

國立交通大學

網路工程研究所

碩士論文

無線感測器網路中動態資料中心儲存方法之研究

A Study of Dynamic Data-Centric In-Network Storage in
Wireless Sensor Networks

研究生：周裕庭

指導教授：曾煜棋 教授

中華民國九十七年十月

無線感測器網路中動態資料中心儲存方法之研究
A Study of Dynamic Data-Centric In-Network Storage in Wireless Sensor
Networks


研究生：周裕庭

Student：Yu-Ting Chou

指導教授：曾煜棋

Advisor：Yu-Chee Tseng

國立交通大學
網路工程研究所
碩士論文



A Thesis
Submitted to Institute of Network Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in
Computer Science

October 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年十月

無線感測器網路中動態資料中心儲存方法之研究

學生：周裕庭

指導教授：曾煜棋 教授

國立交通大學網路工程研究所碩士班

摘 要

我們探討一個距離基地很遠的無線感測器網路，其中之感測器節點可以偵測不同型態的資料。使用者在網路內部發出即時且集合性的資料詢問。例如：過去四小時內的平均溫度為多少？為了避免不必要的資料傳輸以造成能量消耗，以及加強資料安全性，節點收集到的資料將會被儲存在網路內部，而不是傳送回在遠方的基地。針對此問題，我們提出一個雙層、群集式、資料中心的儲存方法。我們的目標是達到有效率的資料擷取，以及延長網路生命。我們把網路虛擬地切割成群集，不同型態的資料將會被儲存到不同的群集。而在群集內部，資料將會以環狀複製法來儲存。我們提出兩種資料型態和群集的對應方法：雜湊法與查表法。

關鍵字：無線感測器網路、資料中心儲存法、網路群集、無線傳輸。

A Study of Dynamic Data-Centric In-Network Storage in Wireless Sensor Networks

Student : Yu-Ting Chou

Advisor : Prof. Yu-Chee Tseng

Institute of Network Engineering

National Chiao Tung University

ABSTRACT

We study the problem of a wireless sensor network system which is deployed far from the base station and is for collecting heterogeneous types of data. Users make queries of real-time, collective data inside the network. For example, what is the average temperature in the last 2 hours? For avoiding unnecessary data transmissions and enhancing data security, the collected data are stored inside the sensor network itself instead of transmitting all the data back to the base station. We propose a two-tier, cluster-based, data-centric storage system for the studied scenario. Our goals are to achieve efficient storage and retrieval of collective data, and to extend system lifetime through load balancing. In our system, the network is virtually divided into clusters. Different types of sensed data will be stored in their corresponding clusters. Inside a cluster, data is replicated in a ring-based fashion. As for deciding which cluster the data will be forwarded to, we propose two methods: hashing and table mapping. Both methods provide load balancing to prolong network lifetime.

Keywords: wireless sensor networks, data-centric storage, clustering, wireless communication.

致謝

首先，由衷感謝曾煜棋教授在研究期間的耐心指導，讓我能夠順利完成此篇論文並且取得碩士學位。

此外也要感謝王友群學長，為我的論文提供許多寶貴的建議和細心的指導。以及 HSCC 實驗室的全體成員，在這段期間給我的所有幫助與勉勵，謝謝大家。

最後，要感謝我的母親，永遠帶給我溫暖的關懷和最深切的期許。當然還要感謝可愛的詠晴，總是給予我最貼心的支持與鼓勵。

周裕庭 於
國立交通大學網路工程研究所碩士班
中華民國九十七年十月



Contents

| | |
|-------------------------------------|-----|
| 摘要 | i |
| Abstract | ii |
| 致謝 | iii |
| Contents | iv |
| List of Figures | v |
| 1 Introduction | 1 |
| 2 Related Works | 3 |
| 3 Network Initialization | 5 |
| 4 Cluster Allocation | 8 |
| 4.1 Hashing Method | 8 |
| 4.2 Table-Mapping Method | 9 |
| 5 Ring-based Data Replication | 11 |
| 6 Simulation Results | 13 |
| 7 Conclusions | 16 |
| References | 17 |



List of Figures

| | |
|--|----|
| Figure 1: The proposed scenario of our system. | 1 |
| Figure 2: A clustered network example. | 6 |
| Figure 3: Example of routing table inside each node. | 7 |
| Figure 4: A cluster with 3 rings. | 12 |
| Figure 5: The effects of the allocation methods on query latency. | 14 |
| Figure 6: The effects of the allocation methods on system lifetime. | 14 |
| Figure 7: The effects of replication methods on query latency. | 15 |
| Figure 8: The effects of replication methods on system lifetime. | 15 |



Chapter 1

Introduction

A wireless sensor network (WSN) is a collection of wireless sensors, which are deployed in interested regions to gather information in a collaborative manner [2]. Recent advances in the technologies of wireless sensor networks, especially the developments of the sensors which can simultaneously sense different kinds of data, known as the heterogeneous/multi-function sensors[8, 1], have led to a wide variety of applications. In this paper, we study the scenario where the wireless sensor network is deployed far from the base station. Heterogeneous sensors are used to sense the specific types of data based on the application. The scenario is depicted in figure 1. Users roaming inside or at the boundaries of the network may make queries for different types of data. The applications include environmental exploration and military surveillance. For instance, the deployed field can be a jungle or a desert where users are researchers collecting the information of temperature, humidity, sounds and lightness. Alternatively, the deployed field can be a battle field where agents are in need of the data of vibrations, pressure, etc. Here are several query examples: What is the average temperature in the last 4 hours? What is the maximum lightness discovered in the last 6 hours? Or what are the total detected vibrations in the last 2 hours?

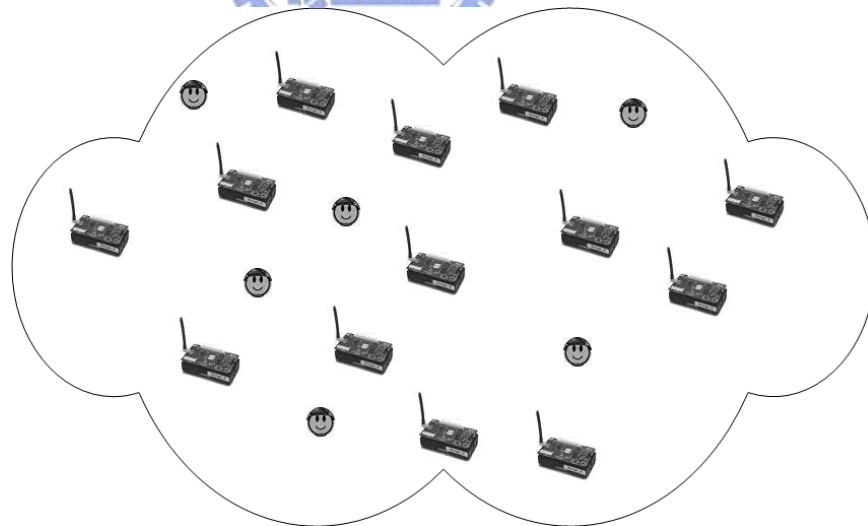


Figure 1: The proposed scenario of our system.

In this paper, we present a data storage and retrieval system in wireless sensor networks for the proposed scenario. Because the sensing field is distant from the base station and the users who will try to retrieve sensing data are closer to the network itself than the base station, our system is a sensor network without an external sink. All the data generated is stored inside the network, instead of transmitting back to a base station.

There are two reasons for storing the data locally. First, not all the data generated will be useful to the users. Since long-range communication is energy-consuming, a significant amount of energy will be wasted on transmitting uninterested data. Second, the gathered data can be confidential, especially in the military applications. Long-range transmissions will increase the possibilities of data overhearing by hostile parties. Due to energy-saving and security concerns, in-network data storage is carried out in our system.

In order for users to retrieve collective data, we design a cluster-based data storage system. Because it is both time-consuming and energy-consuming to broadcast the query to the whole network every time a user wants to retrieve a certain type of data, we propose to congregate all the generated data in a specific location in the network. Then all the queries of the same type will subsequently go to the place where the data is stored. Moreover, we are considering heterogeneous types of data. Therefore, there will be more than one location inside the network to serve as a data reservoir. In our system, we virtually divide the network into clusters by first selecting landmark nodes as cluster heads. Different types of data will be forwarded and stored in different clusters. Two methods are provided to achieve the data-cluster mapping. First, we consider the hashing method, by which the data will go to the hashed cluster according to a random hash function. Second, we use a table mapping method. All the data will use a table mapping method to decide where it should be forwarded to. For wireless sensor networks, energy-saving is an important issue. Both of our mapping methods provide load balancing to avoid energy holes. The hashing method achieves load balancing by the randomness property of the hash function, while the table-mapping method achieves load balancing by a more centralized ranking maintenance technique.

Inside the cluster, data will be stored by a ring-based replication technique. Initially, data is stored in all of the one-hop neighbors of the cluster head. Replications only occur when the query rate to this cluster becomes higher than a predefined threshold. With this replication technique, we achieve load balancing by alleviating the heavy loads of certain nodes. Our system is a two-tier structure because all the inter-cluster communications are independent of intra-cluster ones.

The remainder of this paper is organized as follows: In Section 2, we discuss some related works. In Section 3, we describe some network initialization issues. In Section 4 we present our cluster allocation methods. The ring-based data replication is presented in Section 5. Simulation results are examined and discussed in section 6. In Section 7, we conclude the paper.

Chapter 2

Related Works

Directed Diffusion [7] is an in-network, data-centric model for data storage and retrieval in sensor networks. In Directed Diffusion, the sink nodes flood their queries (attribute - value pairs) to the network and set up gradients toward the originating node. Sensor nodes with readings that match the query will propagate their data to the sink node according to these gradients. Although it does not need an external base station and can achieve in-network data retrieval, it is not suitable for our scenario because each query requires a flooding to the network. In our scenario with a possible high query rate, the network will be depleted in a short period of time.

TTDD [12] improves the flooding problem of Directed Diffusion by separating the network into a two-level hierarchy. Every time a node detects data, it constructs a virtual grid structure in the network. Whenever a query is made at a node, the node only floods the query in its own grid and then follows the grid points to the data generating node. By the two-level hierarchy, TTDD avoids flooding to the whole network, but it requires all the nodes to have their own geographic information. In our work, the sensor nodes do not need to know their geographic locations to achieve a two-level hierarchy. Also, in TTDD, the sensed data is stored at the sensing node. If TTDD is used in our scenario, queries of collective data might be scattered around the whole network to gather the needed information. The total energy dissipation of a single query will be very large.

GHT[11] is another data-centric storage mechanism in sensor networks. Data is hashed to geographic locations in the network by the content of the data. Users apply the same hash function to retrieve the data by forwarding the query to the hashed location. The disadvantage of GHT is that when a certain data becomes a popular query, the hashed location of the data turns into a hotspot. The energy consumption of the nodes storing the data becomes very high and causes the nodes to die rapidly. In our work, we dynamically distribute the load of the nodes that are queried more often than others by the table mapping method described in section 4.2.

Clustering in wireless sensor networks is a well-studied subject. Nodes in a wireless sensor network are grouped into clusters according to their geographic positions to enhance network scalability and achieve energy saving. Many clustering algorithms [3, 4, 9, 6] have been proposed for wireless ad hoc and sensor networks.

However, the algorithms in [3, 4, 9] require clock synchronization and only generate one-hop clusters. Hence they are only suitable for networks with a small number of nodes. LEACH[6] is a distributed clustering-based routing protocol. Its main purpose is minimizing global energy usage in a wireless sensor network. LEACH is designed for networks with the base station located far from the sensors while our work focuses on networks without external base stations. [5] is a landmark-based mechanism which groups the sensor nodes by the Voronoi graph generated from the cluster heads (landmarks). It is distributed and performs local data aggregation. However, it applies to one type of data while our work can scale to multiple types of data in a same network.



Chapter 3

Network Initialization

We assume a wireless sensor network without a sink. A sink is an external node which is responsible for gathering the collected data from the network and transmitting the data back to a remote base station. There are two reasons for this sink-less design. First, the base station may be very distant from the sensor network according to our application scenarios. Transmitting all the data back the base station would cost a lot of unnecessary energy consumption because not all of the data would be useful to the users. Second, there are security concerns in a battlefield application. Local data storage is considered safer than long-range transmissions.

We also assume that the nodes are static because it would be costly and unnecessary to deploy mobile nodes in our scenarios. The nodes do not have the information of its geographic location. Each node has a certain amount of internal memory which is used to store the gathered data and other routing information. The memory could be built-in flash memory or memory cards. Also we assume all the nodes are energy-aware. Each node can monitor its own residual energy to facilitate the energy-saving algorithms of the network.

The initialization begins with the clustering of the sensor nodes. Here we use the landmark selection scheme described in [10]. Each node in the network generates a random number v in $[0,1)$. Then it waits for Kv time period, where K is an adjustable parameter. The time unit is the time it takes to send a packet from a node to its neighbor. During this waiting period, if a node is not suppressed by any other node, it declares itself a cluster head. This cluster head then broadcasts its status to suppress all the non-cluster head nodes. This is a simple and distributive heuristic to select the cluster heads and to achieve good distribution. Figure 2 is an example of a clustered network.

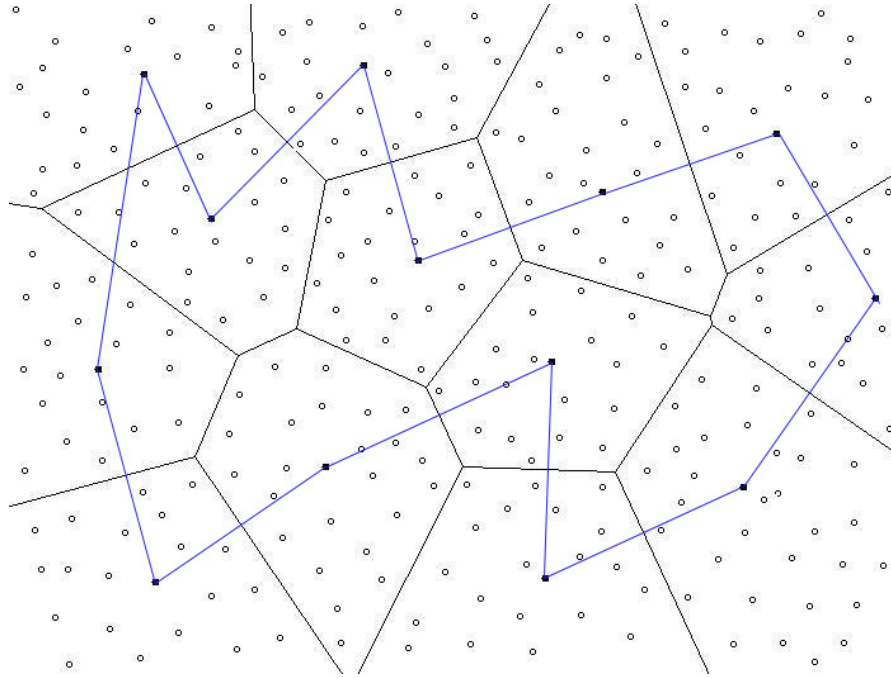


Figure 2: A clustered network example. The black nodes are the cluster heads and the white nodes are normal nodes. An *update path* is set up to connect all the cluster heads in the network.

Initially, each node has its own unique ID. Upon the selection of cluster heads, every selected cluster head broadcast a *find* message which contains its node ID and a hop counter which is initially set to 0. When a non-cluster head node receives a *find* message, it caches the message. Therefore it knows the minimum distance (hop counts) to the originating cluster head. Then it broadcasts the message again with the hop counter incremented by 1. Whenever the node receives another find message with the same cluster ID, it compares the received hop counter with the counter that is already cached in its memory. If the newly received counter is smaller than the cached value, the node would cache and broadcast the new message with the counter incremented. Otherwise, the node simply drops it.

After the above process, the final node ID in each node's cache with the smallest hop count value is the node's cluster head. Also, each node knows the distances to all the cluster heads in the network. Since all the transmissions in our network are between a non-cluster head node and a cluster head, the routing is done by passing the packet to the neighbor with the smallest distance to destination.

When the initialization is completed, each node in the network has set up its own routing table. The table is shown in figure 3. Every node has the knowledge of all the cluster heads and the knowledge of how to forward a packet to any one of the cluster

heads. The “data type” field in figure 3 will be determined in the next chapter: Cluster Allocation.

| Dest. CH | NextHop | DataType | Sensing Data |
|----------|---------|----------|--------------|
| 1 | 71 | 1 | |
| 2 | 73 | 2 | |
| 3 | 73 | 3 | |
| 4 | 68 | - | |
| 5 | 74 | - | |

Figure 3: Example of routing table inside each node.



Chapter 4

Cluster Allocation

We assume that our network contains multiple types of sensor nodes. Each node has the ability to sense multiple types of data. For example, in a scientific research inside a forest, sensor nodes may be designed to collect temperature, humidity, lightness, etc. In a battlefield, sensors are designed to gather pressure, vibration, sounds, even pictures and videos. The number of data types d in a network is predefined and fixed. We will cluster the network for storing those collected data. Each data type will be forwarded to a designated cluster. In this paper, we present two methods to map data types to clusters: *hashing* and *table mapping*.

4.1 Hashing Method

A hash function is a mathematical function for turning some kind of data into an integer. In our design, all nodes have the same set of built-in hash functions. The inputs to these hash functions are the data types. The outputs of these functions are integers between 0 and the total number of nodes. We use the output number y to determine the corresponding cluster to which a type of sensor data will be forwarded to. For example, with a certain data type d_l and a hash function f , the function operates as $f(d_l) = y$. Then we choose the cluster head whose node ID x is closest to y , that is, $|x - f(d_l)|$ is smallest, to serve as the storage of the sensor data of type d_l . In this way, the cluster-data type relationship is decided.

Whenever a query is made or some data is generated at a specific node, the node will perform the hash function with the type of the data as input. Then it forwards the query or the data to the cluster whose cluster head ID is closest to the resulting output from the hash function. Since all the nodes have the same set of hash functions, the hashing method can work correctly.

In order to balance the workload of cluster heads, the nodes can change the hash function after a predefined period p . All the nodes will pick the same next function in a synchronous manner. Also, all the stored data have an expiration period e . For the storage and retrieval of data to work correctly, we set the hash function changing period as a multiple of the data expiration period, that is, $p = k * e$ where k is an adjustable integer.

The hashing method achieves load balancing through the random nature of the

hash functions. Another advantage of the hashing method is that besides a predefined universal function set and changing sequence, the method is totally distributed. No extra communication between the cluster heads is needed. However, because of the randomness, more than one type of data might be forwarded to the same cluster. This will cause congestion and rapid decrease of energy in the same area, and will lead to an early hole inside the network. Thus the effectiveness of the network is diminished. Therefore, we provide another method to avoid the above disadvantage.

4.2 Table-Mapping Method

Our second method for cluster allocation is table mapping. It is more centralized than the hashing method in the previous section, but it is more effective in prolonging the network lifetime.

In this method, we choose the cluster head with the lowest Node ID as the coordinator. Right after the setup phase is completed, the coordinator computes an *update path* which is a route containing all the cluster heads. A mapping table which provides the cluster – data type relationship is used in this method. The coordinator is responsible of manipulating and adjusting the mapping table.

In the beginning, the coordinator maps one data type to one cluster head. Because the number of data types d is fixed, we adjust the clustering parameter K to make sure the number of clusters is two to three times of d . We use a simple network example to demonstrate the table mapping method. Consider a network with 10 clusters, whose cluster heads are numbered 1, 2, 3 ... 10. Assume that five data types are to be handled by the network, that is, $d = 5$. The coordinator is the cluster head with ID = 1. It initially maps the five data types to the first five cluster heads respectively, leaving cluster heads 6~10 unused. We use a vector representation to show the mapping. In this example, the mapping vector is (1, 2, 3, 4, 5, -, -, -, -, -). Then it sends the mapping table through the update path to inform all the other cluster heads the current mapping vector. Since all the cluster heads have the mapping table, whenever a query is made at any node inside the network, the node can send a direct request to its local cluster head to check which cluster it should be forwarding the query to.

We conduct a dynamic maintenance in the method. Each cluster head records a *query count* (qc) which is the number of queries arrived at that cluster in a time period p . For every p , all cluster heads exchange through the update path a *count vector* which contains the information of all the cluster heads' residual energy and their individual query counts. In our example from the previous paragraph, the count vector after the

first period p could be (30|90%, 120|15%, 10|95%, 5|95%, 5|95%, 0|100%, 0|100%, 0|100%, 0|100%, 0|100%). The first number in each item is the query count recorded by the corresponding cluster head. The second number is the node's residual energy. A low energy threshold E_{low} and two query threshold θ_1 and θ_2 are predefined depending on the application circumstances. After each period, the coordinator will do a rearrangement to adjust the mapping table. A cluster heads whose qc is higher than θ_n , $n=1$ or 2 , will get help from n more clusters to alleviate its loading. Also, a cluster head whose remaining energy is lower than E_{low} will step down and become a non-storing cluster head. In our example we set $E_{low} = 20$, $\theta_1 = 20$ and $\theta_2 = 40$. Therefore, after p , the cluster - data type mapping will change from (1, 2, 3, 4, 5, -, -, -, -, -) to (1, -, 3, 4, 5, 1, 2, 2, 2, -). If the available clusters are not enough for the new mapping arrangement, the coordinator will firstly give the new clusters to the data type whose qc is the highest. If there are unused clusters left, then they will be given to the second highest, and so on.

From the example, we can derive that the more popular data types will be stored in more clusters. Therefore, the consumed energy will be dispersed and the probability that network holes will happen is reduced. The energy threshold E_{low} is used to prevent the nodes from dying by stopping its storing functionality. The threshold allows a node to become a normal node to simply transmitting data. Also, from the above example, we can see that the disadvantage of two or more data types mapping to a single cluster head in the hashing method is avoided.

In this method, we exploit the fact that queries to a particular data type might be made successively. Therefore, the popularity of different types of sensor data is an important factor in data retrieval. For instance, in the environmental research scenario, when the temperature changes rapidly, all the researchers will be asking the temperature information in the same period of time. Or in the battlefield scenario, when our soldiers start hearing unfamiliar sounds which might come from the hostile troops, the sound information will certainly be heavily queried in a short period of time..

Chapter 5

Ring-based Data Replication

All the previous discussions are focused on the communication between clusters. In this chapter, we present an intra-cluster optimization for data storage and retrieval. The method presented in this chapter is independent from the previous cluster-level mapping algorithms, so it can be applied to both the hashing and the table mapping methods.

Inside each cluster, there are two straight-forward methods to store its sensing data. The first one is to store all the data in the cluster head. However, it may quickly consume the cluster head's energy and thereby cause a hole. The second method is to flood the data to all the nodes inside the cluster. It minimizes the retrieval latency and avoids the cluster head from dying too fast. But it also could waste too much energy in copying sensing data to all the nodes, because some of the copied data might not be queried at all.

Here, we propose a *ring-based* data replication method to find a balance between the above two straight-forward methods. Inside each cluster, the nodes which are k hops to the cluster head are called k -ring. For example, in figure 4, we show a cluster with 3 rings. Initially, the data are replicated in 1-ring. When a node which has the replicated data receives a query, it sends a plus message to the cluster head to increase the local query count. For every sub-period sp , the cluster head calculates the query rate qr which is the local query count divided by the time interval. We also have a set of predefined threshold $T_j, j = 1, 2, 3 \dots$. For every sp , if $T_j \leq qr < (T_{j+1})$, data will be replicated to $(j+1)$ -ring.

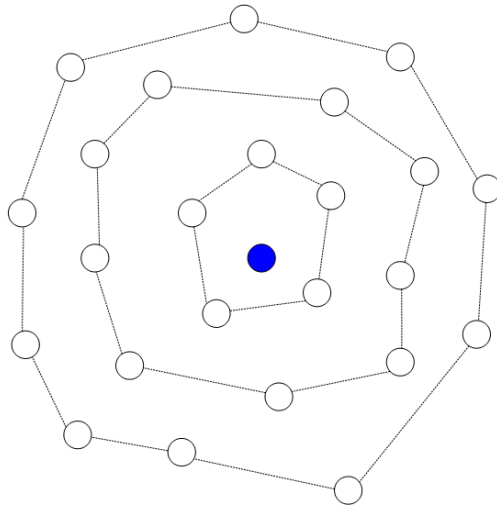


Figure 4: A cluster with 3 rings.

When a query arrives at a cluster, it will forward the query from the largest ring to the cluster head. If it finds the data matching the query on its way to the cluster head, it stops forwarding and retrieve the data. Because of the data-centric property of our system, the query is a (data-time) pair. For example, if the query asks the temperature from the previous 4 hours, and the data in the largest ring only contain the temperature from the previous 2 hours, the query and the data are not matched and the query will be forwarded to the next ring to find the matching data.

In the ring-based data replication method, we only replicate the data to more nodes when the data becomes more popular, therefore avoid the unnecessary data copying and lowering the probability of causing an energy depletion hole near the cluster head.

Chapter 6

Simulation Results

We develop our simulation in Java. 2000 nodes are randomly distributed inside a 316m x 316m field. The communication range of the nodes is 11m. With this setting we can achieve a reasonable average connectivity of 6.x. We ignore the power consumption for sensing and data processing because they are significantly less than the power consumed by transmitting and receiving. We assume each communication between two sensor nodes takes one time unit and costs one unit of power. The battery power is set to 1000 units of power.

First, we do a simulation for comparing the hashing and the table-mapping method. We also compare the two methods with the naive broadcasting mechanism. In the broadcasting method, users simply flood their queries to the whole network to gather data. Figure 5 shows the average query latency of the three methods. The latency is the time between a query is made and the needed data is received by the user. The X-axis represents how many queries have been made in the network. The Y-axis is the average latency of the queries. We can observe from figure 4 that both hashing and table-mapping are better than the broadcasting method. Hashing performs slightly better than table-mapping in the beginning because of the fact that table-mapping requires extra communication between the cluster heads. However, table-mapping outperforms hashing as time goes by. It is because table-mapping is more dynamic and is capable of adjusting cluster allocation according to the query rate.

In figure 6, we show the impact of the methods on system lifetime. The X-axis is still the query numbers. The Y-axis now represents the number of dead nodes in the network. A node which has zero power remaining is considered a dead node, because it can not perform any functions. From figure 6, we can see that the broadcasting method causes a lot of dead nodes in a short time. The hashing method, again, performs better than the table-mapping method because hashing does not require extra communications between cluster heads. The table-mapping method, on the other hand, has a better performance in the long run. This is also due to the fact that it has a dynamic mechanism to prolong system lifetime by monitoring the query rates of the system and the residual energy of the sensor nodes.

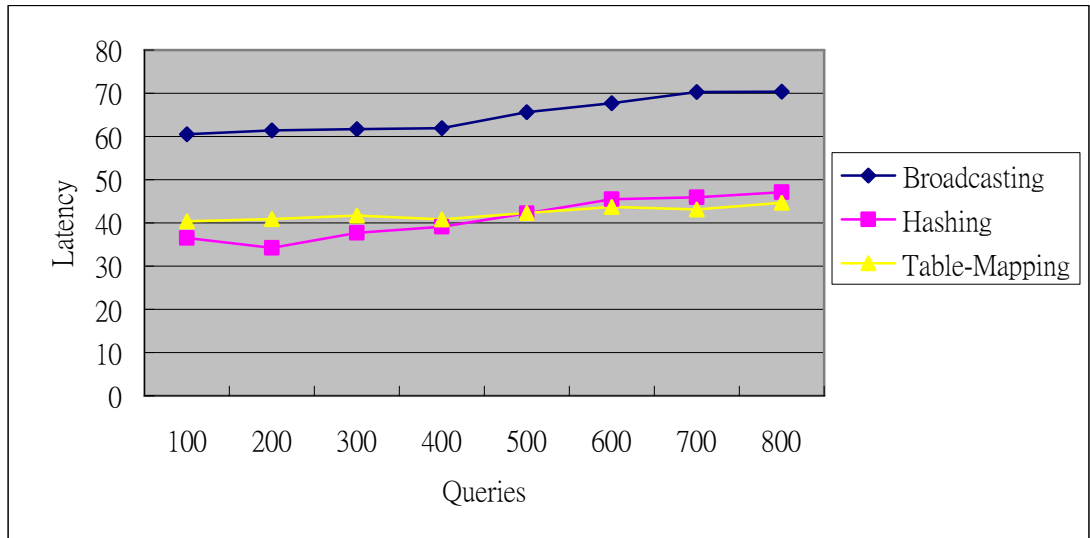


Figure 5: The effects of the allocation methods on query latency.

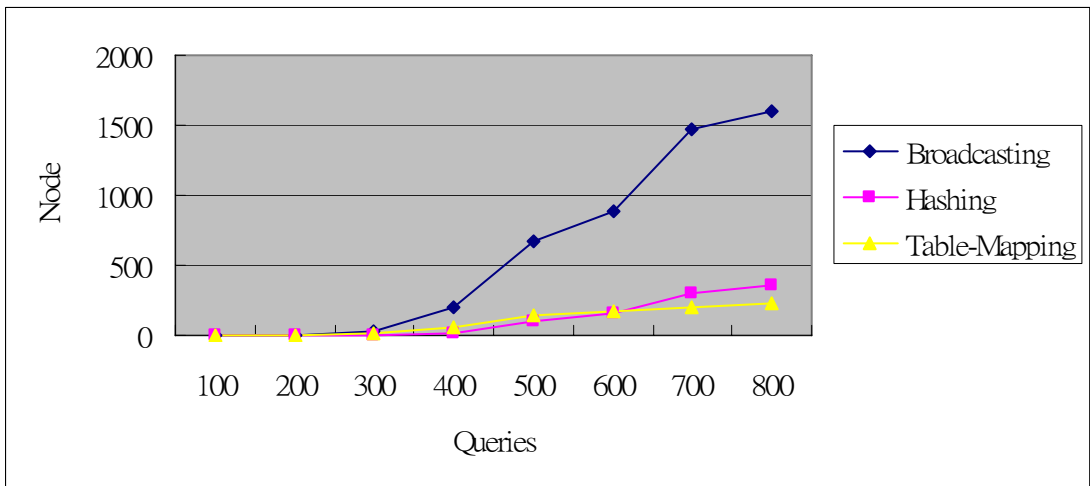


Figure 6: The effects of the allocation methods on system lifetime.

We examine the ring-based replication by the same metrics: average query latency and system lifetime. Here, the query latency represents the time between a query arrives at the cluster and the matching data is retrieved and leaves the cluster. We compare the ring-based replication with two other mechanism of storing data inside a cluster. The first one is storing all the data in the cluster head. The data is not replicated in any other nodes. The second one is flooding the data to all the nodes inside the cluster. In figure 7, we can see that the best query latency is achieved by flooding. It is obvious because every query can retrieve their desired data immediately at any node. The no-replication method performs the worst because every query will have to travel to the cluster head in the middle of the cluster. Ring-based replication has query latency between the other two methods.

Although flooding has the best query latency, its energy consumption is very high. This fact is shown in figure 8. When the query counts approach 500, almost all the nodes die at the same time. The no-replication method and the ring-based replication method have similar performance in the beginning. However, when the query counts increase, the cluster head and nearby nodes in no-replication will die rapidly. From the simulation results, we can derive that ring-based replication significantly improves the system lifetime, and also has respectable query latency.

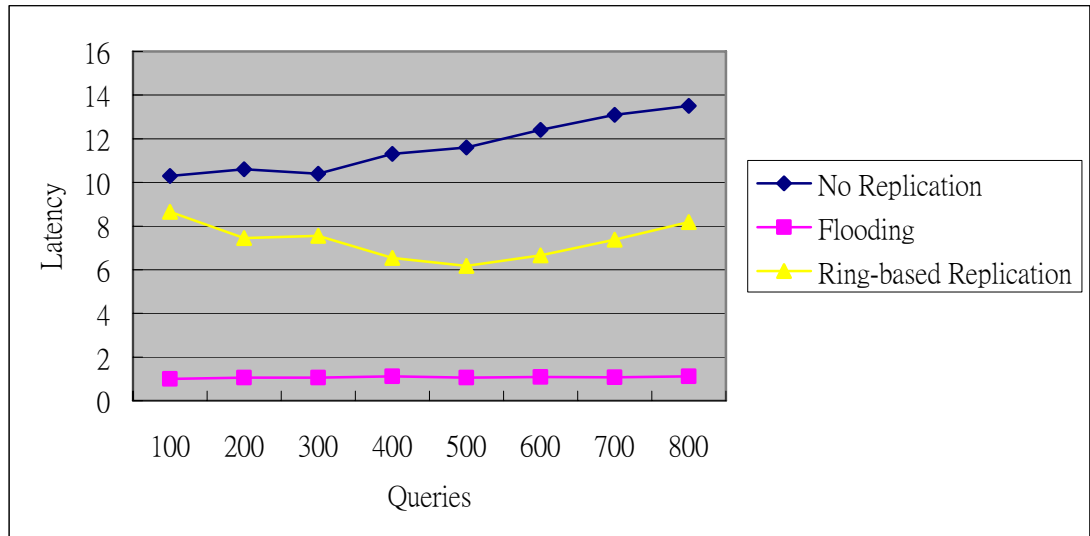


Figure 7: The effects of replication methods on query latency.

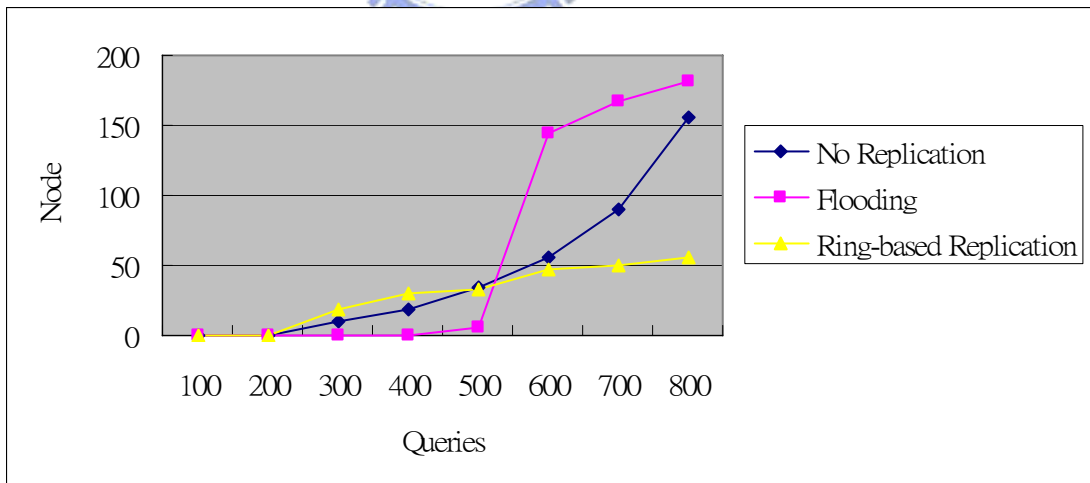


Figure 8: The effects of replication methods on system lifetime.

Chapter 7

Conclusions

In this paper, we studied the scenario of a wireless sensor network which has no external base station. We proposed two cluster-based, data-centric methods for in-network data storage and retrieval for the studied scenario. In our simulation, we have shown that the two methods both provide better query latency than the straight-forward broadcasting method, and both prolong system lifetime. We also proposed a ring-based data replication scheme for intra-cluster storage. Our simulation showed that the ring-based replication can prolong network lifetime and maintain good query latency.



References

- [1] <http://www.freepatentsonline.com/5971597.html>.
- [2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, *Wireless sensor networks: a survey*, 2001.
- [3] D. J. Baker and A. Ephremides, *The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm*, IEEE Transactions on Communications, Vol. 29, No. 11, pp. 1694-1701, November 1981.
- [4] S. Basagni, *Distributed Clustering for Ad Hoc Networks*, in *Proceedings of International Symposium on Parallel Architectures, Algorithms and Networks*, 1999, pp. 310-315.
- [5] Q. Fang, J. Gao and L. J. Guibas, *Landmark-Based Information Storage and Retrieval in Sensor Networks*, In *Proc. of the 25th Conference of the IEEE Communication Society (INFOCOM' 06)*, 2006.
- [6] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan, *Energy-Efficient Communication Protocol for Wireless Microsensor Networks*, *Proceedings of the 33rd Hawaii International Conference on System Sciences*, 2000.
- [7] C. Intanagonwiwat, R. Govindan and D. Estrin, *Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks*, *Pro. 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 2000)*. 2000.
- [8] S.-T. Kim and C. Nguyen, *A Millimeter-Wave Multifunction Sensor for Wireless Monitoring of Displacement and Velocity*, *IEEE/ACES International Conference on Wireless Communications and Applied Computational Electromagnetics, 2005.*, 2005.
- [9] C. R. Lin and M. Gerla, *Adaptive Clustering for Mobile Wireless Networks*, *Journal on Selected Areas in Communication*, Vol. 15 (1997), pp. 1265-1275.
- [10] A. Nguyen, N. Milosavljevic, Q. Fang, J. Gao and L. J. Guibas, *Landmark Selection and Greedy Landmark-Descent Routing for Sensor Networks*, *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, 2007.
- [11] S. Ratnasamy, B. Karp and D. Estrin, *GHT: A geographic hash table for data-centric storage*, *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications, Atlanta, GA*, 2002.
- [12] F. Ye, H. Luo, J. Cheng, S. Lu and L. Zhang, *A Two-Tier Data Dissemination Model for Largescale Wireless Sensor Networks*, In *Proceedings of International Conference on Mobile Computing and Networking (MobiCom)*, 2002.

