor network is considered to be a data-driven routing system in that routing is a bottom-up view while general database is a top-down view. As well as general databases, sensor database should provide a high level of robustness to node failures or network failures.

Transactions in the sensor database server must always be consistent and reliable. Reliable transac-tion management ensures that the database returns to a consistent or correct state after a hardware or software failure. A database transaction is a set of operations. The transaction may be an entire pro-gram or single read/write operation[4].

2. BACKGROUND

In IoT environment, small sensor nodes have frequently disconnected and narrow wireless channel, limited battery power, and limited storage capacity. Therefore, IoT applications built on this limited conditions will inevitably suffer from poor data services. The surest way to minimize these unreliable data services is to replicate critical data to multiple sites.

Examples of the advanced data replication in the IoT are *fresh food logistics* ("cold chain") where maintaining freshness is critical, or a life-rescue network in disaster areas. In these sensor network environment, the recent blockchain technology is also utilized for reliable data sharing. Especially, critical sensor data such as the location of vehicles and food temperature are collected continuously and used for big data analysis [5,6].

Revised Manuscript Received on May15, 2019.

Siwoo Byun, 708-113 Dept. Software, Anyang Univ., Anyang 5-dong, Manan-gu, Anyang, Korea(swbyun@anayng.ac.kr)

IoT sensor database uses column-oriented storages[7,8] where sensor data can be compressed and stored in a lower storage devices for processing large amounts of data at high speed, and the in-put/output operations are performed at high speed in memory after restoring compressed data as need-ed. To effectively and reliably support this, advanced system should avoid failure of the data access rather than heavy data recovery after failures.

In particular, the column-oriented database uses big data and its contents are intended for semantic analysis such as business strategy and smart marketing, so it should perform urgent tasks such as swift decision making. As a result, time overhead of re-analysis after data recovery and reloading is too large. Therefore, minimal data replication is critical to avoid data failures and unnecessary recovery in advance. That is, the recovery of column-oriented big sensor data is complex and time-consuming, and therefore additional methods of managing at least one more copy is needed to increase the reliability.

Replication control (RC) scheme typically targets both consistency and high data availability.While traditional RC schemes demonstrate reasonable costs and superior performance in a typical environment, they are insufficient to be utilized in a fault-prone sensor network environment. In this paper, we propose a new clustering models and replication control techniques to improve both reliabil-ity and performance in fault-prone IoT sensor networks.

3. RELIABLE DATA CONTROL FOR FAULT-PRONE IOT NETWORKS

Essentially, database systems store various data on a storage device before performing search and update operations. Thus, supporting reliable data services requires proactive failure prevention to eliminate costly post-recovery. In general, the data volume generated from the sensors is very large and the IoT query is unreliable due to the fault-prone characteristics of sensor nodes. Therefore, side effects caused by loss of sensor data and unstable values lead to the inaccurate results of group func-tions such as overall average.

To reduce this unreliability in sensor big database, some important data should be duplicated in ad-vance to prevent data service failure. However, it is inefficient to store measured data continuously in the long distant collection server to cope with future sensor queries. Note that continuous sensor transmission could contribute to severe battery consumption and network channel congestion.

In this study, instead of costly data replication in long distant collection server, we exploit a nearby sensor node and two cluster group according to data stability of each nodes. Thus, long-distance com-munication costs and power consumption can be reduced. For example, if a certain sensor node is expected to be disconnected (due to 10% of battery power, or 10% of wireless signal, or 5% of storage space, etc.), the sensor node should send its last data to one of the other stable nodes in reliable cluster group.

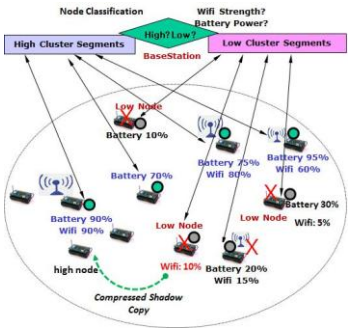
These replication process can lead to more efficient query planning in the future. For example, if user issues a true/false query such as "Is there any sensor node in which inner temperature is greater than 40 degrees?", then the query manager could search only the stable nodes inside the stable cluster. That is, if the search results is true, then the query manager doesn't have to search the unstable nodes inside the unstable cluster. This sequential query execution reduces the amount of running queries, overall communication costs and power consumption, while maintaining battery energy balance and providing fast query results.

In this study, to improve the reliability of the IoT sensor database, we propose RaClus-RC (Relia-bility-aware Clustering) for replication control. At first, RAID3,5 was also considered in the design, but our scheme is based on RAID1 (Mirroring) to reduce excessive message transmission across mul-tiple sensor nodes.

For stable mirroring, reliable sensor nodes with high battery power and strong wireless signals are selected as nearby sensor nodes to reduce communication burden. In addition, column-based compres-sion of the sensor data contribute to reduce overall network overhead, transmission power consump-tion of the sending node, receiving power consumption and storage consumption of the receiving node.

The compression method used for our scheme is LZO [9]. LZO is easy to modify because the source is very efficient and is published as the GPL. Our duplication scheme manages both com-pressed version in *shadow node* and non-compressed version in original node, whereas general mirror-ing physically maintains two same version copies. To achieve this availability goal, we propose a compressed dynamic clustering scheme that exploits *high-low classification* and *shadow copy replica-tion.* We classify the sensor nodes into two categories depending on their reliability status: *high-nodes* that are stable or *low-nodes* that are not stable. Each sensor node keeps update-timestamp for the high-low classification.

We also classify clustering area into two categories: high-cluster and low-cluster. The high cluster consists of multiple high segments which contain only high nodes, while the low cluster consists of multiple low segments which contain only low nodes. Compared to the low node, the high node has far more battery power and Wi-Fi signal strength. Although the criteria of the high and low clusters is determined by various experimental setups, we assume the default value of the high-low criteria, 1:3, meaning that the high node, whose average reliability is two times of the low node.

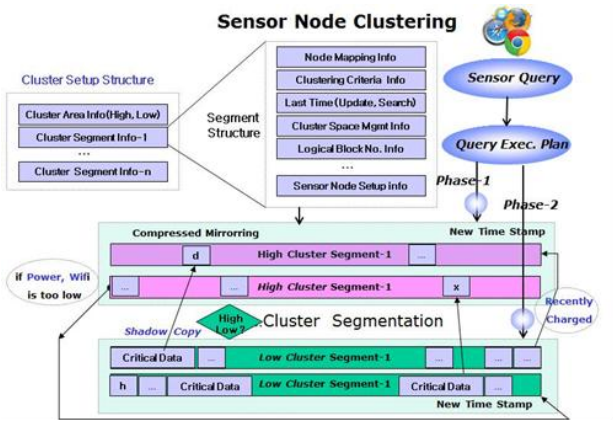


[Fig. 2] Concept of Shadow Replication and Dynamic Clustering Model

The main overhead of unreliable low nodes is two. One is message retransmission or transmission delay due to unreliable Wi-Fi signal. The other one is shutdown or sleeping due to battery failure. This unreliability could lead to severe overhead of IoT sensor query, since this low node basically incurs slow query and lost query during query processes. This node separation technique contributes to lessen query overhead by minimizing access to the low nodes. Note that high nodes have a high possibility of being accessed without failures and then provide quick query execution.

This increased efficiency and availability is the reason why we classify the sensor nodes into high or low nodes. If we collect active nodes in the same high cluster, we can efficiently handle the sensor query in a single action without accessing low nodes. In high-low clusters, a segment has multiple slots of sensor nodes. In RaClus-RC, the shadow copy of low sensor nodes reside in high clusters after data compression.

If a new node is inserted, the node is determined as a low node at first. If a low node show suffi-cient reliability after predefined cycle, the node is determined to be a high node. Then, RaClus-RC moves the node from the low cluster to the high cluster. If a certain high node is unreliable due to weak battery, RaClus-RC determines the node as a low node. Then, RaClus-RC removes the node from the high cluster and inserts it into the low cluster.



[Fig. 3] Structure of Reliability-aware Dynamic Clustering

For transactions to be carried out using shadow copy techniques of high cluster nodes,

SOP_WRITE, SOP_READ and SOP_RECOVERY operations are required as follows:

ShadowOP:

SOP_WRITE(ShadowSn#, *x*, *val*): writes a value of *val* to data entry *x* in ShadowSn# node.

① Sensor node stores the value of *val* for data item *x* in the transfer buffer, and thensend the latest measured value of *x* with status information.

② Shadow sensor nodes receive the transmitted information and then records thestatus information and latest measured value of the sending nodes.

Shadow sensor nodes commits the write operation and then sends *Fin_ack* message to the sending node.

* *SOP_READ*(Sn#, *x*): reads value of data entry *x* in Sn# node.

① If Sn# is unavailable or dead, read *id* of shadow node specified in the base station.

② Reads the last value of *x* from the shadow node.

③ The shadow sensor node send status information with the last *x* value of the dead sensor node.

* *SOP_RECOVERY* (ShadowSn#): If the sensor node is recovered, send the status of the sen-sor node to the shadow node and the base station.

① If the sensor node is recovered after recharging battery or detecting sufficient

Wi-Fi signal, store the last measured value of data item *x* into the transfer buffer.

② Send the latest measured value of data item *x* with status information to theshadow node and the base station.

③ The shadow node and the base station receive the recovery information andthen sends *Fin_ack* message to the recovered sensor node.

For reliable synchronization between unstable(low) node and shadow node, modified two-phase lock control is performed. In order to control lock conflicts, we make use of three lock modes: (1)Read,

(2)Reserve, and (3)Write. These three lock modes will be referred hereafter as *ReadLock*, *RsvLock*, and *WriteLock*. The compatibility table related to the three modes is described in Table 1.

[Table 1] Three lock comparisons in modified 2PL

Holder / Requester	RsvLock	WriteLock	ReadLock
RsvLock	N	N	N
WriteLock	N	N	N
ReadLock	N	N	Y

In case of the write operation for synchronization(shadow data copy), the sensor nodes should get RsvLock before they can be applied safely. The actual write operation is slow process via unstable wireless network. In order to support perfect termination of slow transactions, sensor nodes related to the write operation should be confirmed to update before the physical write operations. This can be achieved by setting RsvLock for later enforcement of the physical write in the sensor nodes. After confirming RsvLock, the RsvLock is escalated to WriteLock and then the physical updates are en-forced. The lock control algorithm for sensor nodes is described as follows.

Process:: Lock Control for Shadow Copy Synchronization {
IF *transaction.type==READ*:
if ((lock mode is compatible)∨(no lock held))
then
Lock:Set(Read-Lock, *transaction.data*);
Call_DM

(*DM_READ*);
return *READ_ACK*with data sent by Data Manger;
else
insert *transaction* into lock-queue and wait;
end_if;
IF *transaction.type==RSV***:**
if ((lock mode is compatible)∨(no lock held))
then
Lock:Set (RSV-Lock, *transaction.data*);
return GET_*RSV_ACK***;**
else
insert *transaction* into lock-queue and wait;
end_if;
IF *transaction.type==WRITE***:**
if ((lock mode is compatible)∨(no lock held))
then
Lock:Set (Write-Lock, *transaction.data*);
Call_DM(*DM_WRITE*);
return *WRITE_ACK***;**
else
insert *transaction* into lock-queue and wait;
end_if;
}
Legend-
Call_DM(o): Send o to DM(Data Manager);
Release_Lock(*d*): unlock*d*;
Lock:Set(*l*, *d*): set*l*lock on data*d*;

4. CONCLUSION

In this study, the IoT network technology environment and IoT sensor database technology were analyzed. It also discussed efficient data management and storage technologies to cope with sensor nodes and communication failures. For data collection and motoring in sensor database environment, a new reliability-based dynamic clustering scheme and compressed shadow replication scheme were proposed to improve the data service reliability and performance.

In addition, a modified synchronizing skill using data mirroring was applied to general distributed storage to store compressed shadow data for storage efficiency. Shadow data replication skill minimiz-es battery and communication failures of unreliable sensor nodes in the lower clusters. Consequently, unnecessary communication costs and battery consumption can be reduced, and overall battery energy balance can be maintained by dynamically managing sensor nodes in the clusters.

5. REFERENCES

1. P. Bonnet, J. Gehrke, and P. Seshadri, Towards Sensor Database Systems, Proceedings of the Sec-ond International Conference on Mobile Data Management, Hong Kong (2011) January, pp.3-14.
2. S. Zöller, A. Reinhardt, M. Meyer, R. Steinmetz, Deployment of Wireless Sensor Networks inLogistics- Potential, Requirements, and a Testbed., Proceedings of the 9th GI/ITG KuVS (2010) September, pp.67-70.
3. Y. Yao, J. Gehrke, Query Processing for Sensor Networks, IEEE Pervasive Computing. (2003), Vol.3, No.1, pp.46-55.
4. S. Byun, and S. Jang. Asymmetric Index Management Scheme for High-capacity Compressed Databases, Journal of Korea Academia-Industrial, (2016), Vol. 17, No.7, pp.293-300.
5. DHL, https://www.logistics.dhl/content/dam/dhl/global/core/documents/pdf/glo-core-blockchain-trend-report.pdf, BLOCKCHAIN IN LOGISTICS (2019)
6. C. Cachin and M Vukolic. Blockchain consensus protocols in the wild. Technical Report arXiv:1707.01873, IBM Research - Zurich, July (2017).
7. S. Ahn, and K. Kim. A Join Technique to Improve the Performance of Star Schema Queries in Column-Oriented Databases, Journal of Korean Institute of Information Scientist and Engineers, (2013.6), Vol. 40, No.3, pp. 209-218.
8. D. Abadi, R. Samuel Madden, N. Hachem, ColumnStores vs. RowStores: How Different Are They Really?, Proceedings of the ACM SIGMOD'08, (2008) June 9-12, pp.967-980, Vancouver, BC, Canada.
9. http://www.oberhumer.com/products/lzo-professional/ Feb. 12 (2018).