國立交通大學

網路工程研究所

碩 士 論 文

使用 Mobile Mule 對孤立的無線感測網路進行資料收集

Using Mobile Mules for Data Collection in an Isolated Wireless Sensor Network

研 究 生：賴婉婷

指導教授：曾煜棋　教授

中 華 民 國　九 十 七　年 六 月

使用 Mobile Mule 對孤立的無線感測網路進行資料收集

Using Mobile Mules for Data Collection in an Isolated Wireless Sensor Network

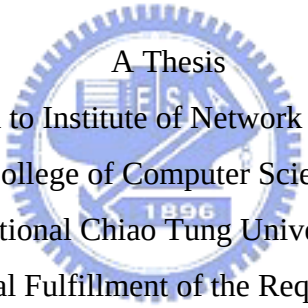
研 究 生：賴婉婷　　Student：Wan-Ting Lai

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

June 2007

Hsinchu, Taiwan, Republic of China

中華民國九十七年六月

使用 Mobile Mule 對孤立的無線感測網路進行資料收集

學生：賴婉婷　　指導教授：曾煜棋老師

國立交通大學資訊工程學系(研究所) 碩士班

摘　要

我的論文研究考慮一個 Isolated Wireless Sensor Network 的情況下，當一個感測網路與外界不能連通時，因為每個感測器測節點的記憶體是有限的，如何做到有效率的資料儲存管理。將整個網路的記憶體空間想成是一個分散式的儲存系統，網路上有行動式收集裝置 mule 來特定的 sink 點收集資料，我們探討下列三個問題：(1) 如何降低因為記憶體不足而產生的封包流失，(2) 當丟棄封包的情況無法避免，如何避免高重要性的封包被丟棄掉，(3) 因為空間或時間的限制，mule 不能一次收集到全部的資料時，如何將高重要性的封包儲存在靠近 sink 的地方，使得 mule 可以先收集較重要的資料。我們提出一個 DSMS 的協定來解決上述的問題。DSMS 是一套分散式的演算法，它使用類似於 heap sort 的封包交換方式，透過與鄰居間交換封包來達到我們的目標。透過實作以及模擬來證明 DSMS 的效用。

Using Mobile Mules for Data Collection in an Isolated Wireless Sensor Network

Student: Wan-Ting Lai Advisors: Prof. Yu-Chee Tseng

Department of Computer Science and Information Engineering
Nation Chiao Tung University

Abstract

This paper considers storage management in an isolated WSN, under the constraint that the storage space per node is limited. The memory spaces of these sensor nodes can be regarded as a distributed storage system. Assuming that there is a sink in the WSN that will be visited by mobile mules, we address three issues: (1) how to buffer sensory data to reduce data loss due to shortage of storage spaces, (2) if dropping of data is inevitable, how to avoid higher priority data from being dropped, and (3) how to keep higher priority data closer to the sink, such that the mobile mules can download more important data first when the downloading time or mule storage is limited. We propose a Distributed Storage Management Strategy (DSMS) based on a novel shuffling mechanism similar to heap sort. It allows nodes to exchange sensory data with neighbors based on only local information. To the best of our knowledge, this is the first work addressing distributed prioritized storaging for isolated WSNs.

誌謝

首先感謝我的指導教授曾煜棋老師給我的指導，在他的帶領下，我才能順利完成我的論文。感謝指導我的黃啟富學長，巫芳璟學姊，讓我在研究過程中常給我適當的幫助與建議。也感謝實驗室所有的成員，讓我於研究過程中能彼此切磋成長。感謝中華扶輪教育基金會提供給我的獎學金。最後感謝我的母親，讓我在她的愛心和支持中，完成碩士學位。

Contents

List of Figures

Chapter 1

Introduction

Wireless sensor networks (WSNs) have gained much attention recently [8]. A WSN is composed of a large number of nodes, each of which is a microprocessor with multiple sensors onboard. Nodes can communicate with one another through their wireless interfaces. WSNs have many applications, such as military safety, health care, and environment surveillance [1][4][9][12].

In this paper, we consider an *isolated* WSN [10] that is disconnected from outside world for the most of the time. It thus relies on *mobile mules* [11] to visit it and carry its sensory data to the outside world. A WSN may be isolated for many reasons, such as node failure owing to destructive events or distance and cost constraints. For example, a WSN could be deployed under water to monitor undersea oilfields [14]. Such applications do not need real-time information, so data can be collected once in several months.

Since mules may not visit an isolated WSN frequently, how to buffer data is a challenging issue. We thus address the related storage management problem, under the constraint that the storage space per node is limited. (For example, MPR300CB has only 4KB RAM and 4KB ROM [3].) We regard the memory spaces of sensor nodes as a distributed storage system. Assuming that there is a *sink* in the WSN that will be visited by mobile mules, we address three issues: (1) how to buffer sensory data to reduce data loss due to shortage of storage spaces, (2) if dropping of data is inevitable, how to avoid higher priority data from being dropped, and (3) how to keep higher priority data closer to the sink, such that the mobile mules can download more important data first when the downloading time or mule storage is limited. In (3), the remaining data may be downloaded next time. We propose a *Distributed Storage Management Strategy (DSMS)* for data buffering in an isolated WSN. DSMS is designed based on a novel shuffling mechanism similar to heap sort [13] to keep higher priority data closer to the sink. However, unlike heap sort, which is based on a tree structure, DSMS uses a mesh-like structure to facilitate data exchange and thus to keep higher-priority data close to the sink.[1]

There have been many works related to mules. Data collection using mules

[1]Note that heap sort must be conducted a complete binary tree. Insertion begins at a leaf and moves up toward the root, while deletion begins by removing the root element, moving the rightmost leaf element to the root, and then adjusting the heap. These operations are basically centralized operations and can not be implemented in a realistic distributed WSN environment.

is addressed in [5][6][11]. Reference [5] analyzes the upper bound of the optimal data transfer with mules. In [6], it shows that using mules with predictable mobility can significantly reduce communication power in WSNs. Using mules to connect sparse sensor networks at the cost of higher latencies is explored in [11]. The routing problem in a highly disconnected ad hoc network using mobile ferries is discussed in [7][15][16]. However, how to buffer packets generated by an isolated WSN remains an obscure problem. To the best of our knowledge, this is the first work addressing distributed prioritized storage for isolated WSNs using mobile mules.

The rest of the paper is organized as follows. Section 2 presents our system model. DSMS is given in section 3. Some extensions of DSMS are in Section 4. Section 5 contains our simulation results. Section 7 concludes this paper.

The rest of the paper is organized as follows. Chapter 2 presents our system model. DSMS is given in Chapter 3. Some extensions of DSMS are in Chapter 4. Chapter 5 contains our simulation results. Our implementation results are shown in Chapter 6. Chapter 7 concludes this paper.

Chapter 2

System Model

We consider a heterogeneous WSN consisting of some *static sensors* and *mobile mules*. Static sensors, or simply nodes, can continuously monitor the environment and periodically generate *reporting packets*, or simply *packets*. Each node has the same storage space of S_{sn} (in unit of packet) and communication range of R_{sn}. Two nodes u and v can communicate with each other if their distance $dist(u, v) \leq R_{sn}$. These static sensors form a connected network through multi-hop routing. The WSN is deployed in a remote field and is isolated from the outside world. Mobile mules are thus designed to collect sensory data from them. They can stop by a specific node, called *sink*, for a period of time to collect sensory data. During this period, the sink can relay its own and others' packets to the mule. The stop-by period can be a fixed or a random length. The movement of mules can be by intention (pre-arranged) or by chance (opportunistic, such as a

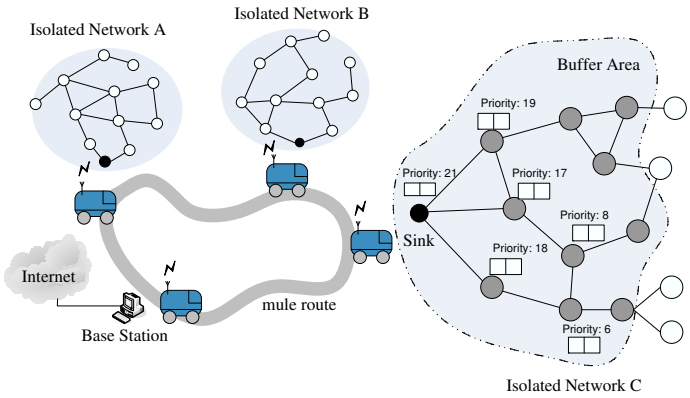


Figure 2.1: System model of the DSMS system.

passing-by bus). These mules can later deliver the collected packets to an external base station. Fig. 2.1 shows an example.

We assume that packets generated by nodes have priorities to reflect their importance. It can be assigned by a pre-agreed function, by an aging process, or by sensor readings (which may reflect level of emergency). Higher priority means more importance. Packets will be stored in a *Buffer Area (BA)*, which is defined as the set of nodes that are within a certain distance from the sink. In Fig. 2.1, the BA of network C contains nodes within 3 hops from the sink. We regard the storage spaces of nodes in BA as a distributed storage system. Our goal is to design a distributed protocol to achieve three goals.

G1 : Dropping of packets should be minimized.

G2 : If dropping of packets is necessary, the lower-priority ones should be dropped first.

G3 : To facilitate mobile mules to collect data, higher priority packets should be stored closer to the sink.

Definition 1. *Given a graph $G = (V, E)$, a $sink \in V$, a subset $BA \subseteq V$, and a priority assignment function F for packets, the problem is to develop a packet exchange protocol to maintain packets being generated by the WSN such that G1-G3 are met and $\sum_{v \in BA, p \to v} F(p)$ is maximized, where $p \to v$ means that a packet p is stored at the storage of v.*

Chapter 3

Properties and Protocols of DSMS

DSMS is a distributed solution. Nodes not in BA will forwards their packets to BA, while nodes in BA will observe their neighbors' states and exchange packets as necessary. We assume that after the WSN is deployed, each node u has calculated its distance $D(u)$ to the sink and its neighbor set $N(u)$. Without loss of generality, we assume that each node u has only one buffer space (i.e., $S_{sn} = 1$). So the (only) packet in u is written as $P(u)$ and its priority is $F(P(u))$ (if u has no packet, $F(P(u)) = -1$). Our scheme can be easily extended to $S_{sn} > 1$.

DSMS tries to maintain the following properties for each node $u \in BA$

P1 : For each node $v \in N(u)$ such that $D(v) > D(u)$, $F(P(v)) \leq F(P(u))$.

P2 : For each node $v \in N(u)$ such that $D(v) < D(u)$, $F(P(v)) \geq F(P(u))$.

P3 : For each node $v \in N(u)$ such that $D(v) = D(u)$, $max\{F(P(w))|w \in N(u), D(w) > D(u)\} \leq F(P(v)) \leq min\{F(P(w))|w \in N(u), D(w) <$

$D(u)\}$.

P1 (resp., **P2**) implies that nodes that are farther from (resp., closer to) the sink than u should have lower-priority (resp., higher-priority) packets than u. **P3** enforces that nodes that have the same distance to the sink as u should have the same property as u. When a node has the above properties, we say that it is *in-order*. In Fig. 3.1 (a), every node is in-order except node m and j.

For each node u, we let $maxPost(u)$ be the packet with the highest priority of all neighbors v of u such that $D(v) > D(u)$, $minPre(u)$ be the packet with the lowest priority of all neighbors v of u such that $D(v) < D(u)$, $maxEqual(u)$ be the packet with the highest priority of all neighbors v of u such that $D(v) = D(u)$, and $minEqual(u)$ be the packet with the lowest priority of all neighbors v of u such that $D(v) = D(u)$. Based on the above properties, we design our packet exchange rules for node $u \in BA$ as follows:

E1 : When $F(maxPost(u)) > F(P(u))$, node u tries to exchange packet with $maxPost(u)$.

E2 : When $F(P(u)) > F(minPre(u))$, node u tries to exchange packet with $minPre(u)$.

E3.1 : When $F(maxEqual(u)) > F(minPre(u))$, these two packets are exchanged.

E3.2 : When $F(maxPost(u)) > F(minEqual(u))$, these two packets are exchanged.

The above rules are event-triggered ones. These events are triggered when a node changes its packet, including exchanging with others or generating a new one, or when its neighbors change their packets. When multiple events are triggered, a node should prioritize rules **E1, E2, E3.1**, and **E3.2** in that order because we prefer nodes exchanging with those at different distance first. For u to exchange packet with v, it can send a *Request_To_Exchange (RTE)* to node v. Node v, if agrees, replies a *Clear_To_Exchange (CTE)*. Then the exchange can be conducted. These operations should be atomic. If an exchange happens, a node should broadcast the priority of its new packet to its neighbors.

For a node $u \notin BA$, when it has a packet, it will try to send it to any neighbor v such that $D(v) < D(u)$. When a node $w \in BA$ receives the packet, it will accept it if $F(P(w)) = -1$, drop it if $F(P(w)) \geq F(P(u))$, and replace $P(w)$ by $P(u)$ if $F(P(u)) > F(P(w))$.

Fig. 3.1 gives an example. Node a is the sink and there is a new packet with priority 12 arriving at node m in Fig. 3.1(a). Node m will realize that it violates **P2** and will exchange with node j by **E2** as shown in Fig. 3.1(b). The same will happen to nodes j, f, and b, resulting in the scenario in Fig. 3.1(c). Now j finds that it violates **P3** because $F(P(i))$ is not between 10 and 4. So j will notify i and m to exchange their packets by **E3.2**. Similarly g will find that it violates **P3** after

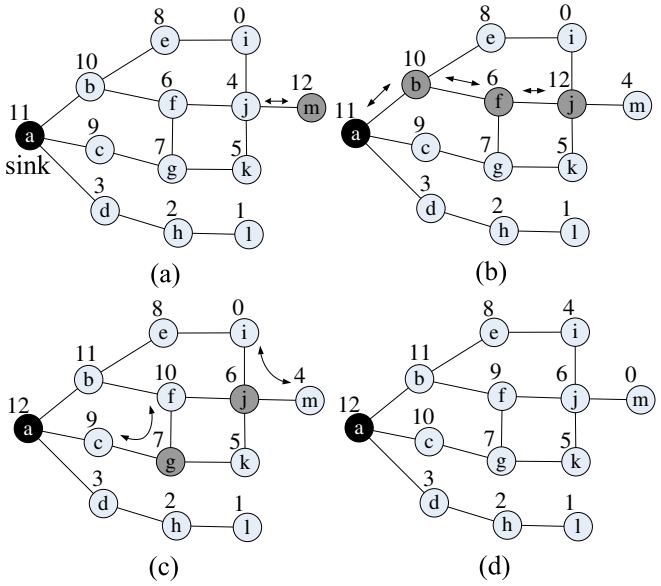


Figure 3.1: An example DSMS packet exchanges.

receiving f's broadcast and notify c and f to exchange their packets by **E3.1**. The final result is in Fig. 3.1(d), where every node is in-order. Note that DSMS does not guarantee an optimal arrangement of packets since it is a distributed protocol and relies only on neighboring information.

Below, we formally prove that DSMS will ultimately stop in a in-order status. We say a packet is *stable* if this packet is stored in a certain node and will not exchange to other nodes, until it is collected by a mule or new packets with higher priority are generated. Next, we first show each packet will become stable in finite time, which means DSMS will eventually stop. Then we show each node is in-order while DSMS stops.

Theorem 1. *Given any arrangement of packets in BA, if no packets are gener-*

*ated during exchange, the packet exchange following **E1**, **E2**, **E3.1** and **E3.2** will eventually stop in finite time.*

Proof. It is obvious that the packet with the highest priority eventually migrates to the sink and becomes stable once it reaches the sink. Once the highest prioritized packet becomes stable, the packet with the second-high priority can become stable once it reaches one of sinks neighbors. Similarly, each packet can become stable if all the packets with higher priorities than it become stable and it reaches the place as close to the sink as possible. Note that, it is not necessary that packets become stable in the order of their priorities. But once higher prioritized packets become stable, a packet can become stable without doubt. Since the BA is limited, the packet exchange can terminate in finite steps. □

Next, we try to prove that all nodes are in-order when the packet exchange stops.

Theorem 2. *After all nodes in BA stop exchanging packets, they are in-order.*

Proof. We prove this theorem by contradiction. If node u is not in-order, then there are only three possible cases:

Case 1 : Node u violates **P1**. That is there is a neighbor $v \in N(u)$ such that $D(v) > D(u)$ and $F(P(v)) > F(P(u))$. Since $F(maxPost(u)) \geq F(P(v)) > F(P(u))$. It will not stop exchanging according to **E1**.

Case 2 : Node u violates **P2** but it follows **P1**. That is there is a neighbor $v \in N(u)$ such that $D(v) < D(u)$ and $F(P(v)) < F(P(u))$. Since $F(minPre(u)) \leq$

$F(P(v)) < F(P(u))$. It will not stop exchanging according to **E2**.

Case 3 : Node u violates **P3** but it follows **P1** and **P2**. That is there is a node $v \in N(u)$ such that $D(v) = D(u)$ and $F(P(v))$ is not between $F(maxPost(u))$ and $F(minPre(u))$. Since node u follows **P1** and **P2**, we can get $F(minPre(u)) \geq F(maxPost(u))$. The value of $F(P(v))$ is either bigger than $F(minPre(u))$ or smaller than $F(maxPost(u))$.

(1) $F(P(v)) > F(minPre(u))$. Since $F(maxEqual(u)) \geq F(P(v)) > F(minPre(u))$. It will not stop exchanging according to **E3.1**.

or

(2) $F(P(v)) < F(maxPost(u))$. Since $F(minEqual(u)) \leq F(P(v)) < F(maxPost(u))$. It will not stop exchanging according to **E3.2**.

Case 1, 2 and 3 all contradict our assumption of stoping exchanging, so it is proved. □

Because DSMS can utilize many mesh-like communication links to exchange packets, higher-priority packets have chance to stay closer to the sink by rules **E3.1** and **E3.1**. One question is: how many packet exchanges may be incurred when a new packet is generated given a stablized network. We will investigate this issue via simulations.

Finally, we comment that a mule arrives at the sink, the sink can broadcast an UPLOAD message. Then every node in BA simply tries to transmit its packet toward the sink in a greedy way.

Chapter 4

Some Extensions

We discuss two extensions below. We first extend DSMS to $S_{sn} > 1$. We define $maxMine(u)$ (resp., $minMine(u)$) to the packet of u with the highest (resp., lowest) priority. Since node may have multiple packets, the exchange rules for node u are modified as follows:

E1' : When $F(maxPost(u)) > F(minMine(u))$, node u tries to exchange its packet $minMine(u)$ with packet $maxPost(u)$.

E2' : When $F(maxMine(u)) > F(minPre(u))$, node u tries to exchange its packet $maxMine(u)$ with packet $minPre(u)$.

E3.1', **E3.2'** are the same as previous as **E3.1** and **E3.2**.

The definition of "in-order" can be directly extended to $S_{sn} > 1$. Note that nodes only need to broadcast the highest and the lowest priorities of its packets. It is not hard to prove that previous properties still hold when $S_{sn} > 1$.

The second extension is to add a few transmission buffers to each node to handle packet overflow. A packet waiting to be transmitted should be put in a transmission buffer. When a node $u \in BA$ whose storage space is full generates a new packet, it will keep packets with higher priorities and move the lowest-priority one to its transmission buffer. We assume that BA is more crowded, so such packets will be forwarded to node v, where $v \in N(u), D(v) > D(u)$ and $F(minMine(v))$ is minimum. However, this decision will not affect the correctness of our protocol.

Chapter 5

Simulation Results

We have conducted some simulations to verify our results. Unless otherwise indicated, the simulation environment contains 400 sensor nodes randomly deployed in 200×200 field, each with a transmission range of 25. The packet arrival rate is $1/50$ per node and each packet has a random priority between 0 and 1000. The BA is within 10 hops from the sink. All results are from the average of 50 test runs. Fig. 5.1 shows a snapshot of priority distribution in a network with the sink at $(0, 0)$ after applying DSMS.

We first compares DSMS using a mesh-like communication graph with a tree structure. To construct a tree structure, we reduce the communication graph into a short-path spanning tree rooted at the sink. Each node only allow to exchange packets with its parent or children. The results are shown in Fig. 5.2. Using communication graph can collect much higher priority packets than using tree

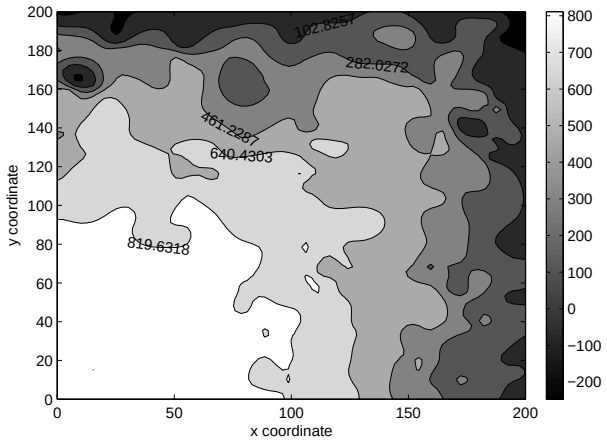


Figure 5.1: A snapshot of priority distribution.

structure.

Fig. 5.3 compares the average priorities of the packets collected by mobile mules under various stop-by intervals of mules and collected sizes (in terms of numbers nodes) of mules. We compare DSMS against *Greedy Forward (GF)*, where a node always tries to send its packets to any node closer to the sink until the latter has no storage space. OPT represents the ideal solution if global optimization is possible. Fig. 5.3(a) shows that as the stop-by interval increases, the average priority also increases. Fig. 5.3(b) shows that the average priority decreases slightly as the collecting sizes increases. However, the impact is insignificant.

Fig. 5.4 shows the effect of the BA size. In Fig. 5.4(a), we vary the BA size but fix the stop-by interval such that $1/3$ of the data in the BA can be collected. In terms of the average priority of collected packets, DSMS outperforms GF and

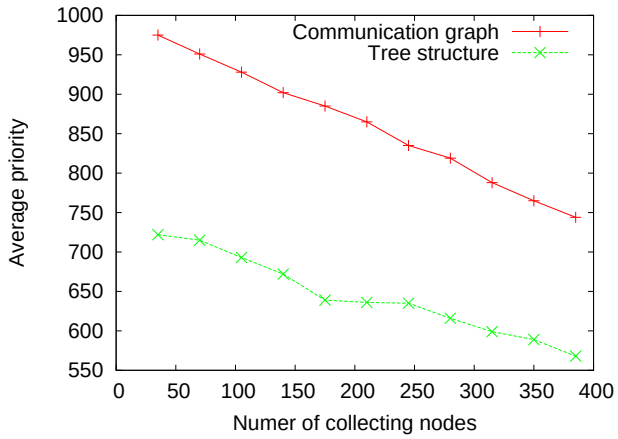
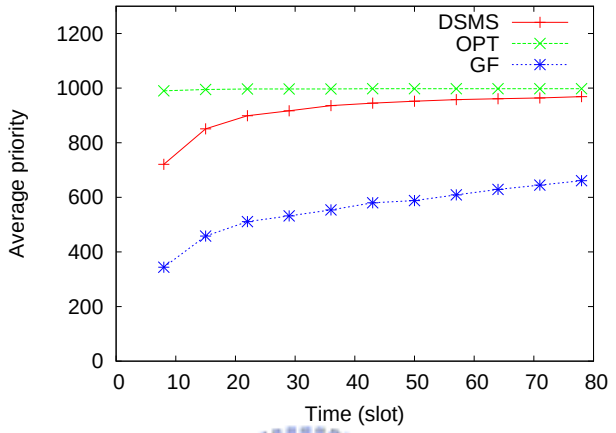


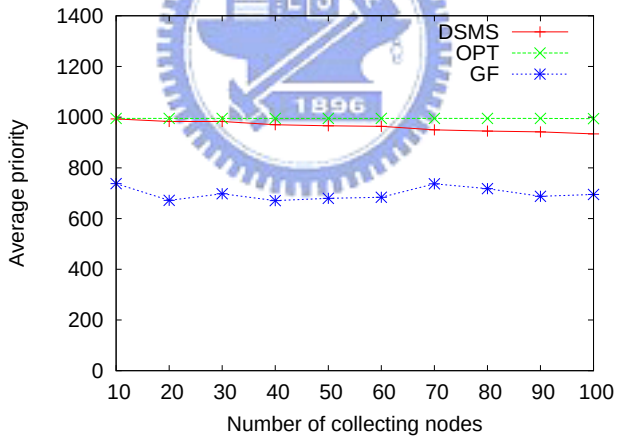
Figure 5.2: DSMS using communication graph and tree structure.

is close to OPT when the hop count is larger than 5. Fig. 5.4(b) shows that as the BA size gets larger, the data loss will decrease since the storage space is larger. On the other hand, traffic overhead will decrease first and then increase as the BA size gets larger. Each packet transmission counts for one. When the BA size is very small (say, 4 hops), packets have to travel long to reach the BA. For example, a packet with a small priority travel long before being dropped. That causes the traffic overhead keep on decreasing before the hop count reaches 7. However, as the hop count is larger than 7, there are more exchanges in BA, causing the overhead to increase.

Fig. 5.5 shows our DSMS overheads. Fig. 5.5(a) compares the overhead by varying the number of nodes and the packet arrival rate. So DSMS gets packets with higher priorities at the cost of more packet exchanges. The overhead of DSMS is about a constant higher than to that of GF. Fig. 5.5(b) shows the number

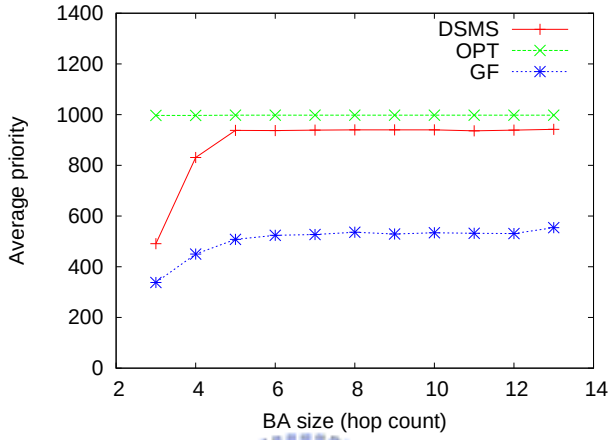


(a)

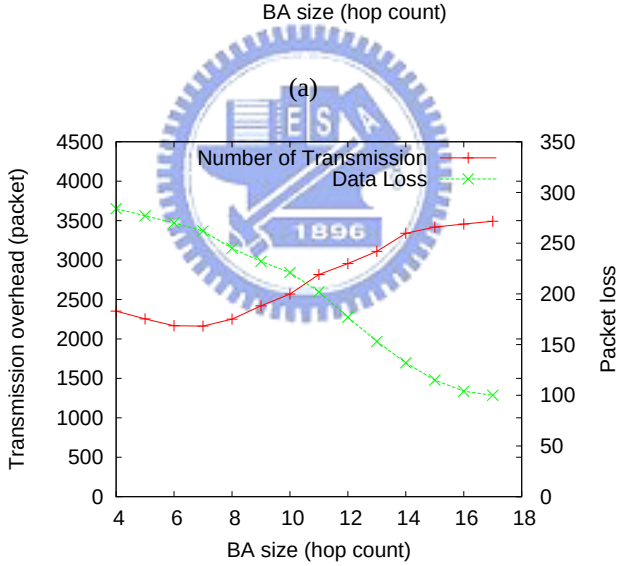


(b)

Figure 5.3: Comparison of average priorities of packets collected by mules by varying (a) stop-by interval of mules and (b) collected size of mules.



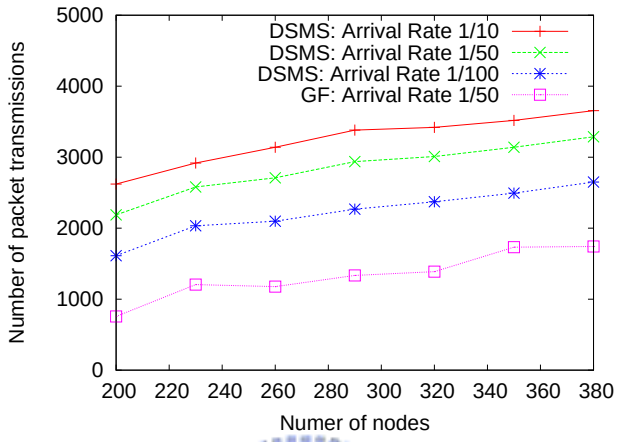
(a)



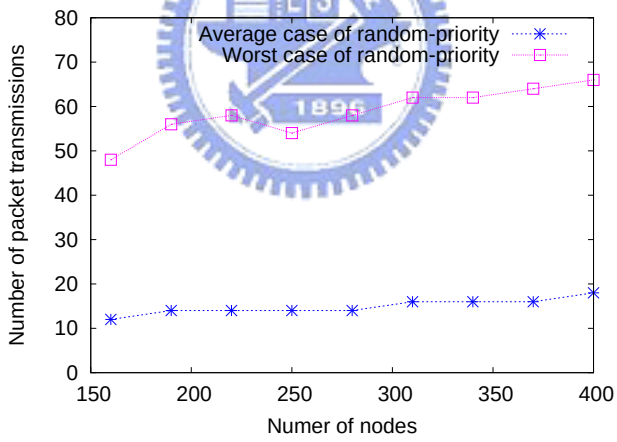
(b)

Figure 5.4: Effect of BA size on (a) average priority and (b) traffic overhead and packet loss.

of packet transmissions incurred when a new packet with a random priority is inserted into a stablized network. The transmission increases while the number of nodes increases, but the effect is insignificant.



(a)



(b)

Figure 5.5: Traffic overheads (a) when packets arrive at arrival rates and (b) when one packet arrives at a random node in a stablized network.

Chapter 6

Implementation

We implemented DSMS in real hardware platform. Our implementation includes a grid WSN and a mule. Fig. 6.1 shows our implementation structure. The WSN has 4×4 sensor nodes and a sink. The communication graph is shown in Fig. 6.2. The electric train performs in the role of mule, comes the sink to collect sensory data.

Our sensor hardware platform includes a low power, low cost wireless microcontroller, JN5139 [2] which is implemented our DSMS. We use a light sensor to generate sensory packets with priority ranging from 0 to 9 and display it on a 7-segment display. Nodes will exchange their packets according to the DSMS exchanging rules. When the mule comes to the sink, it will transmit a COLLECT_DATA message to the sink. After collecting data, the mule will transmit an ACK message to the sink and display the number of collecting data on the display.

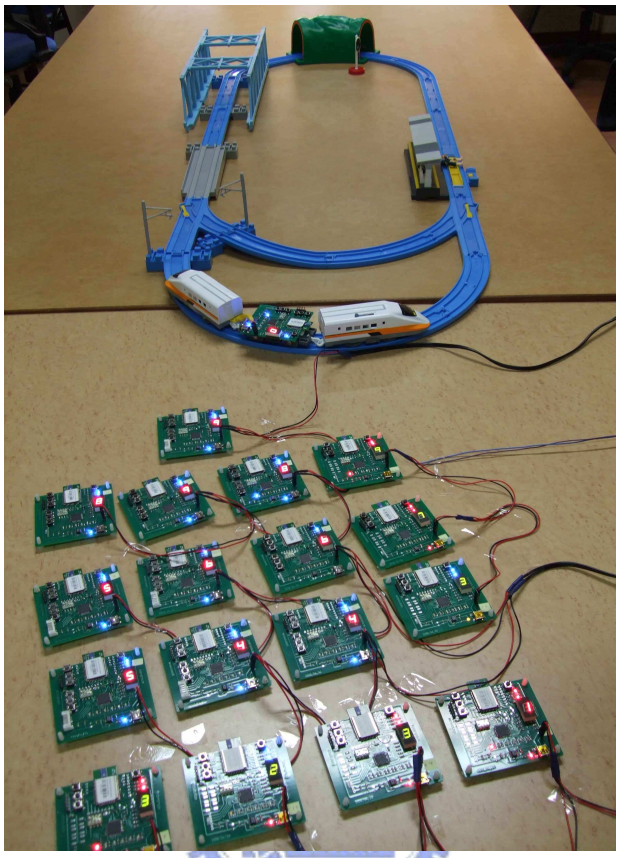


Figure 6.1: DSMS implementation structure.

Our implementation result shows that the DSMS can be easily used in a real sensor platform. Fig. 6.3 shows that after applying the exchange rules, all nodes will be in-order. Because our DSMS protocol is simple and only needs local information, it is suitable for distributed WSN environment.

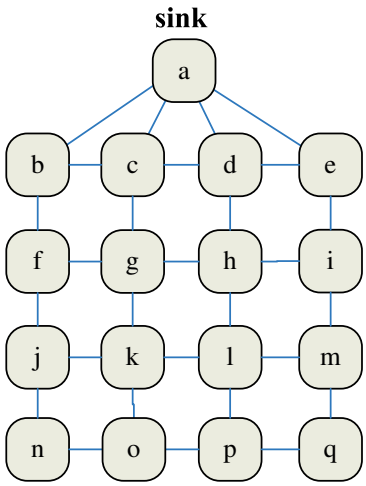


Figure 6.2: DSMS implementation communication graph.

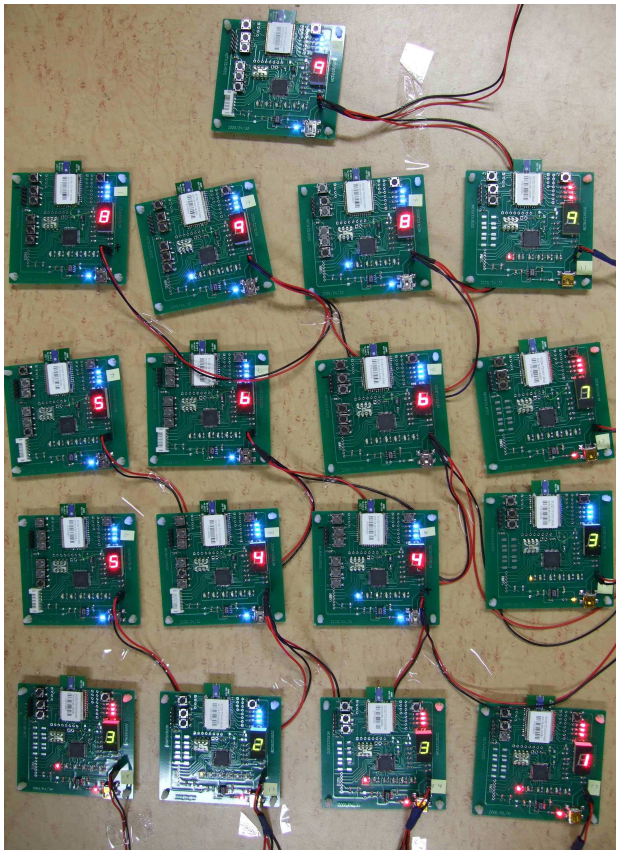


Figure 6.3: DSMS implementation result.

Chapter 7

Conclusions

We have proposed a distributed storage management strategy (DSMS) for data buffering in an isolated WSN. DSMS can reduce data loss while keep higher-priority packets closer to the sink area. Properties of DSMS are proved and its efficiency is verified by simulations. In the future, we will implement our protocol on sensor platforms and develop a storage system.

Bibliography

[1] Design and construction of a wildfire instrumentation system using networked sensors. http://firebug.sourceforge.net/.

[2] Jennic - wireless microcontroller. http://www.jennic.com/.

[3] Mica wireless measurement system. http://www.xbow.com/.

[4] Terrestrial ecology observing systems. http://research.cens.ucla.edu/.

[5] G. Anastasi, M. Conti, E. Monaldi, and A. Passarella. An adaptive data-transfer protocol for sensor networks with data mules. In *Proc. of IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM)*, 2007.

[6] A. Chakrabarti, A. Sabharwal, and B. Aazhang. Using predictable observer mobility for power efficient design of sensor network. In *Proc. of Int'l Symposium on Information Processing in Sensor Networks (IPSN)*, 2003.

[7] Y. Chen, W. Zhao, M. Ammar, and E. Zegura. Hybrid routing in clustered dtns with message ferrying. In *Proc. of ACM/SIGMOBILE Workshop on Mobile Opportunistic Networking(MobiOpp)*, 2007.

[8] C.-Y. Chong and S. P. Kumer. Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247 – 1256, 2003.

[9] C.-Y. Lin, W.-C. Peng, and Y-C. Tseng. Efficient in-network moving object tracking in wireless sensor networks. *IEEE Trans. on Mobile Computing*, 5(8):1044 – 1056, 2006.

[10] L. Luo, C. Huang, T. Abdelzaher, , and J. Stankovic. Envirostore: A cooperative storage system for disconnected operation in sensor networks. In *Proc. of IEEE INFOCOM*, 2007.

[11] R. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: Modeling a three-tier architecture for sparse sensor networks. In *Proc. of IEEE Workshop on Sensor Network Protocols and Applications (SNPA)*, 2003.

[12] A. Terzis, A. Anandarajah, K. Moore, and I.-J. Wang. Slip surface localization in wireless sensor networks for landslide prediction. In *Proc. of Int'l Symposium on Information Processing in Sensor Networks (IPSN)*, 2006.

[13] H. C. Thomas, E. L. Charles, L. R. Ronald, and S. Clifford. *Introduction to Algorithms*. The MIT Press, 2001.

[14] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke. Data collection, storage, and retrieval with an underwater sensor network. In *Proc. of ACM Int'l Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[15] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *Proc. of ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2004.

[16] W. Zhao and M. H. Ammar. Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks. In *Proc. of the 9th IEEE Workshop on Future Trends in Distributed Computing Systems(FTDCS)*, 2003.