A Dynamic Host Selection Algorithm for Layered Data Storage Architecture in a Pervasive Space

Zhuzhong Qian[1], Ilsun You[2], Youyou Lu[3], Sanglu Lu[1]
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China[1]
School of Information Science, Korean Bible University, South Korea[2]
Department of Computer Science and Technology, Tsinghua University, Beijing, China[3]
qzz@nju.edu.cn, isyou@bible.ac.kr, Luyouyou87@gmail.com, sanglu@nju.edu.cn

***Abstract*—Context data is important information for the behaviors of the applications in a pervasive space. To effectively restore huge amount of data, tree-liked layered storage architecture are proposed, where the leaf nodes collect the data from the sensing devices located in its domain. However, the sensing devices may be moving among different domains. In order to integrate the data from the same device, related leaf nodes should upload and store the data to a certain up-layer node, called host node. This paper presents a deep study of the data storage problem and proposes an online algorithm DHS to dynamically select the host node, which reduces the communication cost significantly. We prove the correctness of the algorithm theoretically. The experiment results also show that DHS is correct and effective.**

Keywords- host selection; distributed storage; pervasive space

I. INTRODUCTION

With the development of mobile devices and wireless communication, pervasive computing becomes a hot spot in both academic and industry area. In a pervasive computing environment, sensing and monitoring equipments are deployed to collect context data, which is used to recognize users' activities, collecting resource status, and etc. These devices (e.g. body sensors, RFID tags, wireless cameras) are small and cheap, and widely deployed, based on which applications can track or predicate the location and status of the objects. For example, SpotON [1] utilize RFID technology to locate an object by computing the strength of signals from several base stations and Body Sensor Network (BSN) is discussed in [2] to implement smart homes. Furthermore, some context-aware system also can dynamically change their behaviors to adapt the changing of the environment.

Currently, the memory size of sensor devices is becoming larger and the storage-centric sensor network also has been widely discussed [3-6], which argue that some of the real time data should be stored locally in the sensing devices. However, even in such a network, the sensor node cannot restore all the context data, they have to push the historical data to the servers from time to time. In the mean time, in order to synthetically analyze the system status, we need to integrate the context from different servers. In order to effectively manage the huge amount of data, the layered data storage approach is proposed and widely discussed, where several data servers are organized as a tree-like architecture and each leaf node collects context data from a number of sensing devices within its communication domain. For a sensing device, it sends data to the leaf node called *activity domain*. And then, the context data will be uploaded to the up-layer nodes based on some policies [6]. However, several types of sensing devices such as body sensors and RFID tags are moving. And because of the communication area of these devices are limited, the context data generated by them would be sent to the leaf nodes that is closed to the devices, i.e. the mobile sensing devices have several activity domains. To integrate the context data generated from the same device, we have to upload the data to the data server that can cover all the leaf nodes which have communicated with the device. Fig. 1 is an example of the storage system. As the device moves, the activity domains of the device include 11, 12 and 5.

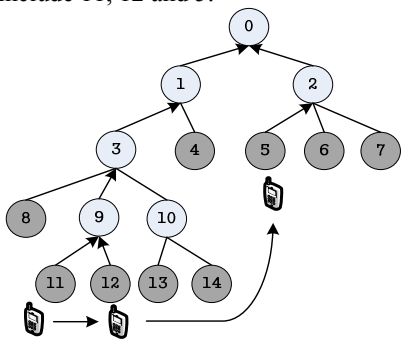

Figure 1. The layered data storage architecture

Generally, although these devices have many activity domains, they usually move among some certain domains, which are represented as *FA-domains* (Frequent Appearance domains). That is, the context data collected from these mobile devices are mainly from some certain leaf nodes. Thus, if we stored the data on the nodes that is close to these leaf nodes, it may reduce communication cost, which benefits a lot to the wireless system and power constraint devices. However, the mobile sensing devices may have different *FA-domains* in different time period. Accordingly, the best data host of a certain mobile device is dynamic and it should be changed as it moves.

In this paper, we present a deep study of the distributed context data storage for mobile sensing devices and propose an on-line host selection algorithm to dynamically evaluate and select a host that would reduce the communication cost. The main contributions are: (1) we present a deep study of the layered data storage architecture; (2) an on-line dynamic

978-0-7695-3967-6/10 $26.00 © 2010 IEEE
DOI 10.1109/CISIS.2010.42

host selection algorithm DHS is proposed to select the best host node for a mobile sensing device; (3) we theoretically approve the correctness of the approach and verify its performance by simulation experiments.

The rest of this paper is organized as follows. Section II presents some related work. Section III gives a brief introduction of the system including the context data storage and query. Section IV is the detailed description of host selection algorithm DHS and proves its correctness. Section V shows the simulation results and discusses the related issue. Section VI concludes the paper.

II. RELATED WORK

In a pervasive environment, sensing devices such as RFID tags, sensors are widely deployed. How to effectively store and query context data is a challenge for huge amount of context data management.

Gonzalez proposes a new warehousing model that preserves object transitions while providing significant compression and path-dependent aggregates [7]. In literature [8], an EPC encoding scheme using bitmap data type is presented to compress the storage of EPC tag collections.

A good data storage framework is a good method to improve the performance. Literature [9] divides framework into four categories: unstructured P2P, structured P2P, metadata integration centralized and data integration centralized architecture. In supply chain management, EPC global [10], an industry driven standard group, raises three layers of data exchange standards: Physical Object Exchange Standards, Infrastructure Standards for Data Capture and Data Exchange Standard. Intra-enterprise data exchange follows the infrastructure standards to define the interaction interfaces for data access of application. Following the data exchange standard, Inter-enterprise data exchange relies on EPC Discovery Services, ONS (Object Naming Service) and other core services. Literature [11] also proposes a query processing technique with combination of P2P to improve data sharing performance in tracking and tracing. Moreover, some work [12, 13] presents data exchange by web services. In vehicle tracking application, literature [14] presents HERO (Hierarchy Exponential Region Organization), in which hierarchy structure with restrict location updating policy is built up, to ensure real time and network traffic needs.

In addition of framework designed for specific applications, a ubiquitous framework and protocol in pervasive environment is presented in literature [16]. The framework consists of RFID readers, Savants and PML (Physical Markup Language) Servers. A PML server is selected as a HLR (Home Location Register) for a tag attached object for maintenance of data records. When the object leaves the area, a VLR (Visitor Location Register) is selected to deal the data records and communicate with HLR.

Most work is designed for specific applications. However, data repository framework for one application does not fit for the other due to specific patterns in specific applications. For example, the transport of goods in supply chain is sequentially delivered from manufacturers, distributors and retailers, while vehicle runs randomly in the whole city. Web services for data exchange used in supply chain do not work well for vehicle tracking, and vice versa.

Currently, the storage centric sensor network is proposed to store part of the data in the local node and upload the rest of the data to the upper node. And the up layered node also restores part of the data in its local area and upload the rest to the upper node till the root node. This kind of storage strategy is just a tree-like data storage architecture. However, these works focus on the data extract policy and integrate all the context data in the root which makes the root be the bottle neck of the system.

This paper has a deep study of layered data storage problem and investigates the host node selection problem to effectively restore the context data.

III. SYSTEM OVERVIEW

A. Layered Data Storage Architecture with Dynamic Host

In order to monitor the status of the pervasive computing environment, the sensing devices are widely deployed to collect the context data and send the context data to the server which is the leaf node in the layered context data storage system. Then, the data will be uploaded to the up-layer servers with a certain policy which is related to different applications. It is worth mention that the tree structure is an overlay storage framework, where each node could be matched to one or more real data servers. And some mobile devices which have more power and memory also could be a server node.

As we mentioned in section I, the mobile sensing devices may send context data to different leaf nodes, which should be integrated and stored in a certain server node, i.e. the host node. Traditionally, the server node that stores the context information from one device should cover all the active domains, so that it can obtain the context from its descendants. However, as the number of active domain grows, if a new leaf node also receive data from the device, then, all the context information has to be moved to another upper-layered nodes accordingly, which results in the heavy load in the up-layer nodes. In the mean time, only few activity domains are *FA-domains*, and the probability of a certain sensing device to some domains is very low, but in order to cover these domains, the host has to be set in a high layer. Thus, most of the context data has to "travel" a long distance to the host.

According to the above analysis, we propose a dynamic host selection strategy that selects the "closest" host for the device, instead of the node covering all the active domains. Here, the "closest" means the host is close to the most FA-domains of the sensing device. In Fig.1, we may select node 9 as the host if the device always send data to node 11 and 12 while only few data is sent to node 5.

B. Data Storage and Query based on Dynamic Host

Once the host for one device is selected, it is necessary to inform the ancestor of its host until the root. And all the data from this device should be transformed to its host node.

1) Context Data Collection

In the layered storage system, all the sensing devices only contact with the leaf nodes and send context data to them. Because both sensing device and leaf node have certain communication areas, so each device may communicate with few certain leaf nodes in a time. In this paper, we assume that the communication area of all the leaf nodes can cover the whole system, so that all sensing device can communicate with at least one leaf node in a time. If the sensing device can communicate with more than one leaf nodes at a time, it will choose the best one based on the communication quality. Once the communication between a pair of device and leaf node is setup, the device will periodically send context data to the leaf node. Each leaf node maintains a *data frequency list*, which records the number of data received from different devices.

2) Host Selection

In this storage system, each storage node in the tree has entire host index information of its descents. The format of the index is <Device_ID, Node_ID, next_hop>. That is, if one of its descent node (say node 4) is the host of device A, it has the index information <A, 4, (its parent node)> The root of the whole storage system maintains an index of the complete context data storage information.

When the host of a sensing is selected, the host node should send an index message to its parent node which will continue to send this message to the upper layer node until the root. When the device moves to a new domain, the host may shift to a new host. In this case, the new host will send a new index message to its parent and this message will be forwarded until the root. And the old host will send an update message to its up-layered nodes to delete the old index. Then, the data stored in the old host will be transferred to the new host.

3) Data Storage

Based on the above description, all the nodes may have indexes of all the host information of their descents. Thus, the root has all the host information of the whole system. Generally, when the device is moving among the *FA-domains*, the node obtains the context data is the descent node of the host of the device, and it maintains the host information. Accordingly, when the context data is sent to these nodes, the data will be uploaded to their upper-layered nodes layer by layer until the host node. However, if the device moves to a new domain, things may be more complicated. The leaf node that obtains the data may not have the host information of this device. In this case, this node will send the data to its parent which will forward the data to its up-layered node until to the node that has the host information of the host. And then, this node will transform the data to the node according to the next_hop, i.e. one of its children nodes.

4) Data Query

Context data query request could be from any node. For any query request, system analyzes the device ID of the request data and find out the host of the related device to get the context data. The query procedure is similar with the storage procedure. That is, if the node has the index of the host information, it forward the request to the next_hop, otherwise, it will forward the request to its parent node.

IV. THE DYNAMIC HOST SELECTION ALGORITHM

The goal of dynamically host selection is to reduce the communication cost. This strategy is based on the fact that a certain mobile device sends data to the different leaf nodes with different frequency. Actually, this frequency difference is big since most devices usually move with a certain domain. For example, the trace of a body sensor is exactly the owner's trace, and as we know, people mainly appear in the working place and home. Thus, choosing a node that is close to the leaf nodes in these two domains may save a lot of communication resources.

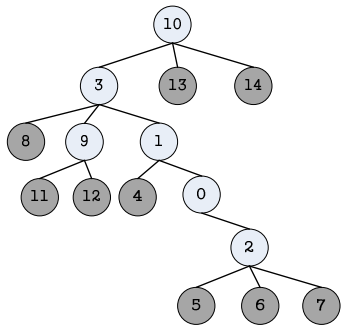

Figure 2. Context storage tree

A. Problem Description

For any mobile device, all leaf nodes may have the chance to communicate with the device and get data. When the host is selected, the storage can be converted into a new overly tree with the host node as the root, which is called *context storage tree*. Please note that this context storage tree is a logic storage structure, it only shows how the context data of a certain device is delivered from the leaf nodes to the host. Each device has its own storage tree, although they have the same physic storage architecture. Based on the physical storage architecture in Fig.1, the storage tree can be described as Fig.2, if the node 10 is selected as the host.

For a mobile device, suppose we know the probabilities that it sends data to the different leaf nodes. And the probabilities are denoted as the weight (w) of the leaf nodes. Concretely, we get the weight from the data frequency list, i.e. we use the number of communication between the device and the node to represent the frequency. Thus, the weight is monotonic increase value as the time goes on. And then, the storage cost can be represented as the weighted path length (WPL) of the tree. Thus, the host selection problem can be represented as the constructing the context storage tree with the minimum WPL.

Definition 1 Host Selection. Given a tree-like context data storage system, Let $WPL(i)$ represents the weighed path length of the context storage tree with the node i as root. The host selection problem is to find a node j, satisfy $WPL(j)=min\{WPL(i), i \in V\}$, where V is the set of all the nodes in the system. And the storage tree with the root V_j is called optimized storage tree.

B. The Dynamic Host Selection (DHS)

In a practical pervasive system, the probabilities of the device sending data to a leaf node are changing from time to

time. We cannot determine the weight of the leaf nodes and the host also should be changed as the device moving. Consequently, we propose an on-line host selection algorithm to dynamically select the host and optimize the host as the device moves on.

The DHS algorithm is composed of two parts: *i*) leaf node weight update and *ii*) host selection.

The leaf nodes update their weight periodically if the weight changes. If the weight changes, the leaf node sends the increased number of communication ∇w to its parent node. And the parent node updates the weighted path length as $WPL = WPL + \nabla w$, and the weight is updated as $w = w + \nabla w$. After that, the parent node will send the update information to its upper-layered node until to the host node.

Algorithm 1 WeightUpdateProcedure

```
1  Recieve <deltaWeight, deltaWPL> from child node;
2  w += deltaWeight;
3  WPL += deltaWPL+deltaWeight;
4  Send <deltaWeight, deltaWPL + deltaWeight> to parent node
   in current spanning tree
```

The host selection will be triggered after the weight is updated.

Algorithm 2 HostReselectionProcedure

```
1   Select node s from children set of r with maximal weight,
    current node r is root of current spanning tree;
2   IF ( 2*ws-wr>0 )
2.1     WPLr = WPLr-WPLs-ws;
2.2     wr = wr-ws;
2.3     WPLs = WPLs+WPLr+wr;
2.4     ws = ws+wr;
2.5     Set node s as the new host of tree;
2.6     Send ReselectHost to node s for reselect;
3   ELSE
3.1     host := r, that is current node r is host;
```

The host V_r select one of its children nodes with the greatest weight and calculate $\nabla = w_r - 2w_s$. If delta is less than 0, then V_s is selected as the new host. And then, the new host will do the same procedure as the original host until the delta is larger than 0 or to the leaf node. Finally, the host node would be the new root of the storage tree.

C. The Proof of Correctness

In the current storage tree with root V_m, the weighted path length is denoted as WPL_m. For the restructured storage tree with root V_n, the weighted path length is denoted as WPL_n'.

Theorem 1. In a storage tree with root V_m, V_n is one of the children nodes, if $\nabla_m = (2*w_n - w_m) > 0$, then $WPL_n' < WPL_m$.

Proof. If V_n is the root of the restructured storage tree, then the weighted path length is,

$$WPL_n' = WPL_n + (WPL_m - WPL_n - w_n) + (w_m - w_n)$$

then,

$$WPL_n' - WPL_m = -2 * w_n + w_m < 0$$

that is,

$$WPL_n' < WPL_m$$

Theorem 1 shows the necessary condition of restructuring the storage tree.

Theorem 2. In a storage tree with root V_m, if $w_n = max\{w_i, V_i \in Children(V_n)\}$, then, V_n is the candidate host.

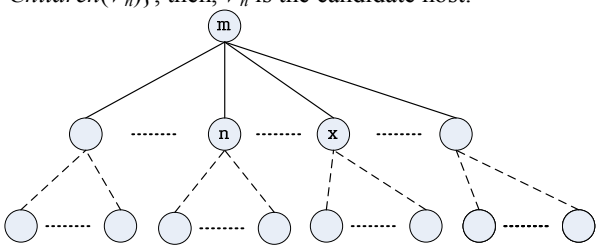

Figure 3. The candidate host

Proof. Suppose V_i is one of the children nodes of the V_m, and the weighted path length of V_i is WPL_i and weight is w_i. If V_i is the root of the restructured tree, then the weighted path length WPL_i is,

$$WPL_i' = WPL_i + (WPL_m - WPL_i - w_i) + (w_m - w_i)$$

then, for any child node of V_m, say V_x, as shown in Fig. 3, we can get,

$$WPL_x' - WPL_n' = 2(w_n - w_x)$$

because the $w_n > w_x$, thus

$$WPL_x' - WPL_n' > 0$$

Consequently, only the V_n whose weight is the biggest could be the candidate host because the WPL_n' is the smallest among the current children node of the root.

Theorem 3. For any edge e_{mn} in an optimized storage tree with root V_r, if V_n is the parent node of V_m, then, $WPL_n' < WPL_m'$.

Proof. The V_n is the parent node of V_m, there is a path from V_r to V_m via V_n, denoted as $P = <V_r, V_1 \dots V_n, V_m>$ (Fig. 4).

Suppose A_i is the success nodes set of (i-1)th node of path P except of the ith node at the path P. That is, $A_i = \{v_x | v_x \in desendant(v_{i-1}) \wedge v_x \notin subtree(v_i)\}$ where V_i is the ith node of path P.

$$w_i = \sum_{v_x \in children(v_i)} w_x$$

and let

$$W_i = \sum_{v_x \in children(v_i) \wedge v_x \in A_i} w_x$$

then,

$$W_i = w_{i-1} - w_i$$

For any e_{mn}, then we can calculate the difference ∇_m of the WPL of the storage tree with V_m as the root and the WPL of the storage tree with V_r as the root.

$$\begin{aligned}\nabla_m = WPL_m' - WPL_r &= [W_1 * d + W_2 * (d-1) \dots + W_{d-1} * 2 + W_d] \\ &- [W_{d+1} * d + W_d * (d-1) \dots + W_3 * 2 + W_2] \\ &= W_1 * d + W_2 * (d-2) + \cdots + W_{d+1} * (1-d)\end{aligned}$$

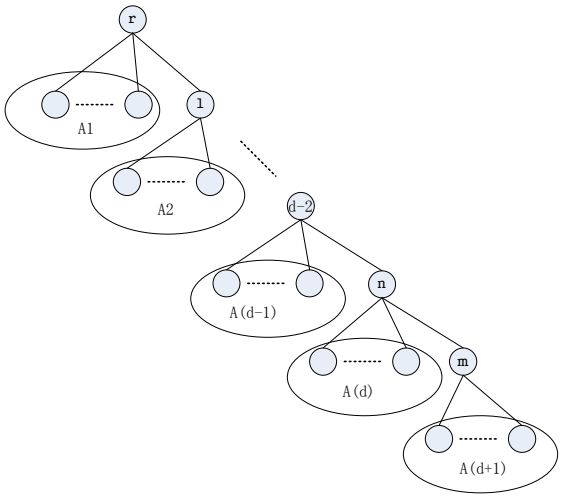

Figure 4. The path from root to V_m

We also can get $\boldsymbol{\nabla_n}$,

$$\boldsymbol{\nabla_n} = WPL'_n - WPL_r = W_1 * (d-1) + W_2 * (d-3) + W_3 * (d-5) + \cdots + W_{d-1} * (3-d) + W_d * (1-d)$$

Thus,

$$\boldsymbol{\nabla_m} - \boldsymbol{\nabla_n} = W_1 + W_2 + W_3 + \cdots + W_{d-1} + W_d - W_{d+1}$$

Since the V_r is the root of the storage tree, we can get $\boldsymbol{\nabla_m} = W_1 * d + W_2 * (d-2) + W_3 * (d-4) + \cdots + W_d * (2-d) + W_{d+1} * (-d) > 0$, thus,

$$\boldsymbol{\nabla_m} - \boldsymbol{\nabla_n} > (W_1 + W_2 + W_3 + \cdots + W_{d-1} + W_d) - \left(\frac{W_1 * d + W_2 * (d-2) + W_3 * (d-4) + \cdots + W_d * (2-d)}{d}\right) = \frac{W_2 * 2 + W_3 * 4 + \cdots + W_d * (2d-2)}{d} > 0$$

Thus, $\boldsymbol{\nabla_m} - \boldsymbol{\nabla_n} = WPL'_m - WPL_r - (WPL'_n - WPL_r) = WPL'_m - WPL'_n > 0$.

After the weights of tree are updated, the maximum weighted child node V_s is selected to compare with the root. If the weight of this node satisfies $(w_n - 2w_s) < 0$, then we know that the storage tree with the root of V_s is better than the original tree. Otherwise, the program is aboard, because the root of the optimized storage tree cannot be the node (Theorem 1); it is also impossible on the subtree of this node (Theorem 3); and other children nodes are also impossible to be the new root (Theorem 2). Thus, we can see that when the program is terminated, the new storage tree would be the optimized storage tree and the root is the host of the device.

V. EXPERIMENTAL ANALYSIS

In order to testify the correctness and the performance of the algorithm, we simulated 14 data servers (9 leaf nodes) and built the layered data storage system as the Fig.1 shows. The communication among the data nodes are based on the TCP socket. We also implemented the data generator to simulate the generation of context data.

A. The Comparison of Different Host

Based on the layered data storage system, these 2 experiments show that selecting different host node results in different communication cost.

In the first experiment, we generate a sequence with 20 context data sending events, which is represented as the sequence of ID: {13, 13, 13, 13, 13, 13, 14, 14, 14, 14, 14, 14, 8, 8, 11, 11, 12, 12, 4, 4, 4, 5, 6, 7}. The total number of data collection is 24 and the probability of leaf node data receiving is as {0.125(4), 0.04(5, 6, 7), 0.08(8, 11, 12), 0.25(13, 14)} where the number in the parenthesis represents the node ID. Obviously, in this experiment, the *FA-domains* of the device are node 13 and 14.

In the second experiment, we generate the sequence as {5, 5, 5, 5, 6, 6, 6, 6, 6, 7, 7, 7, 7, 13, 13, 13, 14, 14, 14, 8, 11, 11, 12, 12}. The total number of data collection is still 24 while the probability is like {0.17(5, 7), 0.21(6), 0.125(13, 14), 0.04(8), 0.08(11, 12)}. Here, the *FA-domains* are node 5, 6 and 7.

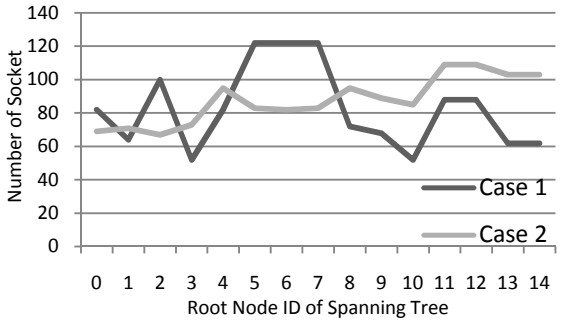

Figure 5. Communication overhead for each spanning tree

The Fig.5 shows the results of the communication cost of choosing different nodes as host. We can see that in the first experiment, if we choose node 3 or node 10 as the host node, the communication cost is minimized, which are only 50 socket communications. And in the second experiment, node 2 is the best node as the host, and the communication number is 67 times.

B. The Correctness of the DHC

To test the correctness of the algorithm, we get the trace of the host selection procedure with the same data set in the above sub section and summarized in Table I. It finally chooses the node 10 as the host which is the correct host as the above result. And for the send data set, we also get the node 2 as the host.

TABLE I. DYNAMIC HOST SELECTION

Host Selection	The Procedure of Host Selection	The value of *w* and WPL of related nodes
In the 7th data sending events, the first host selection is triggered, including 5 sub steps to finally choose node 13 as the host node.	0: 0 -> 1	0: (6, 24) 1: (6, 18)
	1: 1-> 3	1: (6,18) 0: (0,0) 3: (6,12)
	3: 3 -> 10	3: (6,12) 1: (0,0) 10: (6,6)
	10: 10 -> 13	10: (6,6) 3: (0,0) 13: (6,0) 14: (0,0)
	13: host	13: (6,0) 10: (0,0)
In the 15th data sending events, the second host selection is triggered, including 2 steps to finally choose node 10 as the host node.	13:13->10	13: (14,18) 10: (8,10)
	10: host	10: (14,16) 13: (6,0) 14: (6,0) 4: (2,2)

C. The Performance of DHS

The goal of the DHS is to reduce the communication cost of the system during the data storage. This part we will test the communication cost of DHS itself. The cost of DHS algorithm includes the cost of weight update and host reselection procedure.

Fig. 6 shows the communication cost of DHS under the situation that the device moves in a low speed, i.e. it does not frequently switch from different domains. Since most of the sensing devices move as it, we consider it as the normal case. As we can see, the cost of DHS is very low compared with the normal communication. More concretely, $\frac{\text{Host Selection cost}}{\text{Total cost}} = \frac{1}{22}$. And Fig. 7 shows the communication cost under that the device is moving fast, and it sends data to several different nodes in a short time. In this case, the cost of DHS occupied nearly half communication cost in some nodes, because of the frequent host reselection. And $\frac{\text{Host Selection cost}}{\text{Total cost}} = \frac{17}{41}$.

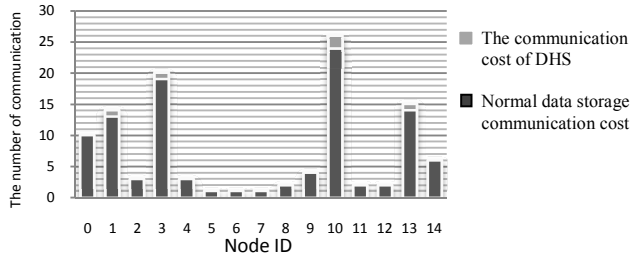

Figure 6. The cost of host selection in normal case

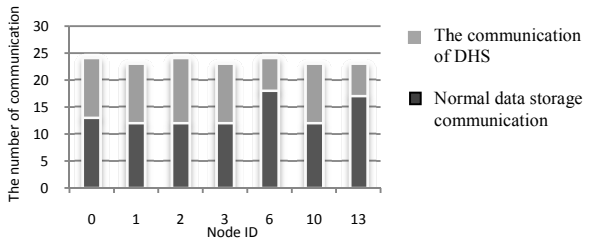

Figure 7. The cost of host selection in frequent domain switch

D. Discussion

As Fig. 7 shown, if the host was changing frequently, the cost of DHS would be high. We call the frequent host switch as host jitter. As we analyzed, the necessary conditions of host jitter are as follows,

- There are two nodes with a relatively long distance have the similar data receiving probability, and
- The nodes in the middle of the path between these two nodes have the similar data receiving probability which is smaller than the two nodes, and
- The device sends data to these two nodes alternately.

In a practical system, the behaviors of sensing devices are relatively stable. The devices seldom skip between two domains and have the asymmetric data sending ratio. Consequently, in a normal case, the costs of host selection only take up a small portion of the total communication costs. And in some special cases, we also can set a threshold to reduce the number of host reselection, so that we can avoid the host jitter.

VI. CONCLUSION

In this paper, we investigate the layered context data storage problem. Since the mobile devices send data to different nodes, which needs to be merged and stored in some certain host node. How to select a suitable host with a minimum communication cost is a challenge. We propose an on-line host selection algorithm DHS to dynamically select host node. We also prove the correctness of the algorithm and set up simulation experiments to approve the performance of DHS. In future work, we will deploy it in a large-scale system and get more experimental data to improve the efficiency of the system.

REFERENCES

[1] G Hightower, R Want, SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength, 2000.

[2] G Zhou, J Liu, C Wan, M Yarvis, and J Stankovic, BodyQoS: Adaptive and Radio-Agnostic QoS for Body Sensor Networks, Infocom, April 2008.

[3] A Deshpande, C Guestrin, S Madden, J M Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In VLDB, 2004.

[4] S Madden, M Franklin, J Hellerstein, and W Hong. Tinydb: An acqusitional query processing system for sensor networks. ACM TODS, 2005.

[5] G Mathur, P Chukiu, P Desnoyers, D Ganesan and P Shenoy, A Storage-centric Camera Sensor Network, Proceedings of the Fourth ACM Conference on Embedded Networked Sensor Systems - Demonstrations (SenSys), Boulder CO, November 1-3, 2006.

[6] Y. Diao, D. Ganesan, G. Mathur and P. Shenoy, Rethinking Data Management for Storage-centric Sensor Networks. Proceedings of the Third Biennial Conference on Innovative Data Systems Research (CIDR), Asilomar CA, January 7 - 10, 2007.

[7] J.H. Hector Gonzalez, Xiaolei Li, and Diego Klabjan, Warehousing and Analysis of Massive RFID Data Sets, Proc. 2006 Int. Conf. on Data Engineering, 2006.

[8] S.S. Ying Hu, Timothy Chorma, Jagannathan Srinivasan, Supporting RFID-based Item Tracking Applications in Oracle DBMS Using a Bitmap Datatype, Proceedings of the 31st VLDB Conference, 2005.

[9] J.A. Hong-Hai Do, Gregor Hackenbroich, Architecture Evaluation for Distributed Auto-ID Systems, International Conference on Database and Expert Systems Applications, 2006.

[10] EPC global, The EPC global Architecture Framework – EPC global Final Version 1.3 Approved 19 March 2009.

[11] R Agrawal, A Cheung, K Kailing, S Schonauer, Towards Traceability across Sovereign, Distributed RFID Databases, Proc. of the 10th Int. Database Engineering & Applications, 2006

[12] S Chalasani, R Boppana, Data Architecture for RFID Transactions, IEEE Transactions on Industrial Informatics, 2007

[13] UCLA WinRFID Middleware. http://www.wireless.ucla.edu/rfid/winrfid.

[14] M.L. Hongzi Zhu, Yanmin Zhu, Lionel M. Ni, HERO: Online Real-Time Vehicle Tracking. IEEE Transactions on Parallel and Distributed Systems, Volume 20, 2009.